



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

348551-007US

September 2025

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 Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries.

Other names and brands might be claimed as the property of others.

Table of Contents

| | | |
|----|---|-----------|
| | 1. About this Document | 17 |
| | 1.1. <i>Scope of this Document</i> | 17 |
| | 1.2. <i>Glossary</i> | 17 |
| 5 | 1.3. <i>Notation</i> | 18 |
| | 1.4. <i>References</i> | 18 |
| | 2. CPU Virtualization Tables | 19 |
| | 2.1. <i>MSR Virtualization</i> | 19 |
| | 2.1.1. MSR Virtualization Description Files | 19 |
| 10 | 2.1.2. IA32_ARCH_CAPABILITIES (MSR 0x10A) | 19 |
| | 2.1.3. IA32_MISC_ENABLE (MSR 0x1A0) | 20 |
| | 2.1.4. IA32_DEBUGCTL (MSR 0x1D9) | 20 |
| | 2.1.5. IA32_X2APIC_* (MSRs 0x800 – 0x8FF) | 20 |
| | 2.2. <i>CPUID Virtualization</i> | 21 |
| 15 | 2.2.1. CPUID Virtualization Description Files | 21 |
| | 3. Data Types | 24 |
| | 3.1. <i>Interface Function Completion Status</i> | 24 |
| | 3.1.1. Function Completion Status Structure | 24 |
| | 3.1.2. Function Completion Status Code Classes (Bits 47:40) | 25 |
| 20 | 3.1.3. Function Completion Status Codes and Operand IDs | 26 |
| | 3.2. <i>Basic Crypto Types</i> | 27 |
| | 3.3. <i>TDX Module Configuration, Enumeration, Initialization and Lifecycle Types</i> | 27 |
| | 3.3.1. CPUID_CONFIG..... | 27 |
| | 3.3.2. TDX Module Version | 28 |
| 25 | 3.3.3. Global-Scope (TDX Module) Metadata | 28 |
| | 3.3.3.1. TDX Features Enumeration | 28 |
| | 3.3.3.2. Global Metadata Fields | 32 |
| | 3.3.4. Global Metadata Description Files | 32 |
| | 3.3.5. CMR_INFO..... | 33 |
| 30 | 3.3.6. TDSYSINFO_STRUCT..... | 33 |
| | 3.3.7. TDMR_INFO | 36 |
| | 3.3.8. FATAL_INFO: TDX Module Fatal Error Logging Format | 37 |
| | 3.3.8.1. Overview | 37 |
| | 3.3.8.2. FATAL_INFO Basic Format..... | 38 |
| 35 | 3.3.8.3. FATAL_INFO Format with TD Handle | 38 |
| | 3.3.8.4. FATAL_INFO Format for Secure EPT Related Failures (with TD Handle)..... | 39 |
| | 3.3.8.5. FATAL_INFO Format for Secure EPT Related Failures (with EPTP) | 39 |
| | 3.3.8.6. FATAL_INFO Format for HPA Related Failures..... | 40 |
| | 3.3.8.7. FATAL_INFO Format for PAMT Related Failures | 40 |
| 40 | 3.3.8.8. FATAL_INFO Format for Unexpected Exceptions..... | 40 |
| | 3.3.8.9. FATAL_INFO Format for an Unexpected VM Exit Related Failures | 41 |
| | 3.4. <i>TD Parameter Types</i> | 41 |
| | 3.4.1. ATTRIBUTES..... | 41 |
| | 3.4.2. XFAM..... | 44 |
| 45 | 3.4.3. CONFIG_FLAGS..... | 44 |
| | 3.4.4. CPUID_VALUES..... | 46 |
| | 3.4.5. TD_PARAMS..... | 46 |
| | 3.4.6. EVENT_FILTER and the EVENT_FILTERS Array | 48 |
| | 3.5. <i>Physical Memory Management Types</i> | 49 |
| 50 | 3.5.1. PAMT Page Type (PT) Values | 49 |
| | 3.5.2. Physical Page Size..... | 50 |

| | | | |
|----|-----------|---|----|
| | 3.5.3. | HPA_LIST and HPA_LIST_INFO | 50 |
| | 3.6. | <i>TD Private Memory Management Data Types: Secure EPT</i> | 51 |
| | 3.6.1. | Secure EPT Levels | 51 |
| | 3.6.2. | Secure EPT Entry Information as Returned by TDX Module Functions..... | 51 |
| 5 | 3.6.2.1. | Returned L1 Secure EPT Entry Content..... | 52 |
| | 3.6.2.2. | Returned L2 Secure EPT Entry Content..... | 52 |
| | 3.6.2.3. | Additional Returned Secure EPT Information | 53 |
| | 3.6.3. | GPA_ATTR: GPA Attributes | 57 |
| | 3.6.3.1. | GPA Attributes Rules..... | 58 |
| 10 | 3.6.4. | GLA List | 58 |
| | 3.6.4.1. | GLA_LIST_ENTRY | 58 |
| | 3.6.4.2. | GLA_LIST | 58 |
| | 3.6.4.3. | GLA_LIST_INFO: GLA List GPA and Additional Information | 58 |
| | 3.7. | <i>TD Entry and Exit Types</i> | 59 |
| 15 | 3.7.1. | Extended Exit Qualification..... | 59 |
| | 3.8. | <i>L2 VM Transition Types</i> | 60 |
| | 3.8.1. | L2_ENTER_GUEST_STATE | 60 |
| | 3.9. | <i>Measurement and Attestation Types</i> | 61 |
| | 3.9.1. | CPUSVN..... | 61 |
| 20 | 3.9.2. | TDREPORT_STRUCT..... | 61 |
| | 3.9.3. | TEE_TCB_INFO (Reference)..... | 62 |
| | 3.9.4. | TEE_TCB_SVN (Reference) | 63 |
| | 3.9.5. | REPORTMACSTRUCT (Reference) | 63 |
| | 3.9.6. | REPORTTYPE (Reference)..... | 63 |
| 25 | 3.9.7. | TDINFO_STRUCT | 64 |
| | 3.9.8. | TDSIGSTRUCT: TD Signature Structure..... | 66 |
| | 3.9.9. | TDKEYREQUEST | 67 |
| | 3.10. | <i>Metadata Access Types</i> | 68 |
| | 3.10.1. | MD_FIELD_ID: Metadata Field Identifier / Sequence Header..... | 68 |
| 30 | 3.10.2. | Meaning of Field Codes..... | 70 |
| | 3.10.3. | Class Codes..... | 71 |
| | 3.10.3.1. | TDX Module Global Scope Field Class Codes | 71 |
| | 3.10.3.2. | TD-Scope (TDR and TDCS) Field Class Codes..... | 71 |
| | 3.10.3.3. | VCPU-Scope (TDVPS) Field Class Codes | 72 |
| 35 | 3.10.4. | Order of Field Identifiers..... | 73 |
| | 3.10.5. | MD_LIST_HEADER: Metadata List Header | 73 |
| | 3.11. | <i>Memory Buffer Types</i> | 73 |
| | 3.11.1. | Private Page List..... | 73 |
| | 3.11.2. | HPA_AND_SIZE: HPA and Size of a Buffer | 74 |
| 40 | 3.11.3. | HPA_AND_LAST: HPA and Last Byte Index of a Page-Aligned Buffer | 74 |
| | 3.12. | <i>Service TD Types</i> | 74 |
| | 3.12.1. | SERVTD_BINDING_TABLE: Service TD Binding Table | 74 |
| | 3.12.2. | SERVTD_BINDING_STATE: Service TD Binding State..... | 75 |
| | 3.12.3. | SERVTD_TYPE: Service TD Binding Type..... | 75 |
| 45 | 3.12.4. | SERVTD_ATTR: Service TD Binding Attributes..... | 75 |
| | 3.12.5. | SERVTD_EXT_STRUCT: Extended Service TD Info | 76 |
| | 3.12.6. | TEE_MODEL_STRUCT | 77 |
| | 3.12.7. | SERVTD_EXT_HASH | 77 |
| | 3.13. | <i>Migration Types</i> | 77 |
| 50 | 3.13.1. | MBMD: Migration Bundle Metadata | 78 |
| | 3.13.1.1. | Generic MBMD Structure..... | 78 |
| | 3.13.1.2. | TD-Scope Immutable Non-Memory State MBMD Fields | 79 |
| | 3.13.1.3. | TD-Scope Mutable Non-Memory State MBMD Fields | 79 |
| | 3.13.1.4. | VCPU-Scope Mutable Non-Memory State MBMD Fields..... | 79 |
| 55 | 3.13.1.5. | TD Private Memory MBMD Fields..... | 79 |
| | 3.13.1.6. | Epoch Token MBMD Fields | 80 |

| | | | |
|----|-----------|---|------------|
| | 3.13.1.7. | Abort Token MBMD Fields | 80 |
| | 3.13.1.8. | TD Migration Protocol Version Compatibility | 80 |
| | 3.13.2. | GPA List | 80 |
| | 3.13.2.1. | Overview | 80 |
| 5 | 3.13.2.2. | GPA_LIST_INFO: HPA, First and Last Entries of a GPA List | 81 |
| | 3.13.2.3. | GPA List Entry | 82 |
| | 3.13.2.4. | GPA List Entry Details | 82 |
| | 3.13.2.5. | TD Migration Protocol Version Compatibility | 86 |
| | 3.13.3. | Memory Migration Buffers List | 86 |
| 10 | 3.13.3.1. | Migration Buffers List Entry | 86 |
| | 3.13.4. | Page Attributes List | 86 |
| | 3.13.5. | Memory Migration Page MAC List | 86 |
| | 3.13.6. | Non-Memory State Migration Buffers List | 87 |
| | 3.13.6.1. | PAGE_LIST_INFO: HPA and Attributes of a Page List | 87 |
| 15 | 4. | TD Metadata (Non-Memory State)..... | 88 |
| | 4.1. | <i>TD-Scope Metadata</i> | 88 |
| | 4.1.1. | TD-Scope Metadata Description Files | 88 |
| | 4.1.2. | TDR | 89 |
| | 4.1.3. | TDCS | 89 |
| 20 | 4.1.3.1. | Overview | 89 |
| | 4.1.3.2. | TDCS.FEATURE_PARAVIRT_CTRL | 90 |
| | 4.1.3.3. | TDCS.PID_MODE[4] | 93 |
| | 4.1.3.4. | TDCS.TD_CTLs | 93 |
| | 4.1.3.5. | TDCS.VM_CTLs | 95 |
| 25 | 4.2. | <i>TDVPS: VCPU-Scope Metadata</i> | 96 |
| | 4.2.1. | Overview | 96 |
| | 4.2.2. | TDVPS (excluding TD and L2 VMCSes) | 97 |
| | 4.2.2.1. | TDVPS Description Files | 97 |
| | 4.2.2.2. | TDVPS.PIDPT_INDEX[4] | 98 |
| 30 | 4.2.2.3. | TDVPS.VCPU_STATE_DETAILS | 99 |
| | 4.2.3. | TD (L1) VMCS and L2 VMCS | 99 |
| | 4.2.3.1. | Common to TD and L2 VMCS | 99 |
| | 4.2.3.2. | TD (L1) VMCS | 100 |
| 35 | 5. | Interface Functions | 102 |
| | 5.1. | <i>How to Read the Interface Function Definitions</i> | 102 |
| | 5.2. | <i>Reserved Leaf Numbers</i> | 102 |
| | 5.3. | <i>Common Algorithms Used by Multiple Interface Functions</i> | 102 |
| | 5.3.1. | VCPU Association with an LP | 103 |
| | 5.4. | <i>Host-Side (SEAMCALL) Interface Functions</i> | 103 |
| 40 | 5.4.1. | SEAMCALL Instruction (Common) | 103 |
| | 5.4.1.1. | Instruction Description | 105 |
| | 5.4.1.2. | Completion Status Codes | 106 |
| | 5.4.2. | TDH.EXPORT.ABORT Leaf | 107 |
| | 5.4.2.1. | Input Operands | 107 |
| 45 | 5.4.2.2. | Output Operands | 107 |
| | 5.4.2.3. | Leaf Function Description | 108 |
| | 5.4.2.4. | Operands Information | 108 |
| | 5.4.2.5. | Completion Status Codes | 108 |
| | 5.4.3. | TDH.EXPORT.BLOCKW Leaf | 110 |
| 50 | 5.4.3.1. | Input Operands | 110 |
| | 5.4.3.2. | Output Operands | 110 |
| | 5.4.3.3. | Leaf Function Description | 110 |
| | 5.4.3.4. | Operands Information | 111 |
| | 5.4.3.5. | Completion Status Codes | 112 |
| 55 | 5.4.4. | TDH.EXPORT.MEM Leaf | 113 |

| | | | |
|----|-----------|---------------------------------------|-----|
| | 5.4.4.1. | Input Operands | 113 |
| | 5.4.4.2. | Output Operands | 114 |
| | 5.4.4.3. | Leaf Function Description | 114 |
| | 5.4.4.4. | Operands Information | 117 |
| 5 | 5.4.4.5. | Completion Status Codes | 118 |
| | 5.4.5. | TDH.EXPORT.PAUSE Leaf | 119 |
| | 5.4.5.1. | Input Operands | 119 |
| | 5.4.5.2. | Output Operands | 119 |
| 10 | 5.4.5.3. | Leaf Function Description | 119 |
| | 5.4.5.4. | Operands Information | 120 |
| | 5.4.5.5. | Completion Status Codes | 120 |
| | 5.4.6. | TDH.EXPORT.RESTORE Leaf | 121 |
| | 5.4.6.1. | Input Operands | 121 |
| | 5.4.6.2. | Output Operands | 121 |
| 15 | 5.4.6.3. | Leaf Function Description | 121 |
| | 5.4.6.4. | Operands Information | 122 |
| | 5.4.6.5. | Completion Status Codes | 122 |
| | 5.4.7. | TDH.EXPORT.STATE.IMMUTABLE Leaf | 124 |
| | 5.4.7.1. | Input Operands | 124 |
| 20 | 5.4.7.2. | Output Operands | 125 |
| | 5.4.7.3. | Leaf Function Description | 125 |
| | 5.4.7.4. | Operands Information | 126 |
| | 5.4.7.5. | Completion Status Codes | 126 |
| | 5.4.8. | TDH.EXPORT.STATE.TD Leaf | 128 |
| 25 | 5.4.8.1. | Input Operands | 128 |
| | 5.4.8.2. | Output Operands | 128 |
| | 5.4.8.3. | Leaf Function Description | 129 |
| | 5.4.8.4. | Operands Information | 129 |
| | 5.4.8.5. | Completion Status Codes | 130 |
| 30 | 5.4.9. | TDH.EXPORT.STATE.VP Leaf | 131 |
| | 5.4.9.1. | Input Operands | 131 |
| | 5.4.9.2. | Output Operands | 131 |
| | 5.4.9.3. | Leaf Function Description | 132 |
| | 5.4.9.4. | Operands Information | 132 |
| 35 | 5.4.9.5. | Completion Status Codes | 133 |
| | 5.4.10. | TDH.EXPORT.TRACK Leaf | 135 |
| | 5.4.10.1. | Input Operands | 135 |
| | 5.4.10.2. | Output Operands | 135 |
| | 5.4.10.3. | Leaf Function Description | 136 |
| 40 | 5.4.10.4. | Operands Information | 136 |
| | 5.4.10.5. | Completion Status Codes | 137 |
| | 5.4.11. | TDH.EXPORT.UNBLOCKW Leaf | 138 |
| | 5.4.11.1. | Input Operands | 138 |
| | 5.4.11.2. | Output Operands | 138 |
| 45 | 5.4.11.3. | Leaf Function Description | 138 |
| | 5.4.11.4. | Operands Information | 139 |
| | 5.4.11.5. | Completion Status Codes | 139 |
| | 5.4.12. | TDH.EXT.INIT Leaf | 141 |
| | 5.4.12.1. | Input Operands | 141 |
| 50 | 5.4.12.2. | Output Operands | 141 |
| | 5.4.12.3. | Leaf Function Description | 141 |
| | 5.4.12.4. | Operands Information | 141 |
| | 5.4.12.5. | Completion Status Codes | 142 |
| | 5.4.13. | TDH.EXT.MEM.ADD Leaf | 143 |
| 55 | 5.4.13.1. | Input Operands | 143 |
| | 5.4.13.2. | Output Operands | 143 |
| | 5.4.13.3. | Leaf Function Description | 143 |
| | 5.4.13.4. | Operands Information | 144 |
| | 5.4.13.5. | Completion Status Codes | 144 |
| 60 | 5.4.14. | TDH.IMPORT.ABORT Leaf | 145 |

| | | | |
|----|-----------|---------------------------------------|-----|
| | 5.4.14.1. | Input Operands | 145 |
| | 5.4.14.2. | Output Operands | 145 |
| | 5.4.14.3. | Leaf Function Description | 145 |
| | 5.4.14.4. | Operands Information | 146 |
| 5 | 5.4.14.5. | Completion Status Codes | 146 |
| | 5.4.15. | TDH.IMPORT.COMMIT Leaf | 148 |
| | 5.4.15.1. | Input Operands | 148 |
| | 5.4.15.2. | Output Operands | 148 |
| | 5.4.15.3. | Leaf Function Description | 148 |
| 10 | 5.4.15.4. | Operands Information | 148 |
| | 5.4.15.5. | Completion Status Codes | 149 |
| | 5.4.16. | TDH.IMPORT.END Leaf | 150 |
| | 5.4.16.1. | Input Operands | 150 |
| | 5.4.16.2. | Output Operands | 150 |
| 15 | 5.4.16.3. | Leaf Function Description | 150 |
| | 5.4.16.4. | Operands Information | 150 |
| | 5.4.16.5. | Completion Status Codes | 151 |
| | 5.4.17. | TDH.IMPORT.MEM Leaf | 152 |
| | 5.4.17.1. | Input Operands | 152 |
| 20 | 5.4.17.2. | Output Operands | 153 |
| | 5.4.17.3. | Leaf Function Description | 153 |
| | 5.4.17.4. | Operands Information | 156 |
| | 5.4.17.5. | Completion Status Codes | 157 |
| | 5.4.18. | TDH.IMPORT.STATE.IMMUTABLE Leaf | 158 |
| 25 | 5.4.18.1. | Input Operands | 159 |
| | 5.4.18.2. | Output Operands | 160 |
| | 5.4.18.3. | Leaf Function Description | 160 |
| | 5.4.18.4. | Operands Information | 161 |
| | 5.4.18.5. | Completion Status Codes | 161 |
| 30 | 5.4.19. | TDH.IMPORT.STATE.TD Leaf | 163 |
| | 5.4.19.1. | Input Operands | 163 |
| | 5.4.19.2. | Output Operands | 163 |
| | 5.4.19.3. | Leaf Function Description | 164 |
| | 5.4.19.4. | Operands Information | 165 |
| 35 | 5.4.19.5. | Completion Status Codes | 165 |
| | 5.4.20. | TDH.IMPORT.STATE.VP Leaf | 167 |
| | 5.4.20.1. | Input Operands | 167 |
| | 5.4.20.2. | Output Operands | 167 |
| | 5.4.20.3. | Leaf Function Description | 168 |
| 40 | 5.4.20.4. | Operands Information | 169 |
| | 5.4.20.5. | Completion Status Codes | 169 |
| | 5.4.21. | TDH.IMPORT.TRACK Leaf | 171 |
| | 5.4.21.1. | Input Operands | 171 |
| | 5.4.21.2. | Output Operands | 171 |
| 45 | 5.4.21.3. | Leaf Function Description | 171 |
| | 5.4.21.4. | Operands Information | 172 |
| | 5.4.21.5. | Completion Status Codes | 172 |
| | 5.4.22. | TDH.INTR.CONFIG Leaf | 174 |
| | 5.4.22.1. | Input Operands | 174 |
| 50 | 5.4.22.2. | Output Operands | 175 |
| | 5.4.22.3. | Leaf Function Description | 175 |
| | 5.4.22.4. | Operands Information | 177 |
| | 5.4.22.5. | Completion Status Codes | 177 |
| | 5.4.23. | TDH.MEM.PAGE.ADD Leaf | 178 |
| 55 | 5.4.23.1. | Input Operands | 178 |
| | 5.4.23.2. | Output Operands | 178 |
| | 5.4.23.3. | Leaf Function Description | 179 |
| | 5.4.23.4. | Operands Information | 179 |
| | 5.4.23.5. | Completion Status Codes | 180 |
| 60 | 5.4.24. | TDH.MEM.PAGE.AUG Leaf | 181 |

| | | | |
|----|-----------|--|-----|
| | 5.4.24.1. | Input Operands | 181 |
| | 5.4.24.2. | Output Operands | 181 |
| | 5.4.24.3. | Leaf Function Description | 182 |
| | 5.4.24.4. | Operands Information | 182 |
| 5 | 5.4.24.5. | Completion Status Codes | 183 |
| | 5.4.25. | TDH.MEM.PAGE.DEMOTE Leaf | 184 |
| | 5.4.25.1. | Input Operands | 184 |
| | 5.4.25.2. | Output Operands | 186 |
| | 5.4.25.3. | Leaf Function Description | 186 |
| 10 | 5.4.25.4. | Operands Information | 188 |
| | 5.4.25.5. | Completion Status Codes | 189 |
| | 5.4.26. | TDH.MEM.PAGE.PROMOTE Leaf..... | 190 |
| | 5.4.26.1. | Input Operands | 190 |
| | 5.4.26.2. | Output Operands | 191 |
| 15 | 5.4.26.3. | Leaf Function Description | 192 |
| | 5.4.26.4. | Operands Information | 193 |
| | 5.4.26.5. | Completion Status Codes | 194 |
| | 5.4.27. | TDH.MEM.PAGE.RELOCATE Leaf..... | 196 |
| | 5.4.27.1. | Input Operands | 196 |
| 20 | 5.4.27.2. | Output Operands | 196 |
| | 5.4.27.3. | Leaf Function Description | 197 |
| | 5.4.27.4. | Operands Information | 197 |
| | 5.4.27.5. | Completion Status Codes | 198 |
| | 5.4.28. | TDH.MEM.PAGE.REMOVE Leaf | 199 |
| 25 | 5.4.28.1. | Input Operands | 199 |
| | 5.4.28.2. | Output Operands | 199 |
| | 5.4.28.3. | Leaf Function Description | 200 |
| | 5.4.28.4. | Operands Information | 200 |
| | 5.4.28.5. | Completion Status Codes | 201 |
| 30 | 5.4.29. | TDH.MEM.RANGE.BLOCK Leaf | 202 |
| | 5.4.29.1. | Input Operands | 202 |
| | 5.4.29.2. | Output Operands | 202 |
| | 5.4.29.3. | Leaf Function Description | 203 |
| | 5.4.29.4. | Operands Information | 203 |
| 35 | 5.4.29.5. | Completion Status Codes | 204 |
| | 5.4.30. | TDH.MEM.RANGE.UNBLOCK Leaf | 205 |
| | 5.4.30.1. | Input Operands | 205 |
| | 5.4.30.2. | Output Operands | 205 |
| | 5.4.30.3. | Leaf Function Description | 206 |
| 40 | 5.4.30.4. | Operands Information | 206 |
| | 5.4.30.5. | Completion Status Codes | 207 |
| | 5.4.31. | TDH.MEM.RD Leaf..... | 208 |
| | 5.4.31.1. | Input Operands | 208 |
| | 5.4.31.2. | Output Operands | 208 |
| 45 | 5.4.31.3. | Leaf Function Description | 208 |
| | 5.4.31.4. | Operands Information | 209 |
| | 5.4.31.5. | Completion Status Codes | 209 |
| | 5.4.32. | TDH.MEM.SCAN.COMP/RANGE – Common | 210 |
| | 5.4.32.1. | GPA List-of-Lists Processing | 210 |
| 50 | 5.4.33. | TDH.MEM.SCAN.COMP Leaf | 212 |
| | 5.4.33.1. | Input Operands | 212 |
| | 5.4.33.2. | Output Operands | 213 |
| | 5.4.33.3. | Leaf Function Description | 213 |
| | 5.4.33.4. | Operands Information | 217 |
| 55 | 5.4.33.5. | Completion Status Codes | 217 |
| | 5.4.34. | TDH.MEM.SCAN.CONFIG Leaf..... | 219 |
| | 5.4.34.1. | Input Operands | 219 |
| | 5.4.34.2. | Output Operands | 220 |
| | 5.4.34.3. | Leaf Function Description | 220 |
| 60 | 5.4.34.4. | Operands Information | 221 |

| | | | |
|----|-----------|-----------------------------------|-----|
| | 5.4.34.5. | Completion Status Codes | 221 |
| | 5.4.35. | TDH.MEM.SCAN.RANGE Leaf | 223 |
| | 5.4.35.1. | Input Operands | 223 |
| | 5.4.35.2. | Output Operands | 224 |
| 5 | 5.4.35.3. | Leaf Function Description | 224 |
| | 5.4.35.4. | Operands Information | 226 |
| | 5.4.35.5. | Completion Status Codes | 227 |
| | 5.4.36. | TDH.MEM.SCAN.RESET Leaf | 228 |
| | 5.4.36.1. | Input Operands | 228 |
| 10 | 5.4.36.2. | Output Operands | 228 |
| | 5.4.36.3. | Leaf Function Description | 228 |
| | 5.4.36.4. | Operands Information | 228 |
| | 5.4.36.5. | Completion Status Codes | 229 |
| | 5.4.37. | TDH.MEM.SEPT.ADD Leaf | 230 |
| 15 | 5.4.37.1. | Input Operands | 230 |
| | 5.4.37.2. | Output Operands | 231 |
| | 5.4.37.3. | Leaf Function Description | 232 |
| | 5.4.37.4. | Operands Information | 233 |
| | 5.4.37.5. | Completion Status Codes | 234 |
| 20 | 5.4.38. | TDH.MEM.SEPT.RD Leaf | 235 |
| | 5.4.38.1. | Leaf Function Description | 236 |
| | 5.4.38.2. | Completion Status Codes | 237 |
| | 5.4.39. | TDH.MEM.SEPT.REMOVE Leaf | 238 |
| 25 | 5.4.39.1. | Input Operands | 238 |
| | 5.4.39.2. | Output Operands | 238 |
| | 5.4.39.3. | Leaf Function Description | 239 |
| | 5.4.39.4. | Operands Information | 240 |
| | 5.4.39.5. | Completion Status Codes | 241 |
| | 5.4.40. | TDH.MEM.SHARED.SEPT.WR Leaf | 242 |
| 30 | 5.4.40.1. | Leaf Function Description | 243 |
| | 5.4.40.2. | Operands Information | 243 |
| | 5.4.40.3. | Completion Status Codes | 244 |
| | 5.4.41. | TDH.MEM.TRACK Leaf | 245 |
| | 5.4.41.1. | Input Operands | 245 |
| 35 | 5.4.41.2. | Output Operands | 245 |
| | 5.4.41.3. | Leaf Function Description | 245 |
| | 5.4.41.4. | Operands Information | 245 |
| | 5.4.41.5. | Completion Status Codes | 246 |
| | 5.4.42. | TDH.MEM.WR Leaf | 247 |
| 40 | 5.4.42.1. | Input Operands | 247 |
| | 5.4.42.2. | Output Operands | 247 |
| | 5.4.42.3. | Leaf Function Description | 248 |
| | 5.4.42.4. | Operands Information | 248 |
| | 5.4.42.5. | Completion Status Codes | 248 |
| 45 | 5.4.43. | TDH.MIG.STREAM.CREATE Leaf | 249 |
| | 5.4.43.1. | Input Operands | 249 |
| | 5.4.43.2. | Output Operands | 249 |
| | 5.4.43.3. | Leaf Function Description | 249 |
| | 5.4.43.4. | Operands Information | 250 |
| 50 | 5.4.43.5. | Completion Status Codes | 250 |
| | 5.4.44. | TDH.MNG.ADDCX Leaf | 251 |
| | 5.4.44.1. | Input Operands | 251 |
| | 5.4.44.2. | Output Operands | 251 |
| | 5.4.44.3. | Leaf Function Description | 251 |
| 55 | 5.4.44.4. | Operands Information | 251 |
| | 5.4.44.5. | Completion Status Codes | 252 |
| | 5.4.45. | TDH.MNG.CREATE Leaf | 253 |
| | 5.4.45.1. | Input Operands | 253 |
| | 5.4.45.2. | Output Operands | 253 |
| 60 | 5.4.45.3. | Leaf Function Description | 253 |

| | | | |
|----|-----------|---|-----|
| | 5.4.45.4. | Operands Information | 253 |
| | 5.4.45.5. | Completion Status Codes | 254 |
| | 5.4.46. | TDH.MNG.INIT Leaf | 255 |
| 5 | 5.4.46.1. | Input Operands | 255 |
| | 5.4.46.2. | Output Operands | 255 |
| | 5.4.46.3. | Leaf Function Latency | 256 |
| | 5.4.46.4. | Leaf Function Description | 256 |
| | 5.4.46.5. | Operands Information | 256 |
| | 5.4.46.6. | Completion Status Codes | 257 |
| 10 | 5.4.47. | TDH.MNG.KEY.CONFIG Leaf | 258 |
| | 5.4.47.1. | Input Operands | 258 |
| | 5.4.47.2. | Output Operands | 258 |
| | 5.4.47.3. | Leaf Function Description | 258 |
| | 5.4.47.4. | Operands Information | 258 |
| 15 | 5.4.47.5. | Completion Status Codes | 259 |
| | 5.4.48. | TDH.MNG.KEY.FREEID Leaf | 260 |
| | 5.4.48.1. | Input Operands | 260 |
| | 5.4.48.2. | Output Operands | 260 |
| 20 | 5.4.48.3. | Leaf Function Description | 260 |
| | 5.4.48.4. | Operands Information | 260 |
| | 5.4.48.5. | Completion Status Codes | 261 |
| | 5.4.49. | TDH.MNG.KEY.RECLAIMID Leaf (Deprecated) | 262 |
| 25 | 5.4.49.1. | Input Operands | 262 |
| | 5.4.49.2. | Output Operands | 262 |
| | 5.4.49.3. | Leaf Function Description | 262 |
| | 5.4.49.4. | Completion Status Codes | 262 |
| | 5.4.50. | TDH.MNG.RD Leaf | 263 |
| 30 | 5.4.50.1. | Input Operands | 263 |
| | 5.4.50.2. | Output Operands | 263 |
| | 5.4.50.3. | Leaf Function Description | 263 |
| | 5.4.50.4. | Operands Information | 264 |
| | 5.4.50.5. | Completion Status Codes | 264 |
| | 5.4.51. | TDH.MNG.VPFLUSHDONE Leaf | 266 |
| 35 | 5.4.51.1. | Input Operands | 266 |
| | 5.4.51.2. | Output Operands | 266 |
| | 5.4.51.3. | Leaf Function Description | 266 |
| | 5.4.51.4. | Operands Information | 266 |
| | 5.4.51.5. | Completion Status Codes | 267 |
| 40 | 5.4.52. | TDH.MNG.WR Leaf | 268 |
| | 5.4.52.1. | Input Operands | 268 |
| | 5.4.52.2. | Output Operands | 268 |
| | 5.4.52.3. | Leaf Function Description | 268 |
| | 5.4.52.4. | Operands Information | 269 |
| 45 | 5.4.52.5. | Completion Status Codes | 269 |
| | 5.4.53. | TDH.MR.EXTEND Leaf | 271 |
| | 5.4.53.1. | Input Operands | 271 |
| | 5.4.53.2. | Output Operands | 271 |
| | 5.4.53.3. | Leaf Function Description | 271 |
| | 5.4.53.4. | Operands Information | 272 |
| 50 | 5.4.53.5. | Completion Status Codes | 272 |
| | 5.4.54. | TDH.MR.FINALIZE Leaf | 274 |
| | 5.4.54.1. | Input Operands | 274 |
| | 5.4.54.2. | Output Operands | 274 |
| | 5.4.54.3. | Leaf Function Description | 274 |
| 55 | 5.4.54.4. | Operands Information | 275 |
| | 5.4.54.5. | Completion Status Codes | 275 |
| | 5.4.55. | TDH.PHYMEM.CACHE.WB Leaf | 276 |
| | 5.4.55.1. | Input Operands | 276 |
| | 5.4.55.2. | Output Operands | 276 |
| 60 | 5.4.55.3. | Leaf Function Description | 276 |

| | | | |
|----|-----------|------------------------------------|-----|
| | 5.4.55.4. | Operands Information | 277 |
| | 5.4.55.5. | Completion Status Codes | 277 |
| | 5.4.56. | TDH.PHYMEM.PAGE.RDMD Leaf | 279 |
| 5 | 5.4.56.1. | Input Operands | 279 |
| | 5.4.56.2. | Output Operands | 279 |
| | 5.4.56.3. | Leaf Function Description | 281 |
| | 5.4.56.4. | Operands Information | 281 |
| | 5.4.56.5. | Completion Status Codes | 281 |
| | 5.4.57. | TDH.PHYMEM.PAGE.RECLAIM Leaf | 282 |
| 10 | 5.4.57.1. | Input Operands | 282 |
| | 5.4.57.2. | Output Operands | 282 |
| | 5.4.57.3. | Leaf Function Description | 283 |
| | 5.4.57.4. | Operands Information | 283 |
| | 5.4.57.5. | Completion Status Codes | 284 |
| 15 | 5.4.58. | TDH.PHYMEM.PAGE.WBINVD Leaf | 286 |
| | 5.4.58.1. | Input Operands | 286 |
| | 5.4.58.2. | Output Operands | 286 |
| | 5.4.58.3. | Leaf Function Description | 286 |
| | 5.4.58.4. | Operands Information | 286 |
| 20 | 5.4.58.5. | Completion Status Codes | 287 |
| | 5.4.59. | TDH.PHYMEM.PAMT.ADD Leaf | 288 |
| | 5.4.59.1. | Input Operands | 288 |
| | 5.4.59.2. | Output Operands | 288 |
| | 5.4.59.3. | Leaf Function Description | 288 |
| 25 | 5.4.59.4. | Operands Information | 289 |
| | 5.4.59.5. | Completion Status Codes | 289 |
| | 5.4.60. | TDH.PHYMEM.PAMT.REMOVE Leaf | 290 |
| | 5.4.60.1. | Input Operands | 290 |
| | 5.4.60.2. | Output Operands | 290 |
| 30 | 5.4.60.3. | Leaf Function Description | 290 |
| | 5.4.60.4. | Operands Information | 291 |
| | 5.4.60.5. | Completion Status Codes | 291 |
| | 5.4.61. | TDH.SERVTD.BIND Leaf | 292 |
| | 5.4.61.1. | Input Operands | 292 |
| 35 | 5.4.61.2. | Output Operands | 292 |
| | 5.4.61.3. | Leaf Function Description | 293 |
| | 5.4.61.4. | Operands Information | 293 |
| | 5.4.61.5. | Completion Status Codes | 293 |
| | 5.4.62. | TDH.SERVTD.REBIND Leaf | 295 |
| 40 | 5.4.62.1. | Input Operands | 295 |
| | 5.4.62.2. | Output Operands | 295 |
| | 5.4.62.3. | Leaf Function Description | 296 |
| | 5.4.62.4. | Operands Information | 296 |
| | 5.4.62.5. | Completion Status Codes | 296 |
| 45 | 5.4.63. | TDH.SERVTD.PREBIND Leaf | 298 |
| | 5.4.63.1. | Input Operands | 298 |
| | 5.4.63.2. | Output Operands | 298 |
| | 5.4.63.3. | Leaf Function Description | 298 |
| | 5.4.63.4. | Operands Information | 298 |
| 50 | 5.4.63.5. | Completion Status Codes | 299 |
| | 5.4.64. | TDH.SYS.CONFIG Leaf | 300 |
| | 5.4.64.1. | Input Operands | 300 |
| | 5.4.64.2. | Output Operands | 301 |
| | 5.4.64.3. | Leaf Function Description | 301 |
| 55 | 5.4.64.4. | Operands Information | 302 |
| | 5.4.64.5. | Completion Status Codes | 302 |
| | 5.4.65. | TDH.SYS.INFO Leaf | 303 |
| | 5.4.65.1. | Input Operands | 303 |
| | 5.4.65.2. | Output Operands | 303 |
| 60 | 5.4.65.3. | Leaf Function Description | 303 |

| | | | |
|----|-----------|---|-----|
| | 5.4.65.4. | Operands Information | 303 |
| | 5.4.65.5. | Completion Status Codes | 304 |
| | 5.4.66. | TDH.SYS.INIT Leaf | 305 |
| 5 | 5.4.66.1. | Input Operands | 305 |
| | 5.4.66.2. | Output Operands | 306 |
| | 5.4.66.3. | Leaf Function Description | 307 |
| | 5.4.66.4. | Operands Information | 307 |
| | 5.4.66.5. | Completion Status Codes | 307 |
| | 5.4.67. | TDH.SYS.KEY.CONFIG Leaf | 309 |
| 10 | 5.4.67.1. | Input Operands | 309 |
| | 5.4.67.2. | Output Operands | 309 |
| | 5.4.67.3. | Leaf Function Description | 309 |
| | 5.4.67.4. | Operands Information | 309 |
| | 5.4.67.5. | Completion Status Codes | 310 |
| 15 | 5.4.68. | TDH.SYS.LP.INIT Leaf | 311 |
| | 5.4.68.1. | Input Operands | 311 |
| | 5.4.68.2. | Output Operands | 311 |
| | 5.4.68.3. | Leaf Function Description | 312 |
| | 5.4.68.4. | Operands Information | 312 |
| 20 | 5.4.68.5. | Completion Status Codes | 312 |
| | 5.4.69. | TDH.SYS.LP.SHUTDOWN Leaf (Deprecated) | 314 |
| | 5.4.69.1. | Input Operands | 314 |
| | 5.4.69.2. | Output Operands | 314 |
| | 5.4.69.3. | Leaf Function Description | 314 |
| 25 | 5.4.69.4. | Completion Status Codes | 314 |
| | 5.4.70. | TDH.SYS.RD Leaf | 315 |
| | 5.4.70.1. | Input Operands | 315 |
| | 5.4.70.2. | Output Operands | 315 |
| | 5.4.70.3. | Leaf Function Description | 315 |
| 30 | 5.4.70.4. | Operands Information | 316 |
| | 5.4.70.5. | Completion Status Codes | 316 |
| | 5.4.71. | TDH.SYS.RDALL Leaf | 317 |
| | 5.4.71.1. | Input Operands | 317 |
| | 5.4.71.2. | Output Operands | 317 |
| 35 | 5.4.71.3. | Leaf Function Description | 317 |
| | 5.4.71.4. | Operands Information | 318 |
| | 5.4.71.5. | Completion Status Codes | 318 |
| | 5.4.72. | TDH.SYS.S4_END Leaf | 319 |
| | 5.4.72.1. | Input Operands | 319 |
| 40 | 5.4.72.2. | Output Operands | 319 |
| | 5.4.72.3. | Leaf Function Description | 319 |
| | 5.4.72.4. | Operands Information | 319 |
| | 5.4.72.5. | Completion Status Codes | 320 |
| | 5.4.73. | TDH.SYS.SHUTDOWN Leaf | 321 |
| 45 | 5.4.73.1. | Input Operands | 321 |
| | 5.4.73.2. | Output Operands | 321 |
| | 5.4.73.3. | Leaf Function Description | 321 |
| | 5.4.73.4. | Operands Information | 322 |
| | 5.4.73.5. | Completion Status Codes | 322 |
| 50 | 5.4.74. | TDH.SYS.TDMR.INIT Leaf | 324 |
| | 5.4.74.1. | Input Operands | 324 |
| | 5.4.74.2. | Output Operands | 324 |
| | 5.4.74.3. | Leaf Function Description | 324 |
| | 5.4.74.4. | Operands Information | 324 |
| 55 | 5.4.74.5. | Completion Status Codes | 325 |
| | 5.4.75. | TDH.SYS.UPDATE Leaf | 326 |
| | 5.4.75.1. | Input Operands | 326 |
| | 5.4.75.2. | Output Operands | 327 |
| | 5.4.75.3. | Leaf Function Description | 327 |
| 60 | 5.4.75.4. | Operands Information | 328 |

| | | | |
|----|-----------|--|-----|
| | 5.4.75.5. | Completion Status Codes | 328 |
| | 5.4.76. | TDH.VP.ADDCX Leaf | 329 |
| | 5.4.76.1. | Input Operands | 329 |
| | 5.4.76.2. | Output Operands | 329 |
| 5 | 5.4.76.3. | Leaf Function Description | 329 |
| | 5.4.76.4. | Operands Information | 329 |
| | 5.4.76.5. | Completion Status Codes | 330 |
| | 5.4.77. | TDH.VP.CREATE Leaf | 331 |
| | 5.4.77.1. | Input Operands | 331 |
| 10 | 5.4.77.2. | Output Operands | 331 |
| | 5.4.77.3. | Leaf Function Description | 331 |
| | 5.4.77.4. | Operands Information | 331 |
| | 5.4.77.5. | Completion Status Codes | 332 |
| | 5.4.78. | TDH.VP.ENTER Leaf | 333 |
| 15 | 5.4.78.1. | Input Operands | 333 |
| | 5.4.78.2. | Output Operands | 335 |
| | 5.4.78.3. | Leaf Function Description | 340 |
| | 5.4.78.4. | Operands Information | 343 |
| | 5.4.78.5. | Completion Status Codes | 343 |
| 20 | 5.4.79. | TDH.VP.FLUSH Leaf | 345 |
| | 5.4.79.1. | Input Operands | 345 |
| | 5.4.79.2. | Output Operands | 345 |
| | 5.4.79.3. | Leaf Function Description | 345 |
| | 5.4.79.4. | Operands Information | 345 |
| 25 | 5.4.79.5. | Completion Status Codes | 346 |
| | 5.4.80. | TDH.VP.INIT Leaf | 347 |
| | 5.4.80.1. | Input Operands | 347 |
| | 5.4.80.2. | Output Operands | 347 |
| | 5.4.80.3. | Leaf Function Description | 347 |
| 30 | 5.4.80.4. | Operands Information | 348 |
| | 5.4.80.5. | Completion Status Codes | 348 |
| | 5.4.81. | TDH.VP.RD Leaf | 350 |
| | 5.4.81.1. | Input Operands | 350 |
| | 5.4.81.2. | Output Operands | 350 |
| 35 | 5.4.81.3. | Leaf Function Description | 350 |
| | 5.4.81.4. | Operands Information | 351 |
| | 5.4.81.5. | Completion Status Codes | 351 |
| | 5.4.82. | TDH.VP.WR Leaf | 353 |
| | 5.4.82.1. | Input Operands | 353 |
| 40 | 5.4.82.2. | Output Operands | 353 |
| | 5.4.82.3. | Leaf Function Description | 353 |
| | 5.4.82.4. | Operands Information | 354 |
| | 5.4.82.5. | Completion Status Codes | 354 |
| | 5.5. | <i>Guest-Side (TDCALL) Interface Functions</i> | 356 |
| 45 | 5.5.1. | TDCALL Instruction (Common) | 356 |
| | 5.5.1.1. | Input Operands | 356 |
| | 5.5.1.2. | Output Operands | 356 |
| | 5.5.1.3. | Instruction Description | 356 |
| | 5.5.1.4. | TDCALL Leaf Numbers | 357 |
| 50 | 5.5.1.5. | Completion Status Codes | 357 |
| | 5.5.2. | TDG.INTR.POST | 358 |
| | 5.5.2.1. | Input Operands | 358 |
| | 5.5.2.2. | Output Operands | 358 |
| | 5.5.2.3. | Leaf Function Description | 358 |
| 55 | 5.5.2.4. | Operands Information | 359 |
| | 5.5.2.5. | Completion Status Codes | 359 |
| | 5.5.3. | TDG.MEM.PAGE.ACCEPT Leaf | 360 |
| | 5.5.3.1. | Input Operands | 360 |
| | 5.5.3.2. | Output Operands | 360 |

| | | | |
|----|-----------|---------------------------------|-----|
| | 5.5.3.3. | Leaf Function Description | 360 |
| | 5.5.3.4. | Operands Information | 361 |
| | 5.5.3.5. | Completion Status Codes | 361 |
| | 5.5.4. | TDG.MEM.PAGE.ATTR.RD Leaf | 363 |
| 5 | 5.5.4.1. | Input Operands | 363 |
| | 5.5.4.2. | Output Operands | 363 |
| | 5.5.4.3. | Leaf Function Description | 364 |
| | 5.5.4.4. | Operands Information | 364 |
| | 5.5.4.5. | Completion Status Codes | 365 |
| 10 | 5.5.5. | TDG.MEM.PAGE.ATTR.WR Leaf | 366 |
| | 5.5.5.1. | Input Operands | 366 |
| | 5.5.5.2. | Output Operands | 367 |
| | 5.5.5.3. | Leaf Function Description | 368 |
| | 5.5.5.4. | Operands Information | 369 |
| 15 | 5.5.5.5. | Completion Status Codes | 369 |
| | 5.5.6. | TDG.MEM.PAGE.RELEASE Leaf..... | 370 |
| | 5.5.6.1. | Input Operands | 370 |
| | 5.5.6.2. | Output Operands | 370 |
| | 5.5.6.3. | Leaf Function Description | 371 |
| 20 | 5.5.6.4. | Operands Information | 372 |
| | 5.5.6.5. | Completion Status Codes | 372 |
| | 5.5.7. | TDG.MR.ASSIGNSVNS | 374 |
| | 5.5.7.1. | Input Operands | 374 |
| | 5.5.7.2. | Output Operands | 374 |
| 25 | 5.5.7.3. | Leaf Function Description | 374 |
| | 5.5.7.4. | Operands Information | 375 |
| | 5.5.7.5. | Completion Status Codes | 375 |
| | 5.5.8. | TDG.MR.KEY.GET..... | 376 |
| | 5.5.8.1. | Input Operands | 376 |
| 30 | 5.5.8.2. | Output Operands | 376 |
| | 5.5.8.3. | Leaf Function Description | 376 |
| | 5.5.8.4. | Operands Information | 377 |
| | 5.5.8.5. | Completion Status Codes | 377 |
| | 5.5.9. | TDG.MR.REPORT Leaf | 378 |
| 35 | 5.5.9.1. | Input Operands | 378 |
| | 5.5.9.2. | Output Operands | 378 |
| | 5.5.9.3. | Leaf Function Description | 378 |
| | 5.5.9.4. | Operands Information | 379 |
| | 5.5.9.5. | Completion Status Codes | 380 |
| 40 | 5.5.10. | TDG.MR.RTMR.EXTEND Leaf..... | 381 |
| | 5.5.10.1. | Input Operands | 381 |
| | 5.5.10.2. | Output Operands | 381 |
| | 5.5.10.3. | Leaf Function Description | 381 |
| | 5.5.10.4. | Operands Information | 381 |
| 45 | 5.5.10.5. | Completion Status Codes | 382 |
| | 5.5.11. | TDG.MR.VERIFYREPORT | 383 |
| | 5.5.11.1. | Input Operands | 383 |
| | 5.5.11.2. | Output Operands | 383 |
| | 5.5.11.3. | Leaf Function Description | 383 |
| 50 | 5.5.11.4. | Operands Information | 383 |
| | 5.5.11.5. | Completion Status Codes | 384 |
| | 5.5.12. | TDG.SERVTD.RD Leaf..... | 385 |
| | 5.5.12.1. | Input Operands | 385 |
| | 5.5.12.2. | Output Operands | 385 |
| 55 | 5.5.12.3. | Leaf Function Description | 386 |
| | 5.5.12.4. | Operands Information | 386 |
| | 5.5.12.5. | Completion Status Codes | 387 |
| | 5.5.13. | TDG.SERVTD.REBIND.APPROVE | 389 |
| | 5.5.13.1. | Input Operands | 389 |
| 60 | 5.5.13.2. | Output Operands | 389 |

| | | | |
|----|-----------|---------------------------------|-----|
| | 5.5.13.3. | Leaf Function Description | 389 |
| | 5.5.13.4. | Operands Information | 390 |
| | 5.5.13.5. | Completion Status Codes | 391 |
| | 5.5.14. | TDG.SERVTD.WR Leaf..... | 393 |
| 5 | 5.5.14.1. | Input Operands | 393 |
| | 5.5.14.2. | Output Operands | 393 |
| | 5.5.14.3. | Leaf Function Description | 394 |
| | 5.5.14.4. | Operands Information | 394 |
| | 5.5.14.5. | Completion Status Codes | 395 |
| 10 | 5.5.15. | TDG.SYS.RD Leaf..... | 397 |
| | 5.5.15.1. | Input Operands | 397 |
| | 5.5.15.2. | Output Operand..... | 397 |
| | 5.5.15.3. | Leaf Function Description | 397 |
| | 5.5.15.4. | Operands Information | 398 |
| 15 | 5.5.15.5. | Completion Status Codes | 398 |
| | 5.5.16. | TDG.SYS.RDALL Leaf | 399 |
| | 5.5.16.1. | Input Operands | 399 |
| | 5.5.16.2. | Output Operands | 399 |
| | 5.5.16.3. | Leaf Function Description | 399 |
| 20 | 5.5.16.4. | Operands Information | 400 |
| | 5.5.16.5. | Completion Status Codes | 400 |
| | 5.5.17. | TDG.VM.RD Leaf..... | 401 |
| | 5.5.17.1. | Input Operands | 401 |
| | 5.5.17.2. | Output Operands | 401 |
| 25 | 5.5.17.3. | Leaf Function Description | 401 |
| | 5.5.17.4. | Operands Information | 402 |
| | 5.5.17.5. | Completion Status Codes | 402 |
| | 5.5.18. | TDG.VM.WR Leaf | 403 |
| | 5.5.18.1. | Input Operands | 403 |
| 30 | 5.5.18.2. | Output Operands | 403 |
| | 5.5.18.3. | Leaf Function Description | 403 |
| | 5.5.18.4. | Operands Information | 404 |
| | 5.5.18.5. | Completion Status Codes | 404 |
| | 5.5.19. | TDG.VP.CPUIDVE.SET Leaf | 405 |
| 35 | 5.5.19.1. | Input Operands | 405 |
| | 5.5.19.2. | Output Operands | 405 |
| | 5.5.19.3. | Leaf Function Description | 405 |
| | 5.5.19.4. | Operands Information | 405 |
| | 5.5.19.5. | Completion Status Codes | 406 |
| 40 | 5.5.20. | TDG.VP.ENTER Leaf | 407 |
| | 5.5.20.1. | Input Operands | 407 |
| | 5.5.20.2. | Output Operands | 408 |
| | 5.5.20.3. | Leaf Function Description | 410 |
| | 5.5.20.4. | Operands Information | 411 |
| 45 | 5.5.20.5. | Completion Status Codes | 412 |
| | 5.5.21. | TDG.VP.INFO Leaf | 413 |
| | 5.5.21.1. | Input Operands | 413 |
| | 5.5.21.2. | Output Operands | 413 |
| | 5.5.21.3. | Leaf Function Description | 414 |
| 50 | 5.5.21.4. | Operands Information | 414 |
| | 5.5.21.5. | Completion Status Codes | 414 |
| | 5.5.22. | TDG.VP.INVEPT Leaf..... | 415 |
| | 5.5.22.1. | Input Operands | 415 |
| | 5.5.22.2. | Output Operands | 415 |
| 55 | 5.5.22.3. | Leaf Function Description | 415 |
| | 5.5.22.4. | Operands Information | 416 |
| | 5.5.22.5. | Completion Status Codes | 416 |
| | 5.5.23. | TDG.VP.INVGLA Leaf | 417 |
| | 5.5.23.1. | Input Operands | 417 |
| 60 | 5.5.23.2. | Output Operands | 417 |

| | | | |
|----|-----------|---------------------------------|-----|
| | 5.5.23.3. | Leaf Function Description | 418 |
| | 5.5.23.4. | Operands Information | 418 |
| | 5.5.23.5. | Completion Status Codes | 418 |
| | 5.5.24. | TDG.VP.RD Leaf | 420 |
| 5 | 5.5.24.1. | Input Operands | 420 |
| | 5.5.24.2. | Output Operands | 420 |
| | 5.5.24.3. | Leaf Function Description | 420 |
| | 5.5.24.4. | Operands Information | 421 |
| | 5.5.24.5. | Completion Status Codes | 421 |
| 10 | 5.5.25. | TDG.VP.VEINFO.GET Leaf | 422 |
| | 5.5.25.1. | Input Operands | 422 |
| | 5.5.25.2. | Output Operands | 422 |
| | 5.5.25.3. | Leaf Function Description | 423 |
| | 5.5.25.4. | Operands Information | 424 |
| 15 | 5.5.25.5. | Completion Status Codes | 424 |
| | 5.5.26. | TDG.VP.VMCALL Leaf | 425 |
| | 5.5.26.1. | Input Operands | 425 |
| | 5.5.26.2. | Output Operands | 426 |
| | 5.5.26.3. | Leaf Function Description | 426 |
| 20 | 5.5.26.4. | Operands Information | 426 |
| | 5.5.26.5. | Completion Status Codes | 427 |
| | 5.5.27. | TDG.VP.WR Leaf | 428 |
| | 5.5.27.1. | Input Operands | 428 |
| | 5.5.27.2. | Output Operands | 428 |
| 25 | 5.5.27.3. | Leaf Function Description | 428 |
| | 5.5.27.4. | Operands Information | 429 |
| | 5.5.27.5. | Completion Status Codes | 429 |

1. About this Document

1.1. Scope of this Document

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

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

Table 1.1: TDX Module Architecture Specification Set

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

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

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

1.2. Glossary

See the [TDX Module Base Spec].

1.3. Notation

See the [TDX Module Base Spec].

1.4. References

See the [TDX Module Base Spec].

2. CPU Virtualization Tables

2.1. MSR Virtualization

2.1.1. MSR Virtualization Description Files

Most of the MSR virtualization information is provided in separate files:

- 5 • **msr_virtualization.pdf** provides a human-readable MSR virtualization table.
- **msr_virtualization.csv** provides the same table in a Comma-Separated Values (CSV) format. This file can be imported, if required, to tools such as MS Excel for further processing.

The tables below describe the information provided by those files. Additional information about specific MSRs is provided in the following sections.

- 10 In addition to the above files, **msr_virtualization.json**, a JSON format file with the same information is provided.

Table 2.1: MSR Virtualization Table Description

| Column | Description |
|------------------------|--|
| First (H) | Index of the first MSR in the range |
| Last (H) | Index of the last MSR in the range |
| Size (H) | Number of MSRs in the range |
| MSR Architectural Name | MSR name |
| On RDMSR | MSR virtualization on read; see the table below |
| On WRMSR | MSR virtualization on write; see the table below |

Table 2.2: On RDMSR / On WRMSR Column Details

| MSR Virtualization | Description |
|----------------------------|---|
| Native | Direct read or write from/to CPU |
| #GP(0) | Inject a #GP(0) exception |
| #VE | For L1: Inject a #VE(NON_CONFIG_PARAVIRT) exception For L2: L2->L1 |
| Inject_GP(condition) | If the condition is true, the TDX Module injects a #GP(0). Else, the TDX Module reads from CPU or writes to the CPU: <pre> if (condition) #GP(0) else Native </pre> |
| Inject_GP_or_VE(condition) | If the condition is true, the TDX Module injects a #GP(0). Else, the TDX Module injects a #VE or does an L2->L1 exit as described above: <pre> if (condition) #GP(0) else if L1, #VE(CONFIG_PARAVIRT) if L2, L2->L1 exit </pre> |

15 2.1.2. IA32_ARCH_CAPABILITIES (MSR 0x10A)

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

2.1.3. IA32_MISC_ENABLE (MSR 0x1A0)

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

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

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

Table 2.3: IA32_MISC_ENABLE (MSR 0x1A0) Virtualization when TDCS.TD_CTL.S.REDUCE_VE is Set to 1

| Bit | Name | Access | Init Value of Shadow (in TDVPS) | On RDMSR | On WRMSR | Description |
|-------|--|--------|---------------------------------|-------------|---|---|
| 0 | Fast-Strings Enable | RW | Native value | From shadow | To shadow | |
| 3 | Automatic Thermal Control Circuit Enable | RW | 0 | From shadow | To shadow | |
| 7 | Perfmon Available | RO | ATTRIBUTES.PERFMON | From shadow | Ignore | |
| 11 | BTS Unavailable | RO | Native (checked to be 1) | From shadow | Ignore | |
| 12 | PEBS Unavailable | RO | Native & ~ATTRIBUTES.PERFMON | From shadow | Ignore | |
| 16 | Enhanced Speed Step | RW | Virt. CPUID(1).ECX[7] | From shadow | If (virt. CPUID(1).ECX[7] == 0) and (value == 1), #GP. Else, write to shadow | Support paravirtualization |
| 18 | Enable MONITOR FSM | RW | Virt. CPUID(1).ECX[3] | From shadow | If (value is being modified), #VE(UNSUPPORTED_FEATURE). Else, write to shadow | Guest TD is not expected to change this bit. |
| 22 | Limit CPUID Max Leaf | RW | 0 | From shadow | If (value == 1), #VE(UNSUPPORTED_FEATURE). Value in shadow remains 0. | Simplify CPUID handling, not supposed to happen with modern OS |
| 23 | xTPR Message Disable | RW | 0 | From shadow | If (virt. CPUID(1).ECX[14] == 0) and (value == 1), #GP. Else, write to shadow | TDs are not allowed to broadcast IPIs. Host VMM can control this bit. |
| 34 | XD Bit Disable | RW | 0 | From shadow | If (value == 1), #GP. Value in shadow remains 0. | This bit is deprecated. |
| Other | Reserved | RO | 0 | From shadow | If (value == 1), #GP. Value in shadow remains 0. | |

10

2.1.4. IA32_DEBUGCTL (MSR 0x1D9)

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

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

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

2.2. CPUID Virtualization

2.2.1. CPUID Virtualization Description Files

Most of the MSR virtualization information is provided in separate files:

- **cpuid_virtualization.pdf** provides a human-readable CPUID virtualization table.
- **cpuid_virtualization.csv** provides the same table in a Comma-Separated Values (CSV) format. This file can be imported, if required, to tools such as MS Excel for further processing.

The tables below describe the information provided by those files.

In addition to the above files, **cpuid_virtualization.json**, a JSON format file with the same information is provided.

Table 2.4: CPUID Virtualization Table Description

| Column | Description |
|----------------------------------|---|
| Leaf | Leaf number (EAX input to the CPUID instruction) |
| Sub-Leaf From | First sub-leaf number (ECX input to the CPUID instruction) in the range |
| Sub-Leaf To | Last sub-leaf number (ECX input to the CPUID instruction) in the range |
| Header | Leaf: Line is a leaf header Sub-Leaf: Line is a sub-leaf header Field: Line defines a bit field |
| Reg | Output register of the CPUID instruction |
| MSB | Most significant bit of the bit field |
| LSB | Least significant bit of the bit field |
| Field Size | Size of the bit field |
| Field Name | Name of the bit field |
| Configuration Details | Fields of TD_PARAMS input to TDH.MNG.INIT that are used to configure the bit fields |
| Bit or Field Virtualization Type | See the table below |
| Virtualization Details | See the table below |

Table 2.5: Bit or Field Virtualization Type and Details

| Bit or Field Virtualization | Meaning | Meaning of Virtualization Details |
|-------------------------------------|---|--------------------------------------|
| #VE | #VE injected to the TD | N/A |
| Attributes & Native | Bits that the host VMM directly configures at guest TD initialization, based on the guest TD's ATTRIBUTES, to either allow their native value or to have a value of 0. | Applicable ATTRIBUTES bit |
| Attributes & CPUID_Enabled & Native | Bits that the host VMM configures at guest TD initialization, by configuring the guest TD's ATTRIBUTES and the virtual value of some other CPUID field, to either allow their native value or to have a value of 0. These are typically CPU features that the host VMM can disable. | Applicable ATTRIBUTES and CPUID bits |

| Bit or Field Virtualization | Meaning | Meaning of Virtualization Details |
|---|---|---|
| Attributes & CPUID_Enabled & Configured & Native | Bits that the host VMM directly configures at guest TD initialization, based on the guest TD's ATTRIBUTES, the virtual value of some other CPUID field and direct configuration, to either allow their native value or to have a value of 0. | Applicable ATTRIBUTES bit and CPUID bit |
| Attributes & CPUID_Enabled & Native(Hybrid) | Bits that the host VMM configures at guest TD initialization, by configuring the guest TD's ATTRIBUTES and the virtual value of some other CPUID field, to either allow their native value or to have a value of 0. These are typically CPU features that the host VMM can disable. On Hybrid SOCs, value is uniform across all core types. | Applicable ATTRIBUTES and CPUID bits |
| Attributes & CPUID_Enabled & Configured & Native(Hybrid) | Bits that the host VMM directly configures at guest TD initialization, based on the guest TD's ATTRIBUTES, the virtual value of some other CPUID field and direct configuration, to either allow their native value or to have a value of 0. On Hybrid SOCs, value is uniform across all core types. | Applicable ATTRIBUTES bit and CPUID bit |
| Attributes & Configured & Native | Bits that the host VMM directly configures at guest TD initialization, based on the guest TD's ATTRIBUTES and direct configuration, to either allow their native value or to have a value of 0. | Applicable ATTRIBUTES bit |
| Attributes & Native (Hybrid) | Bits that the host VMM directly configures at guest TD initialization, based on the guest TD's ATTRIBUTES, to either allow their native value or to have a value of 0. On Hybrid SOCs, value is uniform across all core types. | Applicable ATTRIBUTES bit |
| CPUID_Enabled & Native | Bits that the host VMM configures at guest TD initialization, by configuring the virtual value of some other CPUID field, to either allow their native value or to have a value of 0. These are typically CPU features that the host VMM can disable. | Applicable CPUID bit |
| CPUID_Enabled (Dynamic) & Native | Bits that the host VMM configures at guest TD initialization, by configuring the virtual value of some other CPUID field, to either allow their native value or to have a value of 0. These are typically CPU features that either the host VMM or the guest TD can disable. | Applicable CPUID bit |
| Configured & Native | Bits that the host VMM directly configures at guest TD initialization, to either allow their native value or to have a value of 0. These are typically CPU features that the host VMM can disable. | N/A |
| Configured & Native | Bits that the host VMM directly configures at guest TD initialization, to either allow their native value or to have a value of 0. These are typically CPU features that the host VMM can disable. Special handling is applied. | N/A |
| XFAM & Native | Bits that the host VMM directly configures at guest TD initialization, based on the guest TD's XFAM, to either allow their native value or to have a value of 0. | Applicable XFAM bits |
| XFAM & CPUID_Enabled & Native | Bits that the host VMM directly configures at guest TD initialization, based on the guest TD's XFAM and the virtual value of some other CPUID field, to either allow their native value or to have a value of 0. | Applicable XFAM bits and CPUID bit |

| Bit or Field Virtualization | Meaning | Meaning of Virtualization Details |
|---|---|---|
| XFAM & CPUID_Enabled & Configured & Native | Bits that the host VMM directly configures at guest TD initialization, based on both the guest TD's XFAM, the virtual value of some other CPUID field and direct configuration, to either allow their native value or to have a value of 0. | Applicable XFAM bits and CPUID bit |
| XFAM & Configured & Native | Bits that the host VMM directly configures at guest TD initialization, based on both the guest TD's XFAM and direct configuration, to either allow their native value or to have a value of 0. | Applicable XFAM bits |
| XFAM & Native (Hybrid) | Bits that the host VMM directly configures at guest TD initialization, based on the guest TD's XFAM, to either allow their native value or to have a value of 0. On Hybrid SOCs, value is uniform across all core types. | Applicable XFAM bits |
| Calculated | Bit field is computed by the Intel TDX Module at TDH.SYS.INIT time | Calculation details |
| Configured | Bits fields that the host VMM directly configures at guest TD initialization | N/A |
| Configured | Bit fields that the host VMM directly configures at guest TD initialization, based on TD_PARAMS fields other than the above. | Applicable TD_PARAMS field |
| XFAM | Bits that the host VMM directly configures at guest TD initialization, based on the guest TD's XFAM. | Applicable XFAM bits |
| Calculated | Bit field is computed by the Intel TDX Module at TD run time | Calculation details |
| Fixed | Bit field is always reported as a fixed value to the TD. | Fixed value |
| Fixed | Bit field is always reported as a fixed value to the TD. Identical to FIXED, except that the init check must be VERIFY and the fixed value is taken from the init check value. | N/A, value is taken from the init check details |
| Native @ TD Init | Bit field value reflects the native value returned by executing CPUID at the time the TD was initialized (TDH.MNG.INIT) | N/A |
| Special | Bit field value is calculated in a special way | Special calculation details |
| Special | Bit field value is calculated in a special way, based on direct configuration by the host VMM | Special calculation details |
| Native @ TD Init with Special Handling | Bit field value reflects the native value returned by executing CPUID at the time the TD was initialized (TDH.MNG.INIT). Special handling is applied. | Special calculation details |

3. Data Types

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

3.1. Interface Function Completion Status

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

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

3.1.1. Function Completion Status Structure

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

| Bits | Name | Description |
|-------|--------------------------|---|
| 63 | ERROR | Interface function aborted due to error. 0: Indicates that the function completed successfully – possibly with some warnings. 1: Indicates that the function aborted due to some error. |
| 62 | NON_RECOVERABLE | Recoverability hint: 0: Indicates that the function may possibly be retried after some conditions have been corrected. 1: Indicates that the error is probably not recoverable. In most cases, NON_RECOVERABLE is set only if ERROR is 1. However, TDH.VP.ENTER is a special case. TD entry may succeed (thus ERROR is 0) but later the TD failed in a non-recoverable way (thus NON_RECOVERABLE is 1). |
| 61 | FATAL | Fatality hint – applicable only for SEAMCALL. 0: Indicates that the TD can continue its normal lifecycle. 1: Indicates that the TD entered a state where it can only be torn down. E.g., when an import has failed, and the TD's OP_STATE is FAILED_IMPORT. |
| 60 | HOST_RECOVERABILITY_HINT | As a TDH.VP.ENTER output, indicates a TDCALL that resulted in a trap-like TD exit for which the host VMM needs to provide a recoverability hint in the following TD entry. On the following TDH.VP.ENTER, the host VMM provides a hint to the guest TD, which is the output of the TDCALL: 0: The host VMM hints that the guest-side function may possibly be retried (e.g., the host may have corrected some conditions). 1: The host VMM hints that the error is probably not recoverable. |
| 59:48 | RESERVED | Reserved – set to 0 |
| 47:40 | CLASS | Class of the function completion status |
| 39:32 | DETAILS_L1 | Details of the function completion status |

| Bits | Name | Description |
|------|------------|---|
| 31:0 | DETAILS_L2 | Additional details of the function completion status – e.g., includes: <ul style="list-style-type: none"> • Implicit or explicit operand identifier • CPUID leaf or sub-leaf • MSR index • VMCS field code • VM exit reason • CMR index • TDMR index |

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

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

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

3.1.3. Function Completion Status Codes and Operand IDs

Interface functions completion status codes and operand IDs are provided in separate files:

- **interface_functions_completion_status_details.pdf** and **interface_functions_completion_status_operand_ids.pdf** provides human-readable tables of the completion statuses and the operand IDs.
- **interface_functions_completion_status_details.csv** and **interface_functions_completion_status_operand_ids.csv** provides the same tables in a Comma-Separated Values (CSV) format. These files can be imported, if required, to tools such as MS Excel for further processing.

The tables below describe the information provided by those files.

In addition to the above files, **interface_functions_completion_status.json**, a JSON format file with the same information is provided.

Table 3.3: Function Completion Status Table Description

| Column | Description | |
|-------------------------------------|---|--|
| Base Value (Bits 63:32, Hex) | Base value of the status code | |
| Status | Value | Details |
| | Success | The operation is successful. |
| | Error | The operation resulted in an error. |
| | Recover. Error | The operation resulted in an error but may be retried by the caller. |
| | Host Recover. Error | The operation resulted in an error but may be retried after some host VMM recovery. |
| | Non-Recover. | TD exit operation is successful from the TDX Module perspective, but VCPU state is non-recoverable due to TD s/w operation that caused a triple-fault condition. |
| | Fatal | The operation resulted in a fatal error; the TD is disabled and should be torn down |
| Name | Status name | |
| DETAILS_L2 (Bits 31:0) | Additional details of the function completion status. For Operand IDs, see the table below. | |
| Description | Status description | |

Table 3.4: Function Completion Operand ID Table Description

| Column | Description |
|--------------------------|---|
| Operand ID | Operand ID value |
| Explicit/Implicit | Whether the operand was an explicit input of the interface function |
| Operand | Operand name |
| Description | Operand description |

3.2. Basic Crypto Types

Table 3.5: 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 |

3.3. TDX Module Configuration, Enumeration, Initialization and Lifecycle Types

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

3.3.1. CPUID_CONFIG

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

Table 3.6: CPUID_CONFIG Definition

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

3.3.2. TDX Module Version

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

Table 3.7: TDX Module Version Definition

| Field | As Text | Size (Bytes) | Description |
|---|-------------------------------------|--------------|--|
| MAJOR_VERSION and MINOR_VERSION | 1 digit each, e.g., “1.5” | 16 bits each | Together, represent the main version number of the TDX Module. Usually related to the supported SOCs. |
| UPDATE_VERSION | 2 digits, e.g., “1.5.08” | 16 bits | A sub-version of the major/minor version. The update version number is incremented after every production-signed drop of the TDX Module. E.g., if version 1.5.01 is production-signed, the next drop will not use 01 as its update version, regardless of being a production-signed drop or not. |
| INTERNAL_VERSION | 2 digits, e.g., “1.5.08.04” | 16 bits | A sub-version of the update version. Denotes internal release number. |
| BUILD_NUM | 4 digits, e.g., “1.5.08.04.0234” | 16 bits | A unique build number |

5

3.3.3. Global-Scope (TDX Module) Metadata

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

3.3.3.1. TDX Features Enumeration

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

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

Table 3.8: TDX_FEATURES0 Definition

| Bit(s) | Name | Description |
|--------|-------------------|---|
| 0 | TD_MIGRATION | The TDX Module supports TD migration. Further information is provided by the Migration fields. |
| 1 | TD_PRESERVING | The TDX Module supports TD preserving updates. Further information is provided by the TDX Module Handoff metadata fields. |
| 2 | SERVICE_TD | The TDX Module supports Service TDs. Further information is provided by the Service TD fields. |
| 3 | ENHANCED_METADATA | The TDX Module supports enhanced metadata interface functions: <ul style="list-style-type: none"> Version 1 of previously existing functions: TDH.MNG.RD, TDH.VP.RD and TDG.VM.RD. New functions: TDH.SYS.RD, TDH.SYS.RDALL, TDG.SYS.RD, TDG.SYS.RDALL, TDG.VP.RD/WR. |
| 4 | RELAXED_MEM_MNG | The TDX Module’s memory management requirements are relaxed vs. TDX 1.0: |

| Bit(s) | Name | Description |
|--------|-------------------------------|---|
| | | <ul style="list-style-type: none"> Many interface functions allow concurrent memory management operations by exclusively locking specific Secure EPT entries instead of the whole Secure EPT tree: <ul style="list-style-type: none"> TDH.MEM.PAGE.AUG/DEMOTE/PROMOTE/RELOCATE/REMOVE TDH.MEM.SEPT.ADD TLB tracking (e.g., TDH.MEM.RANGE.BLOCK followed by TDH.MEM.TRACK and IPIs) may be skipped if the TD's OP_STATE is such that the TD can't be running, i.e., in the following cases: <ul style="list-style-type: none"> On normal TD build, the TD's measurement has not yet been finalized by TDH.MR.FINALIZED. The TD has been paused for export by TDH.EXPORT.PAUSE, and export has not been aborted by TDH.EXPOR.ABORT. On import, TD execution has been enabled by TDH.IMPORT.COMMIT or TDH.IMPORT.END. |
| 5 | CPUID_VIRT_GUEST_CTRL | Guest TD may request that on certain CPUID leaves/sub-leaves a #VE(CONFIG_PARAVIRT) will always be injected, using the TDG.VP.RD/WR to access the TDVPS' CPUID_CONTROL fields. |
| 6 | TDX_CONNECT | Both the TDX Module and the CPU support TDX Connect, and TDX Connect has been enabled. |
| 7 | TD_PARTITIONING | <p>The TDX Module supports TD partitioning:</p> <ul style="list-style-type: none"> New interface functions: TDG.MEM,PAGE.ATTR.RD/WR, TDG.VP.ENTER, TDG.VP.INVEPT, TDG.VP.INVGLA Version 1 of existing interface functions: TDH.MEM.PAGE.PROMOTE, TDH.MEM.SEPT.ADD, TDH.MEM.SEPT.REMOVE Backward-compatible updates to existing interface functions, with new input and/or output operands. |
| 8 | LOCAL_ATTESTATION | <p>The TDX Module supports local attestation:</p> <ul style="list-style-type: none"> New interface function: TDG.MR.VERIFYREPORT |
| 9 | TD_ENTRY_ENHANCEMENTS | <p>The TDX Module supports the following TD entry enhancements:</p> <ul style="list-style-type: none"> HOST_RECOVERABILITY_HINT: On trap-like asynchronous TD exit, bit 60 of the TDH.VP.ENTER completion status (returned in RAX) may be set to 1. In this case, the host VMM may set the following TD entry's input value of RCX' HOST_RECOVERABILITY_HINT bit; this bit is copied to the guest RAX bit 60, which the guest interprets as part of a TDCALL completion status. |
| 10 | HOST_PRIORITY_LOCKS | The TDX Module implements host-priority locks to avoid denial-of-service by guest TDs. This requires the host VMM to retry operations that fail with a TDX_OPERAND_BUSY status. |
| 11 | CONFIG_IA32_ARCH_CAPABILITIES | The TDX Module allows the host VMM to configure the virtualization of IA32_ARCH_CAPABILITIES MSR. |
| 12 | SEALING | <p>The TDX Module supports signed TDs and seal keys bound to new TD properties:</p> <ul style="list-style-type: none"> New interface functions: TDG.MR.KEY.GET SEALING support depends on TD_SIGNING_AND_SVN (bit 22) support <p>The sealing key size is enumerated by SEALKEY_128 (bit 50).</p> |
| 13 | S4 | The TDX Module supports state hibernation and restoration across S4 CPU state: |

| Bit(s) | Name | Description |
|--------|--------------------------|---|
| | | <ul style="list-style-type: none"> New interface function: TDH.SYS.S4_END New input flag to TDH.EXPORT/IMPORT.STATE.IMMUTABLE |
| 14 | ACT | The TDX Module manages memory access control using the CPU's Access Control Table (ACT). |
| 15 | WBINVD_DOMAINS | TDH.PHYMEM.CACHE.WB needs to be called per WBINVD domain that might be different than a whole package. WBINVD domains are enumerated with the WBINVD_DOMAIN* metadata fields. If SKIP_PHYMEM_CACHE_WB (bit 34) is 1, then WBINVD_DOMAINS is 0. |
| 16 | PENDING_EPT_VIOLATION_V2 | The TDX Module supports enhanced handling of EPT violation on guest TD access to PENDING pages: <ul style="list-style-type: none"> Decision on whether a #VE(PENDING) is injected into the guest TD can be guest configurable. Extended exit qualification is provided to the host VMM. EPT violation on L2 VM access to a PENDING page always causes an L2→L1 exit. |
| 17 | FMS_CONFIG | The TDX Module supports configuration of virtual CPUID(1).EAX (Family/Model/Stepping) value for migratable TDs. |
| 18 | NO_RBP_MOD | The TDX Module supports a TD configuration where RBP is never modified by any host-side (SEAMCALL) and guest-side (TDCALL) interface function. This is configured by TD_PARAMS.CONFIG_FLAGS.NO_RBP_MOD (see 3.4.3). |
| 19 | L2_TLB_INVLD_OPT | The TDX Module supports additional address translation invalidation modes on TDG.VP.ENTER. |
| 20 | TOPOLOGY_ENUM | The TDX Module supports virtual topology enumeration and configuration of CPUID(0xB), CPUID(0x1F) and x2APIC ID. |
| 21 | PARTITIONED_TD_MIGRATION | The TDX Module supports migration of partitioned TDs. |
| 22 | TD_SIGNING_AND_SVN | The TDX Module supports the following: <ul style="list-style-type: none"> REPORTTYPE.VERSION value of 2 TDSIGSTRUCT Additional attestation configuration (MRSIGNER, PRODID, etc.) TDG.MR.ASSIGNSVNS interface function |
| 23 | CLFLUSH_BEFORE_ALLOC | When allocating a memory page to be used as TD private memory or TD control structure page, the host VMM is required to ensure that none of the cache lines associated with the page is in a MODIFIED state. |
| 24 | EVENT_FILTERING | The TDX Module supports filtering of performance monitoring events, based on configuration by the host VMM as part of TDH.MNG.INIT. |
| 25 | ICSSD | Instruction-Count based Single-Step Defense: Indicates that the TDX Module supports single-step attack detection based on counting TD VCPU instructions. This feature is only available for guest TDs where performance monitoring is not enabled (ATTRIBUTES.PERFMON == 0). |

| Bit(s) | Name | Description |
|--------|------------------------------|--|
| 26 | FIXED_CTR12_PROF | System profiling by IA32_FIXED_CTR1 and IA32_FIXED_CTR2 is supported. IA32_FIXED_CTR1 and IA32_FIXED_CTR2 continue counting while the TDX Module is running. If a TD is not enabled for performance monitoring (ATTRIBUTES.PERFMON == 0) and not debuggable (ATTRIBUTES.DEBUG == 0) then the counters continue counting while that TD is running. |
| 27 | MAXPA_VIRT | The TDX Module supports virtualization of physical address width, as enumerated by CPUID(0x80000008).EAX[7:0]. |
| 28 | APX | The TDX Module supports Intel® APX (Advanced Performance Extensions). Note: This bit may be set regardless of CPU support of APX. |
| 29 | CPUID2_VIRT | The TDX Module supports virtualization of CPUID(2) |
| 30 | VE_REDUCTION | The TDX Module supports run time controls by the guest TD to reduce the cases where #VE is injected by the TDX Module on guest TD execution of CPUID, RDMSR/WRMSR and other instructions. Note: VE_REDUCTION implies TOPOLOGY_ENUM and CPUID2_VIRT. |
| 31 | ENHANCED_EVENT_FILTERING | The TDX Module supports enhanced filtering of performance monitoring events, based on configuration by the host VMM as part of TDH.MNG.INIT. |
| 32 | TDX_CONNECT_PARTITIONING | The TDX Module supports TDX Connect for partitioned TDs. |
| 33 | MAXGPA_VIRT | The TDX Module supports virtualization of guest physical address width, as enumerated by CPUID(0x80000008).EAX[23:16]. |
| 34 | SKIP_PHYMEM_CACHE_WB | The host VMM needs not call TDH.PHYMEM.CACHE.WB as part of the TD teardown sequence. |
| 35 | NON_BLOCKING_RESIZE | The TDX Module supports TDH.MEM.PAGE.DEMOTE and TDH.MEM.PAGE.PROMOTE without blocking and TLB tracking. |
| 36 | DYNAMIC_PAMT | The TDX Module supports dynamic PAMT mode, where PAMT entries are dynamically allocated by the host VMM. |
| 37 | FATAL_DIAGNOSTICS | The TDX Module provides diagnostic information in case of a fatal error. |
| 38 | PAGE_RELEASE | The TD may release memory pages using TDG.MEM.PAGE.RELEASE. |
| 39 | NRX | The TDX Module supports non-root extensions. |
| 40 | ENHANCED_INTR_STATE | The TDX Module supports additional interrupt state indications by TDVPS.VCPU_STATE_DETAILS and the IMM_RESUME_HINT output flag of TDH.VP.ENTER. |
| 41 | NON_BLOCKING_EXPORT | The TDX Module supports TD migration non-blocking export, based on Secure EPT entries' Dirty bit. |
| 42 | PERF_MASK | The TDX Module supports partial allocation of Perfmon counters to a TD. |
| 43 | SCAN_EXPORT_RESTORE | The TDX Module supports the EXPORT_RESTORE operation of TDH.MEM.SCAN. |
| 44 | IMPORT_PAGE_STATUS | TDH.IMPORT.MEM provides status for pages in its page list. |
| 45 | ENHANCED_INTR_VIRTUALIZATION | The TDX Module supports enhanced interrupt virtualization, including posted interrupt and IPI processing. |

| Bit(s) | Name | Description |
|--------|----------------------------------|--|
| 46 | VEINFO_INTR_STATE | The TDX Module supports returning the guest interruptibility state by TDG.VP.VEINFO.GET. |
| 47 | UPDATE_COMPATIBILITY | The TDX Module supports enhancements to verify TD-preserving update compatibility, either before or after update. |
| 48 | SERVTD_REBIND | The TDX Module supports rebinding a new Service TD to a TD. |
| 49 | UUID_REPORTING | The TDX Module supports TD_UUID as a field of TDREPORT_STRUCT. |
| 50 | SEALKEY_128 | Applicable only if SEALING (bit 12) is set to 1. A value of 0 indicates that the TDX Module uses a 256-bit sealing key. A value of 1 indicates that the TDX Module uses a 128-bit sealing key. |
| 51 | ENHANCED_DEMOTE_INTERRUPTIBILITY | Indicates that TDH.MEM.PAGE.DEMOTE is not interruptible when the TD is not partitioned (no L2 VMs). |
| 63:52 | RESERVED | Set to 0 |

3.3.3.2. Global Metadata Fields

3.3.4. Global Metadata Description Files

Most of the TD-scope metadata information is provided in separate files:

- **global_metadata.pdf** provides a human-readable TD-scope metadata table.
- **global_metadata.csv** provides the same table in a Comma-Separated Values (CSV) format. This file can be imported, if required, to tools such as MS Excel for further processing.

The tables below describe the information provided by those files.

In addition to the above files, **global_metadata.json**, a JSON format file with the same information is provided.

Table 3.9: Global Metadata Table Description

| Column | Description |
|--------------------------------|--|
| TDX_FEATURES Enum. Bits | The set of TDX_FEATURES field bits, readable by TD*.SYS.RD*, which enumerate support for this field. One or more comma-separated bit numbers may be specified. If the value of any of those bits is, the field is supported. The special value “Always” means that the field is always supported. The special value “None” means that there is no enumeration for the support of this field. This is only used for fields that are accessible only for debuggable TDs. |
| Class | Field class, see below for details |
| Field | Field name |
| Description | Field description |
| Type | Field type |
| Field Size (Bytes) | Field size |
| Max Num Fields | Maximum number of fields in an array. The description text specified how the actual number of fields in enumerated. A value of “None” indicates that no maximum number is specified. |
| Num Elem. | Number of elements within each field |
| Elem. Size (Bytes) | Size of each element |
| Base FIELD_ID (Hex) | Base FIELD_ID |
| VMM Access | Host VMM access |

| Column | Description |
|--------------|-----------------|
| Guest Access | Guest TD access |

3.3.5. CMR_INFO

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

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

Table 3.10: 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. |

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

3.3.6. 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.

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

TDSYSINFO_STRUCT's size is 1024B.

Table 3.11: 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 |

| Section | Field Name | Offset (Bytes) | Type | Size (Bytes) | Description |
|----------------------------|------------------------------|----------------|---------|--------------|--|
| | MINOR_VERSION | 14 | Integer | 2 | Minor version number of the Intel TDX Module |
| | MAJOR_VERSION | 16 | Integer | 2 | Major version number of the Intel TDX Module |
| | SYS_RD | 18 | Boolean | 1 | A non-0 value indicates that the information in this structure is incomplete. TDH.SYS.RD or TDH.SYS.RDALL should be used to obtain TDX Module information. |
| | RESERVED | 19 | N/A | 13 | This field is reserved for enumerating future Intel TDX Module capabilities. Set to 0 |
| Memory Info | MAX_TDMRS | 32 | Integer | 2 | The maximum number of TDMRs supported |
| | MAX_RESERVED_PER_TDMR | 34 | Integer | 2 | The maximum number of reserved areas per TDMR |
| | PAMT_ENTRY_SIZE | 36 | Integer | 2 | The size of a PAMT entry – determines the number of bytes that need to be reserved for the three PAMT areas: <ul style="list-style-type: none"> • 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 |

| Section | Field Name | Offset (Bytes) | Type | Size (Bytes) | Description |
|-----------------|--------------------|----------------|--------------|--------------|---|
| TD Capabilities | ATTRIBUTES_FIXED0 | 64 | Bitmap | 8 | If any certain bit is 0 in ATTRIBUTES_FIXED0, it must be 0 in any TD's ATTRIBUTES. The value of this field reflects the Intel TDX Module capabilities and configuration and CPU capabilities. |
| | ATTRIBUTES_FIXED1 | 72 | Bitmap | 8 | If any certain bit is 1 in ATTRIBUTES_FIXED1, it must be 1 in any TD's ATTRIBUTES. The value of this field reflects the Intel TDX Module capabilities and configuration and CPU capabilities. |
| | XFAM_FIXED0 | 80 | Bitmap | 8 | If any certain bit is 0 in XFAM_FIXED0, it must be 0 in any TD's XFAM. |
| | XFAM_FIXED1 | 88 | Bitmap | 8 | If any certain bit is 1 in XFAM_FIXED1, it must be 1 in any TD's XFAM. |
| | RESERVED | 96 | N/A | 32 | Set to 0 |
| | NUM_CPUID_CONFIG | 128 | Integer | 4 | Number of the following CPUID_CONFIG entries |
| | CPUID_CONFIG[0] | 132 | CPUID_CONFIG | 24 | Enumeration of the CPUID leaves/sub-leaves that contain bit fields whose virtualization by the Intel TDX Module is either: <ul style="list-style-type: none"> • 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. |
| | CPUID_CONFIG[last] | | CPUID_CONFIG | 24 | See 3.3.1 for details. Note that the virtualization of many CPUID bit fields not enumerated in this list is configurable indirectly via the XFAM and ATTRIBUTES assigned to a TD by the host VMM. |
| Reserved | RESERVED | | N/A | | Fills up to the structure size (1024B) – set to 0 |

3.3.7. TDMR_INFO

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

Table 3.12: 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 (in bytes) of the PAMT_1G range associated with the above TDMR: since a PAMT range is aligned on 4KB, bits 11:0 are always 0. Enumeration: The size of each PAMT_1G entry (in bytes) is enumerated by PAMT_1G_ENTRY_SIZE, readable by TDH.SYS.RD*. The PAMT_1G range must be large enough to support the TDMR_SIZE specified above. |
| 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 (in bytes) of the PAMT_2M range associated with the above TDMR: since a PAMT range is aligned on 4KB, bits 11:0 are always 0. Enumeration: The size of each PAMT_2M entry (in bytes) is enumerated by PAMT_2M_ENTRY_SIZE, readable by TDH.SYS.RD*. The PAMT_2M range must be large enough to support the TDMR_SIZE specified above. |
| PAMT_4K_BASE / PAMT_PAGE_BITMAP_BASE | 48 | Physical Address | 8 | Base address of the PAMT_4K range associated with the above TDMR (HKID bits must be 0): since this range is aligned on 4KB, bits 11:0 are always 0. If the TDX Module is configured for static PAMT (default), the PAMT_4K range contains the PAMT_4K table associated with the above TDMR. Else (the TDX Module is configured for dynamic PAMT), the PAMT_4K range contains the PAMT_PAGE_BITMAP associated with the above TDMR. |
| PAMT_4K_SIZE / PAMT_PAGE_BITMAP_SIZE | 56 | Integer | 8 | Size (in bytes) of the PAMT_4K range associated with the above TDMR: since a PAMT range is aligned on 4KB, bits 11:0 are always 0. Enumeration: If the TDX Module is configured for static PAMT (default), the size of each PAMT_4K entry (in bytes) is enumerated by PAMT_4K_ENTRY_SIZE, readable by TDH.SYS.RD*. |

| Name | Offset (Bytes) | Type | Size (Bytes) | Description |
|----------------------|----------------|---------|--------------|---|
| | | | | Else (the TDX Module is configure for dynamic PAMT), the PAMT_4K range contains PAMT_PAGE_BITMAP. The size of each entry (in bits) is enumerated by PAMT_PAGE_BITMAP_ENTRY_BITS. In both cases, the PAMT_4K range must be large enough to support the TDMR_SIZE specified above. |
| RESERVED_OFFSET[0] | 64 | Integer | 8 | Offset of reserved range 0 within the TDMR. A reserved range is aligned on 4KB, thus bits 11:0 must be 0. |
| RESERVED_SIZE[0] | 72 | Integer | 8 | Size of reserved range 0 within the TDMR: <ul style="list-style-type: none"> A size of 0 indicates a null entry. All the following reserved range entries must also be null. A reserved range is aligned on 4KB, thus bits 11:0 must be 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 3.3.63.3.3.
- For backward compatibility, this value is also enumerated by TDSYSINFO_STRUCT.MAX_RESREVED_PER_TDMR (see 3.3.6).
- Within each TDMR entry, all reserved areas must be sorted from the lowest offset to the highest offset, and they must not overlap with each other.
- All TDMRs and PAMTs must be contained within CMRs.
- A PAMT area must not overlap with another PAMT area (associated with any TDMR), and it must not overlap with non-reserved areas of any TDMR. PAMT areas may reside within reserved areas of TDMRs.

3.3.8. FATAL_INFO: TDX Module Fatal Error Logging Format

Enumeration: FATAL_INFO is applicable if TDX_FEATURES0.FATAL_DIAGNOSTICS (bit 37), readable by the host VMM using TDH.SYS.RD*, is 1.

3.3.8.1. Overview

FATAL_INFO is the information logged by the TDX Module if an internal sanity check detects an internal fatal error. It is a 64-byte data structure, residing in a memory location specified by the host VMM on TDH.SYS.INIT.

Depending on the fatal error case, FATAL_INFO may provide different information. The overall format of FATAL_INFO is shown below:

Table 3.13: FATAL_INFO Overall Format

| Offset (Bytes) | Size (Bytes) | Name | Description |
|----------------|--------------|------------|--|
| 0 | 16 | BASIC_INFO | Basic diagnostic information, same format for all error cases. |

| Offset (Bytes) | Size (Bytes) | Name | Description |
|----------------|--------------|---------------|---|
| 16 | 48 | EXTENDED_INFO | Extended diagnostic information. Several formats are supported, as defined below. |

3.3.8.2. FATAL_INFO Basic Format

Table 3.14: FATAL_INFO Basic Format

| Offset (Bytes) | Size (Bytes) | Name | Description | |
|----------------|--------------|---------------|---|---|
| 0 | 1 | STATE | State of FATAL_INFO | |
| | | | Value | Meaning |
| | | | 0x00 | FATAL_INFO does not contain valid information |
| | | | 0x01 | Information is being written by the TDX Module |
| | | | 0xFE | FATAL_INFO contains valid P-SEAMLDR fatal error information |
| | | | 0xFF | FATAL_INFO contains valid TDX Module fatal error information |
| Other | Reserved | | | |
| 1 | 1 | FORMAT | Format of FATAL_INFO | |
| | | | Value | Meaning |
| | | | 0 | Basic information only |
| | | | 1 | Information for TD-specific errors: basic information + TD handle |
| | | | 2 | Information for Secure EPT related errors (with TD handle) |
| | | | 3 | Information for Secure EPT related errors (with EPTP) |
| | | | 4 | Information for page HPA related errors |
| | | | 5 | Information for PAMT related errors |
| | | | 6 | Information for an unexpected exception |
| | | | 7 | Information for an unexpected VM exit |
| Other | Reserved | | | |
| 2 | 2 | RESERVED | Reserved, set to 0 | |
| 4 | 4 | X2APIC_ID | X2APIC ID of the logical processor where the fatal error occurred | |
| 8 | 8 | ERROR_ID | Identifies the place in the TDX Module code where the fatal error was identified. The meaning of ERROR_ID is not architectural. This identification is specific to the failing TDX Module release; it may not apply to other releases. | |
| 16 | 48 | EXTENDED_INFO | Extended diagnostic information, per FORMAT. See below. | |

5 3.3.8.3. FATAL_INFO Format with TD Handle

This format is used when FATAL_INFO identifies a specific TD.

Table 3.15: FATAL_INFO with TD Handle

| Offset (Bytes) | Size (Bytes) | Name | Description | |
|----------------|--------------|------------|------------------------------|---|
| 0 | 16 | BASIC_INFO | Basic diagnostic information | |
| 16 | 8 | TD_HANDLE | Identifies the TD | |
| | | | Bits | Meaning |
| | | | 11:0 | Reserved, set to 0 |
| | | | 51:12 | Bits 51:12 of the TDR HPA (HKID bits are 0) |
| | | 63:52 | Reserved, set to 0 | |
| 24 | 40 | RESERVED | Reserved, set to 0 | |

3.3.8.4. FATAL_INFO Format for Secure EPT Related Failures (with TD Handle)

This format is used when FATAL_INFO identifies an error related to Secure EPT, providing a TD handle.

Table 3.16: FATAL_INFO for Secure EPT Related Failures (with TD Handle)

| Offset (Bytes) | Size (Bytes) | Name | Description | |
|----------------|--------------|---------------|--|--|
| 0 | 16 | BASIC_INFO | Basic diagnostic information | |
| 16 | 8 | TD_AND_VM | Identifies the TD and specific VM (L1 or L2) | |
| | | | Bits | Meaning |
| | | | 11:0 | Reserved, set to 0 |
| | | | 51:12 | Bits 51:12 of the TDR HPA (HKID bits are 0) |
| | | | 53:52 | VM (L1 or L2) index |
| | | 63:54 | Reserved, set to 0 | |
| 24 | 8 | GPA_AND_LEVEL | Identifies the guest physical address and the SEPT level | |
| | | | Bits | Meaning |
| | | | 2:0 | Offending Secure EPT entry level |
| | | | 11:3 | Reserved, set to 0 |
| | | | 51:12 | Bits 51:12 of the offending guest physical address |
| | | 63:52 | Reserved, set to 0 | |
| 32 | 8 | SEPT_ENTRY | Secure EPT entry | |
| 40 | 24 | RESERVED | Reserved, set to 0 | |

3.3.8.5. FATAL_INFO Format for Secure EPT Related Failures (with EPTP)

This format is used when FATAL_INFO identifies an error related to Secure EPT, providing EPTP.

Table 3.17: FATAL_INFO for Secure EPT Related Failures (with EPTP)

| Offset (Bytes) | Size (Bytes) | Name | Description |
|----------------|--------------|------------|------------------------------|
| 0 | 16 | BASIC_INFO | Basic diagnostic information |
| 16 | 8 | EPTP | EPT Pointer |

| Offset (Bytes) | Size (Bytes) | Name | Description | |
|----------------|--------------------|---------------|--|--|
| 24 | 8 | GPA_AND_LEVEL | Identifies the guest physical address and the SEPT level | |
| | | | Bits | Meaning |
| | | | 2:0 | Offending Secure EPT entry level |
| | | | 11:3 | Reserved, set to 0 |
| | | | 51:12 | Bits 51:12 of the offending guest physical address |
| 63:52 | Reserved, set to 0 | | | |
| 32 | 8 | SEPT_ENTRY | Secure EPT entry | |
| 40 | 24 | RESERVED | Reserved, set to 0 | |

3.3.8.6. FATAL_INFO Format for HPA Related Failures

This format is used when FATAL_INFO identifies an error related to page HPA.

Table 3.18: FATAL_INFO for Page HPA Related Failures

| Offset (Bytes) | Size (Bytes) | Name | Description |
|----------------|--------------|------------|---|
| 0 | 16 | BASIC_INFO | Basic diagnostic information |
| 16 | 8 | HPA | HPA that was used when a fatal error was detected NULL_PA (-1) if not applicable |
| 24 | 1 | SIZE | Page size (0: 4KB, 1: 2MB, 2: 1GB) |
| 25 | 39 | RESERVED | Reserved, set to 0 |

5

3.3.8.7. FATAL_INFO Format for PAMT Related Failures

This format is used when FATAL_INFO identifies an error related to PAMT.

Table 3.19: FATAL_INFO for PAMT Related Failures

| Offset (Bytes) | Size (Bytes) | Name | Description |
|----------------|--------------|------------|--|
| 0 | 16 | BASIC_INFO | Basic diagnostic information |
| 16 | 16 | PAMT_ENTRY | PAMT entry where a fatal error was detected 0 if not applicable |
| 32 | 8 | HPA | HPA that was used when a fatal error was detected NULL_PA (-1) if not applicable |
| 40 | 1 | LEVEL | Level of PAMT where a fatal error was detected |
| 41 | 7 | RESERVED | Reserved, set to 0 |
| 48 | 8 | PAMT_PAGE | HPA or linear address (depending on the error case) of the PAMT page, if applicable. |
| 56 | 8 | RESERVED | Reserved, set to 0 |

10

3.3.8.8. FATAL_INFO Format for Unexpected Exceptions

This format is used when FATAL_INFO identifies an unexpected exception.

Table 3.20: FATAL_INFO for Unexpected Exceptions

| Offset (Bytes) | Size (Bytes) | Name | Description |
|----------------|--------------|------------|------------------------------|
| 0 | 16 | BASIC_INFO | Basic diagnostic information |
| 16 | 1 | VECTOR | Exception vector |
| 17 | 47 | RESERVED | Reserved, set to 0 |

3.3.8.9. FATAL_INFO Format for an Unexpected VM Exit Related Failures

This format is used when FATAL_INFO identifies an error related to an unexpected VM exit.

Table 3.21: FATAL_INFO for Unexpected VM Exit Related Failures

| Offset (Bytes) | Size (Bytes) | Name | Description | |
|----------------|--------------------|---------------|--|---|
| 0 | 16 | BASIC_INFO | Basic diagnostic information | |
| 16 | 8 | TD_AND_VM | Identifies the TD and specific VM (L1 or L2) | |
| | | | Bits | Meaning |
| | | | 11:0 | Reserved, set to 0 |
| | | | 51:12 | Bits 51:12 of the TDR HPA (HKID bits are 0) |
| | | | 53:52 | VM (L1 or L2) index |
| 63:54 | Reserved, set to 0 | | | |
| 24 | 4 | EXIT_REASON | The Exit Reason VMCS field | |
| 28 | 4 | EXTENDED_INFO | <ul style="list-style-type: none"> For an unexpected VM exit due to IDT vectoring information, this field will provide the VMCS IDT vectoring information field. For an unexpected VM exit due to RDMSR or WRMSR, this field will provide the offending MSR index. For an unexpected VM exit interruption information, this field will provide the VMCS interruption information field. In other cases, this field will be set to 0. | |
| 32 | 32 | RESERVED | Reserved, set to 0 | |

3.4. TD Parameter Types

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

3.4.1. ATTRIBUTES

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

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

- If any bit in the TUD group is set to 1, the guest TD is under off-TD debug and is untrusted.
- Bits in the TUP group indicate features that impact security and trust. It is up to the remote verifier to decide whether the impact on TD trustworthiness is acceptable.

- Bits in the SEC group indicate features that may impact TD security but are not considered as impacting TD trust. Bits in the SEC group may have a positive or a negative impact on the TD security if set, as specified in the table.
- Bits in the OTHER group indicate features that are attested but do not impact TD security.

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

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

Notes

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

Table 3.22: ATTRIBUTES Definition

| Bits | Group | Description | Bits | Bit Name | TD Security Impact if 1 | Description |
|-------|-------|---|------|----------------------|-------------------------|--|
| 3:0 | TUD | TD Under Debug If any of the bits in this group are set to 1, the guest TD is untrusted. | 0 | DEBUG | Negative | Guest TD runs in off-TD debug mode. Its VCPU state and private memory are accessible by the host VMM. DEBUG may not be set if MIGRATABLE is set. |
| | | | 3:1 | RESERVED | Negative | Reserved for future TUD flags – must be 0 |
| 15:4 | TUP | TD Under Profiling The TD is subject to profiling, which may expose side channel information to untrusted entities. | 4 | HGS_PLUS_PROF | Negative | The TD is subject to HGS+ operation. HGS+ monitors the TD operation as part of the whole system. This bit may be set, if supported by the TDX Module, regardless of CPU support. |
| | | | 5 | PERF_PROF | Negative | The TD is subject to system profiling using performance monitoring counters. Those counters are not context-switched on TD entry and exit; they monitor the TD operation as part of the whole system. This bit may be set, if supported by the TDX Module, regardless of CPU support. |
| | | | 6 | PMT_PROF | Negative | The TD is subject to system profiling using core out-of-band telemetry. Core telemetry monitors the TD operation as part of the whole system. This bit may be set, if supported by the TDX Module, regardless of CPU support. |
| | | | 15:7 | RESERVED | Negative | Reserved for future TUP flags – must be 0 |
| 31:16 | SEC | Security | 16 | ICSSD | Positive | Indicates that the TDX Module must use Instruction-Count based Single-Step Defense to protect against single-step attacks. |

Intel TDX Application Binary Interface (ABI) Reference

| Bits | Group | Description | Bits | Bit Name | TD Security Impact if 1 | Description |
|-------|-----------------|--|-------|------------------------|-------------------------|--|
| | | Attributes that may impact TD security | | | | ICSSD may not be set if PERFMON is set. This bit may only be set if the TDX Module supports ICSSD. |
| | | | 17 | SERVTD_EXT | Positive | Indicates that TDREPORT_STRUCT includes a hash of SERVTD_EXT_STRUCT instead of SERVTD_HASH. |
| | | | 22:18 | RESERVED_P | Positive | Reserved for future SEC flags that will indicate a positive impact on TD security. <ul style="list-style-type: none">As an input to TDH.MN.INIT, must be 0. Attestation verifiers may allow any value. |
| | | | 26:23 | RESERVED_N | Negative | Reserved for future SEC flags that will indicate negative impact on TD security – must be 0 |
| | | | 27 | LASS | Positive | TD is allowed to use Linear Address Space Separation. This bit may only be set if both the TDX Module and the CPU support LASS. |
| | | | 28 | SEPT_VE_DISABLE | Negative | Disable EPT violation conversion to #VE(PENDING) on guest TD access of PENDING pages |
| | | | 29 | MIGRATABLE | Negative | TD is migratable (using a Migration TD). MIGRATABLE may not be set if either DEBUG or PERFMON is set. MIGRATABLE may not be set if CONFIG_FLAGS.TDX_CONNECT is set. This bit may only be set if the TDX Module supports TD Migration. |
| | | | 30 | PKS | Positive | TD is allowed to use Supervisor Protection Keys. This bit may only be set if both the TDX Module and the CPU support PKS. |
| | | | 31 | KL | Positive | TD is allowed to use Key Locker. This bit may only be set if both the TDX Module and the CPU support Key Locker. |
| 55:32 | RESERVED | Reserved | 55:32 | RESERVED | None | Reserved for future expansion of the SEC group - must be 0 |
| 63:56 | OTHER | Attributes that are attested but do not impact TD security | 61:56 | RESERVED | None | Reserved for future OTHER flags – must be 0 |
| | | | 62 | TPA | None | The TD is a TDX Connect Provisioning Agent. This bit may only be set if both the TDX Module and the CPU support TDX Connect. |
| | | | 63 | PERFMON | None | TD is allowed to use Perfmon and PERF_METRICS capabilities. PERFMON may not be set if either MIGRATABLE or ICSSD is set. |

| Bits | Group | Description | Bits | Bit Name | TD Security Impact if 1 | Description |
|------|-------|-------------|------|----------|-------------------------|--|
| | | | | | | This bit may only be set if the TDX Module supports Performance Monitoring virtualization. |

3.4.2. XFAM

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

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

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

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

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

3.4.3. CONFIG_FLAGS

- 15 CONFIG_FLAGS is a set of TD configuration flags.

Table 3.23: TD_PARAMS_STRUCT.CONFIG_FLAGS Definition

| Bits | Name | Description |
|------|----------------------------|--|
| 0 | GPAW | <p>GPAW (Guest Physical Address Width¹) controls the position of the SHARED bit in GPA. It is copied to each TD VMCS and L2 VMCS GPAW execution control on TDH.VP.INIT and TDH.IMPORT.STATE.VP.</p> <p>0: GPA.SHARED bit is GPA[47] 1: GPA.SHARED bit is GPA[51]</p> <p>A value of 1 can only be specified if EPTP_CONTROLS[5:3] is specified as 4 (i.e., 5-level EPT). For details, see the [TDX Arch Extensions Spec].</p> |
| 1 | FLEXIBLE_PENDING_VE | <p>Controls the guest TD's ability to change the PENDING page access behavior from its default value:</p> <p>0: The guest TD cannot change the behavior set by ATTRIBUTES.SEPT_VE_DISABLE. 1: The guest TD can change the default behavior set by ATTRIBUTES.SEPT_VE_DISABLE.</p> <p>Enumeration: Availability of FLEXIBLE_PENDING_VE is enumerated by TDX_FEATURES0.PENDING_EPT_VIOLATION_V2 (bit 16) and by CONFIG_FLAGS_FIXED0/1, readable using TDH.SYS.RD*.</p> |

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

| Bits | Name | Description |
|------|--------------------|---|
| 2 | NO_RBP_MOD | <p>Controls whether RBP value can be modified by TDG.VP.VMCALL and TDH.VP.ENTER:</p> <p>0: RBP can be used as an input to TDG.VP.VMCALL. The value provided by the guest TD is used as an output of TDH.VP.ENTER. The value provided by the host TD to the following TDH.VP.ENTER is used as an output of TDG.VP.VMCALL.</p> <p>1: RBP can't be used as an input to TDG.VP.VMCALL. TDG.VP.VMCALL preserves the guest TD's value of RBP. TDH.VP.ENTER preserves the host VMM's value of RBP.</p> <p>Enumeration: Availability of NO_RBP_MOD is enumerated by TDX_FEATURES0.NO_RBP_MOD (bit 18) and by CONFIG_FLAGS_FIXED0/1, readable using TDH.SYS.RD*.</p> |
| 3 | MAXPA_VIRT | <p>Controls virtualization of physical address width, as enumerated by CPUID(0x80000008).EAX[7:0]:</p> <p>0: The virtual value of CPUID(0x80000008).EAX[7:0] is set to the native value of that field.</p> <p>The virtual value of CPUID(0x80000008).EAX[23:16] is determined by the setting of MAXGPA_VIRT (bit 4 below).</p> <p>1: MAXGPA_VIRT (bit 4 below) must be set to 0.</p> <p>The virtual value of CPUID(0x80000008).EAX[7:0] is configured by the host VMM.</p> <p>The virtual value of CPUID(0x80000008).EAX[23:16] is set to 0.</p> <p>For details, see the [Base Spec] discussion of GPA space size virtualization.</p> <p>Enumeration: Availability of MAXPA_VIRT is enumerated by TDX_FEATURES0.MAXPA_VIRT (bit 27) and by CONFIG_FLAGS_FIXED0/1, readable using TDH.SYS.RD*.</p> |
| 4 | MAXGPA_VIRT | <p>Controls virtualization of guest physical address width, as enumerated by CPUID(0x80000008).EAX[23:16]:</p> <p>0: The virtual value of CPUID(0x80000008).EAX[7:0] is determined by the setting of MAXPA_VIRT (bit 3 above).</p> <p>The virtual value of CPUID(0x80000008).EAX[23:16] is set to 0.</p> <p>1: MAXPA_VIRT (bit 3 above) must be set to 0.</p> <p>The virtual value of CPUID(0x80000008).EAX[7:0] is set to the native value of that field.</p> <p>The virtual value of CPUID(0x80000008).EAX[23:16] is set depending on the value of GPAW (bit 0 above) and the native value of CPUID(0x80000008).EAX[7:0].</p> <p>For details, see the [Base Spec] discussion of GPA space size virtualization.</p> <p>Enumeration: Availability of MAXGPA_VIRT is enumerated by TDX_FEATURES0.MAXGPA_VIRT (bit 33) and by CONFIG_FLAGS_FIXED0/1, readable using TDH.SYS.RD*.</p> |

| Bits | Name | Description |
|------|---------------------|--|
| 5 | TDX_CONNECT | Enables TDX Connect for the current TD: 0: TDX Connect is disabled. 1: TDX Connect is enabled. TDX_CONNECT may not be set if ATTRIBUTES.MIGRATABLE is 1 and the TDX module has been configured for write-blocking based export. Enumeration: Availability of TDX_CONNECT is enumerated by TDX_FEATURES0.TDX_CONNECT (bit 6) and by CONFIG_FLAGS_FIXED0/1, readable using TDH.SYS.RD*. |
| 6 | PAGE_RELEASE | Enables TDG.MEM.PAGE.RELEASE for the current TD. If TDX_CONNECT above is set to 1, PAGE_RELEASE is implicitly set to 1. Enumeration: Availability of PAGE_RELEASE is enumerated by TDX_FEATURES0.PAGE_RELEASE (bit 38) and by CONFIG_FLAGS_FIXED0/1, readable using TDH.SYS.RD*. |
| 63:7 | RESERVED | Must be 0 |

3.4.4. CPUID_VALUES

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

Table 3.24: CPUID_VALUES Definition

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

3.4.5. TD_PARAMS

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

TD_PARAMS' size is 1024B.

Table 3.25: TD_PARAMS Definition

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

| Field | Offset (Bytes) | Type | Size (Bytes) | Description | Included in TDREPORT? | |
|------------------------|----------------|---------------------------|--------------|--|-----------------------|--|
| XFAM | 8 | 64b bitmap in XCRO format | 8 | Extended Features Available Mask: indicates the extended state features allowed for the TD. XFAM's format is the same as XCRO and IA32_XSS MSR. The value set in this field must satisfy the following conditions: <ul style="list-style-type: none"> Natively valid value for XCRO and IA32_XSS (does not contain reserved bits, features not supported by the CPU, or invalid bit combinations) Complies with XFAM_FIXED0 and XFAM_FIXED1 as enumerated by TDH.SYS.RD/RDALL or TDH.SYS.INFO. | Yes | |
| MAX_VCPUS | 16 | Unsigned 16b Integer | 2 | Maximum number of VCPUs Must be higher than 0. Must not be higher than MAX_VCPUS_PER_TD, which may be read by TDH.SYS.RD*. | No | |
| NUM_L2_VMS | 18 | Unsigned 8b Integer | 1 | Number of L2 VMs May be between 0 and 3. A value of 0 indicates no TD Partitioning is supported. | No | |
| MSR_CONFIG_CTLs | 19 | 8b bitmap | 1 | MSR configuration controls: | No | |
| | | | | Bit | | Description |
| | | | | 0 | | Indicates that TD configuration should use the IA32_ARCH_CAPABILITIES_CONFIG field below |
| | | | | Other | | Reserved, must be 0 |
| RESERVED | 20 | N/A | 4 | Must be 0 | No | |
| EPTP_CONTROLS | 24 | EPTP | 8 | Control bits of EPTP – copied to each TD VMCS on TDH.VP.INIT: Bits 2:0 Memory type – must be 110 (WB) Bits 5:3 EPT level – 1 less than the EPT page-walk length. Must be either 3 or 4. Must comply with the EPT page-walk length supported by the CPU. Bits 63:6 Reserved – must be 0 | No | |
| CONFIG_FLAGS | 32 | 64b bitmap | 8 | Non-measured TD-scope execution controls. See 3.4.3 above for details. | No | |
| TSC_FREQUENCY | 40 | 16b unsigned integer | 2 | TD-scope virtual TSC frequency in units of 25MHz – must be between 4 and 400. | No | |
| RESERVED | 42 | N/A | 38 | Must be 0 | No | |
| MRCONFIGID | 80 | SHA384_HASH | 48 | Software-defined ID for non-owner-defined configuration of the guest TD – e.g., run-time or OS configuration | Yes | |

| Field | Offset (Bytes) | Type | Size (Bytes) | Description | Included in TDREPORT? |
|--------------------------------------|----------------|----------------------|--------------|--|-----------------------|
| MROWNER | 128 | SHA384_HASH | 48 | Software-defined ID for the guest TD's owner | Yes |
| MROWNERCONFIG | 176 | SHA384_HASH | 48 | Software-defined ID for owner-defined configuration of the guest TD – e.g., specific to the workload rather than the run-time or OS | Yes |
| IA32_ARCH_CAPABILITIES_CONFIG | 224 | 64b bitmap | 8 | Configuration of IA32_ARCH_CAPABILITIES MSR virtualization (if enabled by MSR_CONFIG_CTLs above). Configuration capabilities are enumerated by the IA32_ARCH_CAPABILITIES_CONFIG_MASK, which can be read by TDH.SYS.RD/RDALL. | No |
| MRCONFIGSVN | 232 | 16b unsigned integer | 2 | SVN corresponding to MRCONFIGID Support of this field is enumerated by TDX_FEATURES0.SEALING (bit 12). If not supported, this field must be 0. | Yes |
| MROWNERCONFIGSVN | 234 | 16b unsigned integer | 2 | SVN corresponding to MROWNERCONFIG Support for this field is enumerated by TDX_FEATURES0.SEALING (bit 12). If not supported, this field must be 0. | Yes |
| RESERVED | 236 | N/A | 20 | Must be 0 | No |
| CPUID_CONFIG[0] | 256 | CPUID_VALUES | 16 | Direct configuration of CPUID leaves/sub-leaves virtualization: the number and order of entries must be equal to the number and order of directly configurable or allowable CPUID leaves/sub-leaves reported by TDH.SYS.RD/RDALL or TDH.SYS.INFO. Note that the leaf and sub-leaf numbers are implicit. Only bits that have been reported as 1 by TDH.SYS.RD/RDALL or TDH.SYS.INFO may be set to 1. | No |
| CPUID_CONFIG[n-1] | | CPUID_VALUES | 16 | | |
| RESERVED | | N/A | | Fills up to TD_PARAMS size (1024B) – must be 0 | No |

3.4.6. EVENT_FILTER and the EVENT_FILTERS Array

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

5 EVENT_FILTER Entry

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

Table 3.26: EVENT_FILTER Entry

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

EVENT_FILTERS Array

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

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

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

3.5. Physical Memory Management Types

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

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

3.5.1. PAMT Page Type (PT) Values

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

15 Table 3.27: PAMT Page Type Values

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

| Page Type | Value | TDX_FEATURES Enumeration | |
|--------------|-------|------------------------------|--|
| PT_TR | 7 | NON_BLOCKING_RESIZE (bit 35) | The physical page that has no current GPA mapping but must be TLB tracked before it can be assigned for any usage. |
| PT_EPT | 8 | N/A | The physical page holds a Secure EPT page. |
| PT_DEVIFCS_R | 9 | TDX_CONNECT (bit 6) | The physical page holds a DEVIFCS structure |
| PT_DEVIF_NR | 10 | TDX_CONNECT (bit 6) | The physical page holds a DEVIFCS internal data |
| PT_IOMMU_MT | 11 | TDX_CONNECT (bit 6) | The physical page holds an I/O (IOMMU/IDE/SPDM) data |
| PT_MMIO_MT | 12 | TDX_CONNECT (bit 6) | The physical page holds an MMIO metadata table |
| PT_DEVIFMT | 13 | TDX_CONNECT (bit 6) | The physical page holds a DEVIF metadata table |
| RESERVED | 14 | N/A | Reserved |
| PT_PAMT | 15 | DYNAMIC_PAMT (bit 36) | Indicates a PAMT non-leaf entry |
| PT_EXT_POOL | 16 | NRX (bit 39) | The physical page belongs to the TDX Module’s extended memory pool |
| PT_NRX_TDR | 17 | NRX (bit 39) | The physical page holds the TD Root (TDR) control structure of an NRX TD. |
| PT_NRX_TDVPR | 18 | NRX (bit 39) | The physical page holds a TD VCPU Root (TDVPR) page of an NRX TD. |
| RESERVED | Other | N/A | Reserved |

3.5.2. Physical Page Size

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

Table 3.28: Page Size Definition

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

5

3.5.3. HPA_LIST and HPA_LIST_INFO

The HPA_LIST is a 4KB page which contains a list of HPAs. It is used, e.g., as an input to TDH.EXT.MEM.ADD. The page is 4KB aligned. It contains up to 512 entries.

Table 3.81 below shows the format of an HPA_LIST entry.

Table 3.29: HPA_LIST Entry Definition

| Bit(s) | Size | Name | Description |
|--------|------|----------|--|
| 11:0 | 12 | RESERVED | Reserved, must be 0 |
| 51:12 | 40 | HPA | Host Physical Address bits 51:12 HKID must be 0 |

10

| Bit(s) | Size | Name | Description |
|--------|------|----------|---------------------|
| 63:52 | 12 | RESERVED | Reserved, must be 0 |

HPA_LIST_INFO references an HPA_LIST page.

Table 3.30: HPA_LIST_INFO

| Bits | Name | Description |
|-------|--------------------|---|
| 2:0 | FORMAT | 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 HPA 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 HPA list |

5 **3.6. TD Private Memory Management Data Types: Secure EPT**

Intel SDM, Vol. 3, 28.2.2 EPT Translation Mechanism

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

3.6.1. Secure EPT Levels

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

Table 3.31: EPT Levels Definition

| Level | 0 | 1 | 2 | 3 | 4 | 5 (5-Level EPT Only) |
|---------------------------------|-------|-------|--------|--------|---|----------------------|
| GPA Range Size | 4KB | 2MB | 1GB | 512GB | 256TB | 16PB ² |
| Child Physical Page Size | 4KB | 2MB | 1GB | N/A | N/A | N/A |
| EPT Page Type | N/A | EPT | EPD | EPDPT | EPML4 | EPML5 |
| Parent EPT Entry Type | EPTE | EPDE | EPDPTE | EPML4E | EPML5E (5-level EPT) or VMCS.EPTP (4-level EPT) | VMCS.EPTP |
| GPA Offset Bits | 20:12 | 29:21 | 38:30 | 47:39 | 51:48 (5-level EPT only) | N/A |

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

15 Many Intel TDX Module functions return Secure EPT entry information. This information is returned in the formats detailed below, which may be different than 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.

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

3.6.2.1. Returned L1 Secure EPT Entry Content

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

Table 3.32: L1 Secure EPT Entry Content as Returned by TDX Interface Functions

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

5

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

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

10

3.6.2.2. Returned L2 Secure EPT Entry Content

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

³ Applies to the SEPT FREE state, as well as the REMOVED and EXPORTED_REMOVED states.

Table 3.33: L2 Secure EPT Entry Content as Returned by TDX Interface Functions

| L2 Secure EPT Entry Field | | | | | | Value Returned in RCX (per Class of Entry State Returned in RDX) | | |
|---------------------------|-----|------|------------|-------------------------------|---------|--|------------|-------------------|
| MSB | LSB | Size | Short Name | Full Name | Enabled | Leaf | Non-Leaf | Free ⁴ |
| 0 | 0 | 1 | R | Read | N/A | R | R | 0 |
| 1 | 1 | 1 | W | Write | N/A | W | W | 0 |
| 2 | 2 | 1 | Xs | Execute | N/A | Xs | Xs | 0 |
| 5 | 3 | 3 | MT | Memory Type | N/A | MT | 0 | 0 |
| 6 | 6 | 1 | IPAT | Ignore PAT | N/A | IPAT | 0 | 0 |
| 7 | 7 | 1 | PS | Leaf | N/A | 1 | 0 | 0 |
| 8 | 8 | 1 | A | Accessed | No | 0 | 0 | 0 |
| 9 | 9 | 1 | D | Dirty | No | 0 | 0 | 0 |
| 10 | 10 | 1 | Xu | Execute (User) | No | Xu | Xu | 0 |
| 11 | 11 | 1 | Ignored | Ignored | N/A | 0 | 0 | 0 |
| 51 | 12 | 40 | HPA[51:12] | Host Physical Address [51:12] | N/A | HPA[51:12] | HPA[51:12] | 0 |
| 57 | 57 | 1 | VGP | Verify Guest Paging | No | 0 / VGP | 0 | 0 |
| 58 | 58 | 1 | PWA | Paging-Write Access | No | 0 / PWA | 0 | 0 |
| 59 | 59 | 1 | Ignored | Ignored | N/A | 0 | 0 | 0 |
| 60 | 60 | 1 | SSS | Supervisor Shadow Stack | No | 0 / SSS | 0 | 0 |
| 61 | 61 | 1 | SPP | Check Sub-Page Permissions | No | 0 | 0 | 0 |
| 62 | 62 | 1 | Ignored | Ignored | N/A | 0 | 0 | 0 |
| 63 | 63 | 1 | SVE | Suppress #VE | Yes | SVE | 0 | 1 |

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

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

3.6.2.3. Additional Returned Secure EPT Information

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

⁴ Applies to the L2 SEPT `L2_FREE` state.

Table 3.34: Additional Secure EPT Entry Information Returned by TDX Interface Functions

| Bits | Name | Description |
|-------|----------|--|
| 2:0 | Level | Level of the returned Secure EPT entry – see 3.6.1 above |
| 7:3 | Reserved | Set to 0 |
| 15:8 | State | The TDX state of the Secure EPT entry – see Table 3.35 below |
| 17:16 | VM | Index of the VM for which the SEPT information is returned |
| 63:18 | Reserved | Set to 0 |

Table 3.35: Secure L1 EPT Entry TDX State Returned by TDX Interface Functions

| L1 SEPT Entry State Name | Public State Number | Class: Free, Non-Leaf or Leaf | Description | TDX_FEATURES0 Enumeration |
|--------------------------|---------------------|-------------------------------|--|---------------------------|
| FREE | 0 | Free | L1 Secure EPT entry does not map a GPA range. | N/A |
| REMOVED | 5 | Free | L1 Secure EPT entry is of a removed page | TD_MIGRATION |
| NL_MAPPED | 132 | Non-Leaf | L1 Secure EPT entry maps a private GPA range which is accessible by the guest TD. | N/A |
| NL_BLOCKED | 129 | Non-Leaf | L1 Secure EPT entry maps a private GPA range, but new address translations to that range are blocked. | N/A |
| MAPPED | 4 | Leaf | L1 Secure EPT entry maps a private GPA page which is accessible by the guest TD. | N/A |
| BLOCKED | 1 | Leaf | L1 Secure EPT entry maps a private GPA page but new address translations to that range are blocked. | N/A |
| REMOVE_IN_PROGRESS | 6 | Free | L1 Secure EPT entry maps a private page that is being removed (TDH.MEM.PAGE.REMOVE has been interrupted). | ACT |
| BLOCKEDW | 8 | Leaf | Write-Blocking Export: L1 Secure EPT entry maps a private GPA page, but new address translations for write operations to that range are blocked. | TD_MIGRATION |
| EXPORTED_BLOCKEDW | 9 | Leaf | Write-Blocking Export: L1 Secure EPT entry maps a private page that has been blocked for writing and exported. | TD_MIGRATION or S4 |
| EXPORTED_DIRTY | 11 | Leaf | Write-Blocking Export: L1 Secure EPT entry maps a private page that was exported, but is not blocked for writing and its content and/or attributes may have since been modified. | TD_MIGRATION |

| L1 SEPT Entry State Name | Public State Number | Class: Free, Non-Leaf or Leaf | Description | TDX_FEATURES0 Enumeration |
|-----------------------------|---------------------|-------------------------------|--|-----------------------------|
| EXPORTED_DIRTY_BLOCKEDW | 12 | Leaf | Write-Blocking Export: L1 Secure EPT entry maps a private page that was previously exported, its content and/or attributes may have since been modified and then it was blocked for writing. | TD_MIGRATION |
| EXPORTED | 24 | Leaf | Non-Blocking Export: The page has been exported. This state indicates that any change in the page content or attributes would require a re-export. | N/A |
| EXPORTED_MODIFIED | 25 | Leaf | Non-Blocking Export: The page was exported and later either identified as dirty based on the Dirty bit (which was cleared by the same operation that checked its value) or some memory management operation changed the page attributes or state. This state indicates that the page must be re-exported as a REMIGRATE operation. | NON_BLOCKING_EXPORT |
| EXPORTED_BLOCKED | 26 | Leaf | Non-Blocking Export: The page was exported and later blocked by TDH.MEM.RANEG.BLOCK. This state indicates that the page must be re-exported as a CANCEL operation. | NON_BLOCKING_EXPORT |
| EXPORTED_REMOVED | 27 | Free | Non-Blocking Export: The page was exported and later removed by TDH.MEM.PAGE.REMOVE. This state indicates that the page should be re-exported, as a CANCEL operation. | NON_BLOCKING_EXPORT |
| EXPORTED_REMOVE_IN_PROGRESS | 28 | Free | Non-Blocking Export: The page was exported and later partially removed by TDH.MEM.PAGE.REMOVE. This state indicates that the page should be re-exported, as a CANCEL operation. | NON_BLOCKING_EXPORT and ACT |
| PENDING | 2 | Leaf | L1 Secure EPT entry maps a 4KB or a 2MB page that has been dynamically added to the guest TD using TDH.MEM.PAGE.AUG and is pending acceptance by the guest TD using TDG.MEM.PAGE.ACCEPT. This page is not yet accessible by the guest TD. | N/A |
| PENDING_BLOCKED | 3 | Leaf | L1 Secure EPT entry is both pending and blocked. | N/A |
| PENDING_BLOCKEDW | 16 | Leaf | Write-Blocking Export: L1 Secure EPT entry is both pending and blocked for writing. | TD_MIGRATION |
| PENDING_EXPORTED_BLOCKEDW | 17 | Leaf | Write-Blocking Export: L1 Secure EPT entry is both pending and exported. | TD_MIGRATION or S4 |

| L1 SEPT Entry State Name | Public State Number | Class: Free, Non-Leaf or Leaf | Description | TDX_FEATURES0 Enumeration |
|---------------------------------|---------------------|-------------------------------|---|---------------------------|
| PENDING_EXPORTED_DIRTY | 19 | Leaf | Write-Blocking Export: L1 Secure EPT entry is both pending and exported, and is not blocked for writing. | TD_MIGRATION |
| PENDING_EXPORTED_DIRTY_BLOCKEDW | 20 | Leaf | L1 Secure EPT entry is both pending and exported, and is blocked for writing. | TD_MIGRATION |
| PENDING_EXPORTED | 29 | Leaf | Non-Blocking Export: The page has been exported. This state indicates that any change in the page attributes would require a re-export. | NON_BLOCKING_EXPORT |
| PENDING_EXPORTED_MODIFIED | 30 | Leaf | Non-Blocking Export: The page was exported and later identified as dirty based on the Dirty bit (which was atomically cleared by the same operation that checked its value). This state indicates that the page must be re-exported as a REMIGRATE operation. | NON_BLOCKING_EXPORT |
| PENDING_EXPORTED_BLOCKED | 31 | Leaf | Non-Blocking Export: The page was exported and later blocked by TDH.MEM.RANEG.BLOCK. This state indicates that the page must be re-exported as a CANCEL operation. | NON_BLOCKING_EXPORT |
| MMIO_MAPPED | 32 | Leaf | L1 Secure EPT entry maps a private MMIO page which is accessible by the guest TD. | TDX_CONNECT |
| MMIO_BLOCKED | 33 | Leaf | L1 Secure EPT entry maps a private MMIO page, but new address translations to that page are blocked. | TDX_CONNECT |
| MMIO_PENDING | 34 | Leaf | L1 Secure EPT entry maps a 4KB, 2MB or 1GB MMIO page that is pending acceptance by the guest TD using TDG.MMIO.ACCEPT. This page is not yet accessible by the guest TD. | TDX_CONNECT |

Table 3.36: Secure L2 EPT Entry TDX State Returned by TDX Interface Functions

| L2 SEPT Entry State Name | Public State Number | Class: Free, Leaf or Non-Leaf | Description | TDX_FEATURES Enumeration |
|--------------------------|---------------------|-------------------------------|---|--------------------------|
| L2_FREE | 64 | Free | L2 Secure EPT entry does not map a GPA range. | TD_PARTITIONING |
| L2_NL_MAPPED | 196 | Non-Leaf | L2 Secure EPT entry maps a private GPA range which is accessible by the L2 VM. | TD_PARTITIONING |
| L2_NL_BLOCKED | 193 | Non-Leaf | L2 Secure EPT entry maps a private GPA range, but new address translations to that range are blocked. | TD_PARTITIONING |
| L2_MAPPED | 68 | Leaf | L2 Secure EPT entry maps a private GPA page which is accessible by the L2 VM. | TD_PARTITIONING |

| L2 SEPT Entry State Name | Public State Number | Class: Free, Leaf or Non-Leaf | Description | TDX_FEATURES Enumeration |
|--------------------------|---------------------|-------------------------------|--|---------------------------------|
| L2_BLOCKED | 65 | Leaf | L2 Secure EPT entry maps a private GPA page but new address translations to that range are blocked. | TD_PARTITIONING |
| L2_MMIO_MAPPED | 96 | Leaf | L2 Secure EPT entry maps a private MMIO page which is accessible by the L2 VM. | TD_PARTITIONING and TDX_CONNECT |
| L2_MMIO_BLOCKED | 97 | Leaf | L2 Secure EPT entry maps a private MMIO page, but new address translations to that page are blocked. | TD_PARTITIONING and TDX_CONNECT |

3.6.3. GPA_ATTR: GPA Attributes

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

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

Table 3.37: GPA_ATTR: GPA Attributes (all VMs) Definition

| Bits | VM Index | Bits | Description |
|-------|----------|-------|---|
| 15:0 | 0 | 15:0 | GPA attributes for L1 (all-0 for migration) |
| 31:16 | 1 | 31:24 | GPA attributes for L2 VM #1 |
| 47:32 | 2 | 47:32 | GPA attributes for L2 VM #2 |
| 63:48 | 3 | 63:48 | GPA attributes for L2 VM #3 |

Table 3.38: GPA_ATTR_SINGLE_VM: GPA Attributes (Single VM) Definition

| Bit(s) | Size | Attribute Type | Name | Description | TDG.MEM.PAGE.ATTR.RD/WR Access | | | Used for Migration | |
|--------|------|----------------|----------|-------------------------|--------------------------------|---------|----------------------|--------------------|-----|
| | | | | | L1 | L2 Mem. | L2 MMIO ⁵ | L1 | L2 |
| 0 | 1 | Intel64 | R | Read | None | RW | RW | No | Yes |
| 1 | 1 | Intel64 | W | Write | None | RW | RW | No | Yes |
| 2 | 1 | Intel64 | Xs | Execute (Supervisor) | None | RW | R | No | Yes |
| 3 | 1 | Intel64 | Xu | Execute (User) | None | RW | R | No | Yes |
| 4 | 1 | Intel64 | VGP | Verify Guest Paging | None | RW | R | No | Yes |
| 5 | 1 | Intel64 | PWA | Paging-Write Access | None | RW | R | No | Yes |
| 6 | 1 | Intel64 | SSS | Supervisor Shadow Stack | None | RW | R | No | Yes |
| 7 | 1 | Intel64 | RESERVED | Reserved | None | None | None | No | No |
| 14:8 | 7 | N/A | RESERVED | Must be 0 | R | R | R | No | No |

⁵ Applicable only if the TDX Module supports TDX Connect

| Bit(s) | Size | Attribute Type | Name | Description | TDG.MEM.PAGE. ATTR.RD/WR Access | | | Used for Migration | |
|--------|------|----------------|-------|--|---------------------------------|---------|----------------------|--------------------|-----|
| | | | | | L1 | L2 Mem. | L2 MMIO ⁵ | L1 | L2 |
| 15 | 1 | TDX | VALID | Indicates that the other bits are valid. If its value is 0, other fields are reserved and must be 0. | RW | RW | RW | No | Yes |

3.6.3.1. GPA Attributes Rules

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

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

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

3.6.4. GLA List

GLA lists are used by TDG.VP.INVGLA.

3.6.4.1. GLA_LIST_ENTRY

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

Table 3.39: GLA_LIST_ENTRY Definition

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

3.6.4.2. GLA_LIST

A GLA_LIST is an array of up to 512 GLA_LIST_ENTRIES.

3.6.4.3. GLA_LIST_INFO: GLA List GPA and Additional Information

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

Table 3.40: GLA_LIST_INFO

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

3.7. TD Entry and Exit Types

3.7.1. Extended Exit Qualification

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

Table 3.41: Extended Exit Qualification

| Bits | Name | Description | | |
|-------|----------|----------------------------------|------------------------------------|---|
| 3:0 | TYPE | Extended exit qualification type | | |
| | | Value | Name | Description |
| | | 0 | NONE | No extended exit qualification |
| | | 1 | ACCEPT | Extended exit qualification for an EPT violation during TDG.MEM.PAGE.ACCEPT |
| | | 2 | GPA_DETAILS | Extended exit qualification for an EPT violation caused by guest-side interface function failure of GPA→HPA translation |
| | | 3 | TD_ENTRY_MSR_LOAD_FAILURE | Extended exit qualification for failures of TD entry due to loading guest MSR state |
| | | 4 | TD_ENTRY_XSTATE_LOAD_FAILURE | Extended exit qualification for failures of TD entry due to loading guest extended state |
| | | 5 | ATTR_WR | Extended exit qualification for an EPT violation during TDG.MEM.PAGE.ATTR.WR |
| | | 6 | PENDING_EPT_VIOLATION ⁶ | Extended exit qualification for an EPT violation due to guest TD access to a PENDING page |
| | | 7 | RELEASE ⁷ | Extended exit qualification for an EPT violation during TDG.MEM.PAGE.RELEASE |
| Other | Reserved | | | |
| 31:4 | Reserved | Set to 0 | | |
| 63:32 | INFO | TYPE-specific information | | |
| | | TYPE | Value | |
| | | NONE | 0 | |
| | | ACCEPT | See the table below | |
| | | GPA_DETAILS | See the table below | |
| | | TD_ENTRY_MSR_LOAD_FAILURE | MSR index | |
| | | TD_ENTRY_XSTATE_LOAD_FAILURE | 0 | |
| | | ATTR_WR | See the table below | |
| | | PENDING_EPT_VIOLATION | 0 | |
| | | RELEASE | See the table below | |
| | | Reserved | 0 | |

⁶ Availability of this indication is enumerated by TDX_FEATURES0.PENDING_EPT_VIOLATION_V2 (bit 16), readable by TDH.SYS.RD*.

⁷ Availability of this indication is enumerated by TDX_FEATURES0.PAGE_RELEASE (bit 38), readable by TDH.SYS.RD*.

Table 3.42: Extended Exit Qualification INFO Field (Bits 63:32) when TYPE is ACCEPT, ATTR_WR or RELEASE

| Bits | Name | Description |
|-------|------------------|--|
| 34:32 | REQ_SEPT_LEVEL | SEPT level requested as an input to TDG.MEM.PAGE.ACCEPT or TDG.MEM.PAGE.ATTR.WR or TDG.MEM.PAGE.RELEASE |
| 37:35 | ERR_SEPT_LEVEL | SEPT level in which TDG.MEM.PAGE.ACCEPT or TDG.MEM.PAGE.ATTR.WR or TDG.MEM.PAGE.RELEASE detected an error |
| 45:38 | ERR_SEPT_STATE | The TDX state of the Secure EPT entry where TDG.MEM.PAGE.ACCEPT, TDG.MEM.PAGE.ATTR.WR or TDG.MEM.PAGE.RELEASE detected an error – see Table 3.35 above |
| 46 | ERR_SEPT_IS_LEAF | Indicates that the SEPT entry where TDG.MEM.PAGE.ACCEPT or TDG.MEM.PAGE.ATTR.WR or TDG.MEM.PAGE.RELEASE detected an error is a leaf entry |
| 63:47 | Reserved | Set to 0 |

Table 3.43: Extended Exit Qualification INFO Field (Bits 63:32) when TYPE is GPA_DETAILS

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

5

3.8. L2 VM Transition Types

3.8.1. L2_ENTER_GUEST_STATE

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

10

Table 3.44: L2_ENTER_GUEST_STATE Definition

| Offset (Bytes) | Size (Bytes) | Name | Description |
|----------------|--------------|------|-------------|
| 0 | 8 | RAX | |
| 8 | 8 | RCX | |
| 16 | 8 | RDX | |
| 24 | 8 | RBX | |
| 32 | 8 | RSP | |
| 40 | 8 | RBP | |
| 48 | 8 | RSI | |
| 56 | 8 | RDI | |
| 64 | 8 | R8 | |
| 72 | 8 | R9 | |

| Offset (Bytes) | Size (Bytes) | Name | Description |
|----------------|--------------|------------------------|---------------------------------|
| 80 | 8 | R10 | |
| 88 | 8 | R11 | |
| 96 | 8 | R12 | |
| 104 | 8 | R13 | |
| 112 | 8 | R14 | |
| 120 | 8 | R15 | |
| 128 | 8 | RFLAGS | |
| 136 | 8 | RIP | |
| 144 | 8 | SSP | |
| 152 | 2 | GUEST_INTERRUPT_STATUS | Bits 7:0: RVI Bits 15:7: SVI |

3.9. Measurement and Attestation Types

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

3.9.1. CPUSVN

5 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, ACTM, Boot Guard ACM, BIOS Guard module and SEAMLDR.

3.9.2. TDREPORT_STRUCT

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

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

- For REPORTTYPE.VERSION values of 0 and 1, TDREPORT_STRUCT's size is 1024 bytes.
- 15 • For REPORTTYPE.VERSION value of 2 (if supported), TDREPORT_STRUCT's size is 1280 bytes.

TD software calling TDG.MR.REPORT should be prepared for a report of up to MAX_TDREPORT_SIZE, as enumerated by TDG.SYS.RD.

Table 3.45: 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 | 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 |

| Name | Offset (Bytes) | Type | Size (Bytes) | Description |
|---------------|----------------|---------------|--------------|--|
| TDINFO | 512 | TDINFO_STRUCT | See 3.9.7 | Structure containing the TD's attestable properties. <ul style="list-style-type: none"> The hash of this structure is found in REPORTMACSTRUCT.TEE_INFO_HASH. Size is determined by REPORTMACSTRUCT.REPORTTYPE.VERSION. See 3.9.7 for details. |

3.9.3. TEE_TCB_INFO (Reference)

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

- 5 The size of TEE_TCB_INFO is 239 bytes.

Table 3.46: TEE_TCB_INFO Definition

| Name | Offset (Bytes) | Size (Bytes) | Description |
|---------------------|----------------|--------------|---|
| VALID | 0 | 8 | Indicates which TEE_TCB_INFO fields are valid. <ul style="list-style-type: none"> 1 in the i^{th} significant bit reflects that the 8 bytes starting at offset $(8 * i)$ are valid 0 in the i^{th} significant bit reflects that either 8 bytes starting at offset $(8 * i)$ is not populated or reserved and is set to zero. Set to 0x301FF. |
| TEE_TCB_SVN | 8 | 16 | TEE_TCB_SVN of the TDX Module that created the TD on the current platform. TD Migration: For a TD which has been migrated, this is the TEE_TCB_SVN of the TDX Module on the destination platform, at the time of destination TD creation (TDH.MNG.CREATE), before import. |
| MRSEAM | 24 | 48 | The measurement of the TDX Module that created the TD on the current platform. TD Migration: For a TD which has been migrated, this is the measurement of the TDX Module on the destination platform, at the time of destination TD creation (TDH.MNG.CREATE), before import. |
| MRSIGNERSEAM | 72 | 48 | Set to all 0's. |
| ATTRIBUTES | 120 | 8 | Set to all 0's. |
| TEE_TCB_SVN2 | 128 | 16 | TEE_TCB_SVN of the current TDX Module on the current platform. TD Migration: For a TD which has been migrated, this is the measurement of the current TDX Module on the destination platform, at the time TDREPORT_STRUCT is generated by TDG.MR.REPORT. Note: TEE_TCB_SVN2 may be different than TEE_TCB_SVN, due to TD-preserving TDX Module updates. |
| RESERVED | 144 | 95 | Set to all 0's. |

3.9.4. TEE_TCB_SVN (Reference)

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

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

3.9.5. REPORTMACSTRUCT (Reference)

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

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

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

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

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

The size of REPORTMACSTRUCT is 256B.

Table 3.47: 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 | For TDX, SHA384 of TEE_TCB_INFO | Yes |
| TEE_INFO_HASH | 80 | SHA384_HASH | 48 | SHA384 of TEE_INFO: a TEE-specific info structure (TDINFO_STRUCT or SGXINFO) or 0 if no TEE is represented | Yes |
| REPORTDATA | 128 | | 64 | A set of data used for communication between the caller and the target. | Yes |
| RESERVED | 192 | | 32 | Must be zero | Yes |
| MAC | 224 | | 32 | The MAC over the REPORTMACSTRUCT with model-specific MAC | No |

20 3.9.6. REPORTTYPE (Reference)

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

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

The size of REPORTTYPE is 4B.

Table 3.48: TDX-Specific REPORTTYPE Definition

| Name | Offset (Bytes) | Size (Bytes) | Description | Value |
|-----------------|----------------|--------------|--|---|
| TYPE | 0 | 1 | Trusted Execution Environment (TEE) Type | 0x00: SGX 0x7F-0x01: Reserved (TEE implemented by CPU) 0x80: Reserved (TEE implemented by a SEAM module) 0x81: TDX 0xFF-0x82: Reserved (TEE implemented by a SEAM module) |
| SUBTYPE | 1 | 1 | TYPE-specific subtype | 0: Standard TDX report Other: Reserved |
| VERSION | 2 | 1 | TYPE-specific version. | For TDX, VERSION may have the following values: 0: There are no bound nor pre-bound service TDs. TDINFO_STRUCT.SERVTD_HASH is not used (its value is 0). 1: TDINFO_STRUCT.SERVTD_HASH is used. 2: The TD has been assigned with an MRSIGNER and SVNs. ⁸ |
| RESERVED | 3 | 1 | Must be zero | 0 |

5 3.9.7. TDINFO_STRUCT

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

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

- For REPORTTYPE.VERSION values of 0 and 1, TDREPORT_STRUCT's size is 512 bytes.
- For REPORTTYPE.VERSION value of 2, TDREPORT_STRUCT's size is 768 bytes.

15 **Table 3.49: Overall TDINFO_STRUCT Definition**

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

⁸ Version 2 is supported if MRCONFIGSVN and MROWNERCONFIGSVN are supported, or if TDG.MR.ASSIGNSVNS is supported.

| Name | Offset (Bytes) | Size (Bytes) | Description | |
|------|----------------|--------------|-------------|------------|
| | | 2 | 320 | See below. |

TDINFO_BASE

Table 3.50: TDINFO_BASE Definition

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

5 TDINFO_EXTENSION for REPORTTYPE.VERSION 2

Table 3.51: TDINFO_EXTENSION for REPORTTYPE.VERSION 2 Definition

| Name | Offset (Bytes) | Offset from Start of TDINFO_STRUCT (Bytes) | Type | Size (Bytes) | Description |
|------------------|----------------|--|-------------------------|--------------|----------------------------|
| MRSIGROOT | 0 | 448 | SHA384_HASH | 48 | Signer Roots of MRSIGNER |
| MRSIGNER | 48 | 496 | SHA384_HASH | 48 | Signer of TD's TDSIGSTRUCT |
| PRODID | 96 | 544 | 128-bit integer | 16 | Product ID |
| ISVSVN | 112 | 560 | 16-bit unsigned integer | 2 | SVNs of the TD |
| MRCONFIGSVN | 114 | 562 | | 2 | SVN of MRCONFIG |
| MROWNERCONFIGSVN | 116 | 564 | | 2 | SVN of MROWNERCONFIG |
| RESERVED | 118 | 566 | N/A | 202 | Must be zero |

3.9.8. TDSIGSTRUCT: TD Signature Structure

TDSIGSTRUCT contains information about the TD from the TDSIGSTRUCT signer. It can be evaluated at any time, allowing it to leverage RTMRs which initially are empty, and is populated during the boot process, eventually reaching the steady state value that will be assigned an SVN.

- 5 TDSIGSTRUCT.POLICY indicates which of the measurement reference values in the TD are populated and should be compared against the current measurement of the TD.

TDSIGSTRUCT includes reference measurements as SHA384 digests, as defined in [FIPS PUB 180-4]. The digests are byte strings of length 48. Each of 6 64-bit words of the SHA384 hash is stored as big-endian, i.e., with its most significant byte at the lowest memory address.

- 10 SIGSTRUCT includes two 3072-bit integers (MODULUS, SIGNATURE). Each such integer is represented in a small-endian format, i.e., a byte string of length 384, with the least significant byte at the lowest memory address.

The 3072-bit integer SIGNATURE should be an RSA signature, where:

- The RSA modulus (MODULUS) is a 3072-bit integer
- The public exponent is set to $2^{16}+1$
- 15 • The signing procedure uses the EMSA-PKCS1-v1.5 format with DER encoding of the “DigestInfo” value, as specified in of PKCS#1 v2.1/RFC 3447.

SIGSTRUCT must be 4KB aligned.

In the table below, the “Signed” column indicates that this field should be included in the signature generated by the TD author.

20 **Table 3.52: TDSIGSTRUCT: TD Signature Structure**

| Name | Offset (bytes) | Size (bytes) | Description | Signed |
|---------------|----------------|--------------|---|--------|
| HEADERTYPE | 0 | 4 | Module type (Must be 6) | Y |
| HEADERLEN | 4 | 4 | Header length including crypto fields (dwords) (Must be 0x00E1) | Y |
| HEADERVERSION | 8 | 4 | Structure Version (Must be 0x00010000) | |
| TYPE | 12 | 4 | bit 31: Available to SW bits 30-0: Must be zero | Y |
| MODVENDOR | 16 | 4 | Module Vendor: <ul style="list-style-type: none"> • Intel: 8086h • ISV: 0000h | Y |
| DATE | 20 | 4 | Build date format is yyyyymmdd in decimal: yyyy=4-digit year, mm=1-12, dd=1-31 | Y |
| SIZE | 24 | 4 | Size of the entire module (dwords) (Must be 0x0540) | Y |
| KEYSIZE | 28 | 4 | RSA public key modulus size (dwords) (Must be 0x0060) | Y |
| MODULUSSIZE | 32 | 4 | RSA public key modulus size (dwords) (Must be 0x0060) | Y |
| EXPONENTSIZE | 36 | 4 | RSA public key exponent size (dword) (Must be 0x0001) | Y |
| RESERVED | 40 | 88 | Reserved. Must be 0 | Y |
| MODULUS | 128 | 384 | Module Public Key (keylength=3072 bits) | N |
| EXPONENT | 512 | 4 | RSA Exponent = $2^{16}+1$ | N |
| SIGNATURE | 516 | 384 | Signature over Header and Body | N |

| Name | Offset (bytes) | Size (bytes) | Description | Signed |
|----------|----------------|--------------|--|--------|
| POLICY | 900 | 2 | Bits indicate which measurements should be compared against the respective reference values. Bit 0: SIGSTRUCTTYPE (MRSIGNER) 0: Leaf SIGSTRUCT 1: Chain SIGSTRUCT Bit 1: MRTD Bit 2: RTMR0 Bit 3: RTMR1 Bit 4: RTMR2 Bit 5: RTMR3 Bits 15:6: Must be Zero | Y |
| ISVSVN | 902 | 2 | SVN assigned to this TD | Y |
| PRODID | 904 | 16 | Product ID for this TD | Y |
| RESERVED | 920 | 8 | Must be 0 | Y |
| MRTDREF | 928 | 8 | Reference value for MRTD | Y |
| RTMRREF | 976 | 48*4 | Array of 4 reference values for RTMRs | Y |
| MRSIGNER | 1168 | 48 | MRSIGNER for this TD | Y |
| RESERVED | 1216 | 64 | Must be 0 | Y |

3.9.9. TDKEYREQUEST

TDKEYREQUEST is used as an input to TDG.MR.KEY.GET.

TDKEYREQUEST's size is 128 bytes.

Table 3.53: TDKEYREQUEST

| Name | Offset (Bytes) | Type | Size (Bytes) | Description |
|------------------|----------------|-------------|--------------|--|
| KEYNAME | 0 | Key Name | 2 | Identifies the key being requested 0: Seal Key Other: Reserved |
| SWKEYNAME | 2 | SW Key Name | 1 | TD SW assigned key name/identifier |
| RESERVED | 3 | RESERVED | 5 | Reserved, must be 0 |
| KEYPOLICY | 8 | TDKEYPOLICY | 8 | See below |
| ATTRIBUTES_MASK | 16 | MASK | 8 | |
| XFAM_MASK | 24 | MASK | 8 | |
| CPUSVN | 32 | CPUSVN | 16 | |
| TEE_TCB_SVN | 48 | TEE_TCB_SVN | 16 | |
| ISVSVN | 64 | SVN | 2 | |
| MRCONFIGSVN | 66 | SVN | 2 | |
| MROWNERCONFIGSVN | 68 | SVN | 2 | |
| SALT | 70 | Random Data | 32 | Random data from the user |

| Name | Offset (Bytes) | Type | Size (Bytes) | Description |
|----------|----------------|----------|--------------|---------------------|
| RESERVED | 102 | RESERVED | 26 | Reserved, must be 0 |

Table 3.54: TDKEYPOLICY

| Name | Bit(s) | Size (Bits) | Description |
|---------------|--------|-------------|---|
| MRTD | 0 | 1 | Include MRTD |
| MROWNER | 1 | 1 | Include MROWNER |
| MRCONFIGID | 2 | 1 | Include MRCONFIGID & CONFIGIDSVN |
| MROWNERCONFIG | 3 | 1 | Include MROWNERCONFIG & OWNERCONFIGSVN |
| RESERVED | 7:4 | 4 | Reserved, must be 0 |
| MRSIGROOT | 8 | 1 | Include MRSIGROOT |
| MRSIGNER | 9 | 1 | Include MRSIGNER |
| MRPRODID | 10 | 1 | Include ISVPRODID & ISVSVN |
| RESERVED | 31:11 | 21 | Reserved for up to 7 more layers, must be 0 |
| RTMR | 47:32 | 16 | 4:0: Include RTMR n 7:5: Reserved, must be 0 |
| RESERVED | 63:48 | 16 | Reserved, must be 0 |

3.10. Metadata Access Types

- 5 **Note:** This section describes control structure field access types, as defined. Implementation may differ. Metadata access is described in the [TDX Module Base Spec].

3.10.1. MD_FIELD_ID: Metadata Field Identifier / Sequence Header

MD_FIELD_ID is used for two purposes:

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

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

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

Table 3.55: MD_FIELD_ID (Metadata Field Identifier / Sequence Header) Definition

| Bits | Size | Name | From Metadata Tables' Base Field ID | Single-Element or Sequence Header | Description |
|-------|------|------------|-------------------------------------|-----------------------------------|---|
| 23:0 | 24 | FIELD_CODE | Yes | Both | For a single-element identifier, identifies the element that is being accessed. For a metadata sequence header, identifies the first field that is being accessed in a sequence. |
| 31:24 | 8 | RESERVED | No | Both | Must be 0 |

| Bits | Size | Name | From Metadata Tables' Base Field ID | Single-Element or Sequence Header | Description |
|-------|------|------------------------|-------------------------------------|-----------------------------------|--|
| 33:32 | 2 | ELEMENT_SIZE_CODE | Yes | Both | Size of a single element of a metadata field: 0: 8 bits 1: 16 bits 2: 32 bits 3: 64 bits For backward compatibility, TDH.MNG.RD, TDH.MNG.WR, TDH.VP.RD and TDH.VP.WR version 0 ignore this field and use a default value based on the field code. |
| 37:34 | 4 | LAST_ELEMENT_IN_FIELD | No | Sequence Header | Number of elements in a metadata field, minus 1 For a single-element identifier, the value is 0. This field is ignored when used as input to TD*.SYS.RDALL. |
| 46:38 | 9 | LAST_FIELD_IN_SEQUENCE | No | Sequence Header | Number of fields in a sequence, minus 1 For a single-element identifier, the value is 0. This field is ignored when used as input to TD*.SYS.RDALL. |
| 49:47 | 3 | RESERVED | Yes | Both | Must be 0 |
| 50 | 1 | INC_SIZE | Yes | Sequence Header | For a single-element identifier, INC_SIZE is ignored. For a sequence header, INC_SIZE specifies how FIELD_CODE is incremented when accessing consecutive elements in a sequence: 0: Increment FIELD_CODE by 1 for each element. 1: Increment FIELD_CODE by 2 for each element. INC_SIZE is designed to support VMCS field encoding, where bit 0 (access type) is always 0 for full access. |
| 51 | 1 | WRITE_MASK_VALID | No | Both | Indicates that a write mask is provided together with the write value. For backward compatibility, single-element metadata write interface functions (e.g., TDH.MNG.WR, TDH.VP.WR etc.) and use an implicit value of 1. This field is ignored by metadata read interface functions. |
| 54:52 | 3 | CONTEXT_CODE | Yes | Both | Specifies the context of the field: 0: Platform (whole Intel TDX Module) 1: TD 2: TD VCPU Other: Reserved All metadata read and write interface functions (e.g., TDH.MNG.RD, TDH.MNG.WR, TDH.VP.RD, TDG.SYS.RDALL etc.) ignore this field when used as an input; they use an implicit value. |
| 55 | 1 | RESERVED | Yes | Both | Must be 0 |
| 61:56 | 6 | CLASS_CODE | Yes | Both | Identifies the class of the fields being accessed Class codes are defined in 3.10.3. |
| 62 | 1 | RESERVED | Yes | Both | Must be 0 |
| 63 | 1 | IGNORED | Yes | Both | This bit is ignored by the TDX Module Note: Previously, this bit was defined as NON_ARCH. |

Values Reserved for Software Use

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

5 **3.10.2. Meaning of Field Codes**

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

Table 3.56: Meaning of Field Codes

| Field Class | Field Code Meaning | | | Reference |
|-------------------|---|---------------------|---|---|
| VMCS | Field code is the architectural VMCS field code. The “HIGH” access type (for accessing the upper 32b of 64b fields) is not supported. | | | [Intel SDM, Vol. 3, 24.11.2 and App. B] |
| | Bits | Name | Description | |
| | 23:16 | RESERVED | Must be 0 | |
| | 15:0 | VMCS_FIELD_CODE | Bits 15:0 of the architectural VMCS field code Note: Bits 32:16 of the VMCS field code are implicitly 0. | |
| MSR Bitmap | Offset (in 8B units) from the beginning of the architectural MSR bitmaps page | | | [Intel SDM, Vol. 3, 24.6.9] |
| Secure EPT Root | Offset (in 8B units) from the beginning of the page | | | |
| 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 | Each field contains two 64-bit elements, with the values returned by CPUID for the leaf and sub-leaf, as follows: Element 0[31:0]: EAX Element 0[63:32]: EBX Element 1[31:0]: ECX Element 1[63:32]: EDX The field code is packed as shown below: | | | |
| | Bits | Name | Description | |
| | 23:17 | RESERVED | Must be 0 | |
| | 16 | LEAF_31 | Leaf number bit 31 | |
| | 15:9 | LEAF_6_0 | Leaf number bit 6:0 Note: Leaf bits 30:7 are implicitly 0. | |
| | 8 | SUBLEAF_NA | Sub-leaf not applicable flag | |
| | 7:1 | SUBLEAF_6_0 | Sub-leaf number bits 6:0 If SUBLEAF_NA is 1, then SUBLEAF_6_0 is all-1. Note: Sub-leaf bits 31:7 are implicitly 0. | |
| | 0 | ELEMENT_I | Element index within field | |
| GPR State | Architectural GPR number | | | |
| MSR State | Architectural MSR index, packed as shown below: | | | |
| | Bits | Description | | |
| | 23:14 | Reserved, must be 0 | | |

| Field Class | Field Code Meaning | | Reference |
|-----------------------|---|---|-----------|
| | 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 | | |

3.10.3. Class Codes

3.10.3.1. TDX Module Global Scope Field Class Codes

TDX Module global scope field classes are defined as follows:

Table 3.57: 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 |
| 26 | Memory Management |
| 27 | Measurement |
| 32 | Migration |
| 33 | Service TD |
| 34 | TD Partitioning |
| 48 | TDX Connect |

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

TD-scope field classes are defined as follows:

Table 3.58: TD Scope (TDR and TDCS) Field Class Codes Definition

| Class Code | Control Structure | Field Class Name |
|------------|-------------------|--------------------|
| 0 | TDR | TD Management |
| 1 | TDR | Key Management |
| 2 | TDR | TD Preserving |
| 3 | TDR | TDX I/O |
| 16 | TDCS | TD Management |
| 17 | TDCS | Execution Controls |

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

3.10.3.3. VCPU-Scope (TDVPS) Field Class Codes

TDVPS field classes are defined as follows:

Table 3.59: TD VCPU Scope (TDVPS) Field Class Codes Definition

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

3.10.4. Order of Field Identifiers

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

- 5 1. CONTEXT_CODE
2. CLASS_CODE
3. FIELD_CODE

3.10.5. MD_LIST_HEADER: Metadata List Header

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

10 **Table 3.60: MD_LIST_HEADER Definition**

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

3.11. Memory Buffer Types

3.11.1. Private Page List

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

Table 3.61: Private Page List Entry as Used by TDH.IMPORT.MEM

| 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 |
| 60:52 | RESERVED | Reserved: must be 0 (unless bit 63 indicates an invalid entry) |
| 61 | REMOVED | If INVALID (bit 63) is 0: <ul style="list-style-type: none"> • On input, must be 0. • On output, this bit indicates that the HPA is of a page removed by a CANCEL import operation. |
| 62 | PAMT_REMOVAL_HINT | If INVALID (bit 63) is 0: <ul style="list-style-type: none"> • On input, must be 0. • On output, if REMOVED (bit 61) is 1 and the TDX Module is configured for Dynamic PAMT, provides a hint that the PAMT pages mapping the removed page are empty and can be removed. |
| 63 | INVALID | A value of 1 indicates that this entry is invalid |

3.11.2. HPA_AND_SIZE: HPA and Size of a Buffer

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

Table 3.62: 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 |

5 3.11.3. HPA_AND_LAST: HPA and Last Byte Index of a Page-Aligned Buffer

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

Table 3.63: HPA_AND_LAST

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

3.12. Service TD Types

10 3.12.1. SERVTD_BINDING_TABLE: Service TD Binding Table

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

Table 3.64: Service TD Binding Entry Definition

| Field | Type | Offset (Bytes) | Size (Bytes) | Description |
|------------------|----------------------|----------------|--------------|--------------------------------------|
| STATE | SERVTD_BINDING_STATE | 0 | 1 | See below and [TDX Module Base Spec] |
| Reserved | | 1 | 1 | Must be 0 |
| TYPE | SERVTD_TYPE | 2 | 2 | See below and [TDX Module Base Spec] |
| Reserved | | 4 | 4 | Must be 0 |
| ATTR | SERVTD_ATTR | 8 | 8 | See below and [TDX Module Base Spec] |
| UUID | 256-bit blob | 16 | 32 | See [TDX Module Base Spec] |
| INFO_HASH | SHA384_HASH | 48 | 48 | See [TDX Module Base Spec] |
| Reserved | | 96 | 32 | Must be 0 |

TD-Preserving Update TDX Module Handoff Compatibility

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

Table 3.65: SERVTD_BINDING_TABLE Compatibility with TD Preserving Updates

| Module Handoff Version | Value |
|------------------------|-------|
| Minimum MODULE_HV | 0 |
| Maximum MODULE_HV | 0 |

3.12.2. SERVTD_BINDING_STATE: Service TD Binding State

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

Table 3.66: SERVTD_BINDING_STATE Values

| Value | Name |
|-------|--------------|
| 0 | NOT_BOUND |
| 1 | PRE_BOUND |
| 2 | BOUND |
| 3 | REBIND_BOUND |

3.12.3. SERVTD_TYPE: Service TD Binding Type

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

Table 3.67: SERVTD_TYPE Definition

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

3.12.4. SERVTD_ATTR: Service TD Binding Attributes

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

Table 3.68: SERVTD_ATTR Definition

| Bit(s) | Name | Description |
|--------|----------------------|---|
| 31:0 | 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 MROWNERCONFIG field when calculating SERVTD_INFO_HASH |

| Bit(s) | Name | Description |
|--------|-------------------------|--|
| 38 | IGNORE_RTMR0 | If set to 1, a value of 0 is used instead of the service TD's RTMR0 field when calculating SERVTD_INFO_HASH |
| 39 | IGNORE_RTMR1 | If set to 1, a value of 0 is used instead of the service TD's RTMR1 field when calculating SERVTD_INFO_HASH |
| 40 | IGNORE_RTMR2 | If set to 1, a value of 0 is used instead of the service TD's RTMR2 field when calculating SERVTD_INFO_HASH |
| 41 | IGNORE_RTMR3 | If set to 1, a value of 0 is used instead of the service TD's RTMR3 field when calculating SERVTD_INFO_HASH |
| 42 | IGNORE_SERVTD_HASH | If set to 1, a value of 0 is used instead of the service TD's SERVTD_HASH field when calculating SERVTD_INFO_HASH |
| 43 | IGNORE_MRSIGROOT | If set to 1, a value of 0 is used instead of the service TD's MRSIGROOT field when calculating SERVTD_INFO_HASH Enumeration: Support of this bit is enumerated by TDX_FEATURES0.TD_SIGNING_AND_SVN (bit 22). If not supported, it must be 0. |
| 44 | IGNORE_MRSIGNER | If set to 1, a value of 0 is used instead of the service TD's MRSIGNER field when calculating SERVTD_INFO_HASH Enumeration: Support of this bit is enumerated by TDX_FEATURES0.TD_SIGNING_AND_SVN (bit 22). If not supported, it must be 0. |
| 45 | IGNORE_PRODID | If set to 1, a value of 0 is used instead of the service TD's PROPID field when calculating SERVTD_INFO_HASH Enumeration: Support of this bit is enumerated by TDX_FEATURES0.TD_SIGNING_AND_SVN (bit 22). If not supported, it must be 0. |
| 46 | IGNORE_ISVSVN | If set to 1, a value of 0 is used instead of the service TD's ISVSVN field when calculating SERVTD_INFO_HASH Enumeration: Support of this bit is enumerated by TDX_FEATURES0.TD_SIGNING_AND_SVN (bit 22). If not supported, it must be 0. |
| 47 | IGNORE_MRCONFIGSVN | If set to 1, a value of 0 is used instead of the service TD's MRCONFIGSVN field when calculating SERVTD_INFO_HASH Enumeration: Support of this bit is enumerated by TDX_FEATURES0.SEALING (bit 12). |
| 48 | IGNORE_MROWNERCONFIGSVN | If set to 1, a value of 0 is used instead of the service TD's MROWNERCONFIGSVN field when calculating SERVTD_INFO_HASH Enumeration: Support of this bit is enumerated by TDX_FEATURES0.SEALING (bit 12). |
| Other | RESERVED | Must be 0 |

3.12.5. SERVTD_EXT_STRUCT: Extended Service TD Info

SERVTD_EXT_STRUCT provides additional attestation context about the TD's TCB.

Table 3.69: Service TD Extended Hash Definition

| Name | Offset | Size | Type | Description |
|----------------|--------|------|--------|--|
| INIT_INFO_HASH | 0 | 48 | SHA384 | Service TD binding entry's INFO_HASH at TDH.MR.FINALIZE time |

| Name | Offset | Size | Type | Description |
|------------------|--------|------|------------------|---|
| INIT_ATTR | 48 | 8 | MASK | Service TD binding entry's ATTR at TDH.MR.FINALIZE time |
| RESERVED | 56 | 8 | | Set to 0 |
| INIT_CPUSVN | 64 | 16 | | CPUSVN at TDH.MNG.INIT time |
| INIT_TEE_TCB_SVN | 80 | 16 | | TEE_TCB_SVN at TDH.MNG.INIT time |
| INIT_TEE_MODEL | 96 | 12 | TEE_MODEL_STRUCT | CPU and Platform Model information, as captured at TDH.SYS.INIT time. See 3.12.6 below. |
| RESERVED | 108 | 4 | | Set to 0 |
| CUR_INFO_HASH | 112 | 48 | SHA384 | Current Service TD binding entry's INFO_HASH |
| CUR_ATTR | 160 | 8 | MASK | Current Service TD binding entry's ATTR |
| RESERVED | 168 | 8 | | Set to 0 |
| RESERVED | 176 | 48 | SHA384 | Reserved for Cascading hash of Audit Log. Set to 0 |
| RESERVED | 224 | 48 | SHA384 | Set to 0 |

3.12.6. TEE_MODEL_STRUCT

TEE_MODEL_STRUCT encodes platform-specific identifiers, including the Family/Model (FM) and Platform ID, into a single value. This value is captured during TDH.SYS.INIT and is used to:

- Bind the TD's initial state to a specific platform configuration.
- Differentiate between SVNs of different platforms across operations such as migration, rebind, or attestation.

Table 3.70: TEE_MODEL_STRUCT Definition

| Name | Offset (Bytes) | Size (Bytes) | Type | Description |
|-------------|----------------|--------------|------------------|---|
| CUSTOM | 0 | 2 | Unsigned Integer | Custom data, set to 0. |
| PLATFORM_ID | 2 | 2 | Unsigned Integer | Platform ID as defined in the IA32_PLATFORM_ID MSR bits 52:50, other bits are 0. |
| FM | 4 | 4 | Unsigned Integer | Family and Model, formatted as in CPUID(1).EAX, with the Stepping value set to 0. |
| RESERVED | 8 | 4 | N/A | Reserved, set to 0x80000000. |

3.12.7. SERVTD_EXT_HASH

- 10 The SERVTD_EXT_HASH is the SHA384 hash of SERVTD_EXT_STRUCT.

When computing the hash for purpose of TDCS.SERVTD_ACCEPT_SERVTD_EXT_HASH, the CUR_INFO_HASH and CUR_ATTR fields are zeroed before computing the hash.

3.13. Migration Types

- 15 **Unreleased Feature:** Some of the text in this section is related to Non-Blocking Export, a feature which has not been released yet at the time of writing of this document. Details related to that feature serve as a preview and are subject to change.

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

3.13.1. MBMD: Migration Bundle Metadata

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

5 3.13.1.1. Generic MBMD Structure

The maximum overall size of MBMD is 128 bytes.

Table 3.71: 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 MBMD format, other migration bundle components format or migration protocol sequence require updating the protocol version. The 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 |
| IV_COUNTER | 16 | 8 | Monotonously incrementing counter, used as a component in the AES-GCM IV | As 0 | Yes |
| Type-Specific Information | 24 | Variable | Variable-sized additional information for each specific type of MBMD | Yes | No |
| MAC | 24+V | 16 | AES-256-GCM MAC over other MBMD fields and any associated migration data (all the migration pages) | No | No |

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

Table 3.72: TD-Scope Immutable Non-Memory State MBMD Fields Definition

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

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

5

Table 3.73: TD-Scope Mutable Non-Memory State MBMD Fields Definition

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

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

Table 3.74: 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 |

10 3.13.1.5. TD Private Memory MBMD Fields

Table 3.75: TD Private Memory MBMD Type-Specific Fields Definition

| Field | Offset (Bytes) | Size (Bytes) | Description | Included In MAC | Included in IV |
|---------------------|----------------|--------------|--|-----------------|----------------|
| NUM_GPAS | 24 | 2 | Number of entries in the GPA list | Yes | No |
| GPA_LIST_ATTRIBUTES | 26 | 1 | Attributes of the GPA list, see Table 3.76 below | Yes | No |
| RESERVED | 27 | 5 | Reserved, must be 0 | Yes | No |

Table 3.76: GPA_LIST_ATTRIBUTES

| Bits | Name | Description | | |
|------|--------|-----------------|--------------|--|
| 2:0 | FORMAT | GPA list format | | |
| | | Value | Name | Description |
| | | 0 | GPA_ONLY | A GPA list page is provided |
| | | 1 | GPA_AND_ATTR | A GPA list page and a page attributes list page are provided |

| Bits | Name | Description | | |
|------|----------|---------------------|----------|----------|
| | | Other | RESERVED | Reserved |
| 7:3 | RESERVED | Reserved: must be 0 | | |

3.13.1.6. *Epoch Token MBMD Fields*

Table 3.77: 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 |

5 3.13.1.7. *Abort Token MBMD Fields*

Table 3.78: Abort Token MBMD Fields Definition

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

3.13.1.8. *TD Migration Protocol Version Compatibility*

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

10

Table 3.79: MBMD Compatibility with TD Migration Versions

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

3.13.2. GPA List

3.13.2.1. *Overview*

15

A GPA list specifies a list of 4KB-aligned GPAs, each with associated attributes, required operation and status. A single 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 the OPERATION field’s value set to 0 (NOP).

Interface functions that process up to 512 GPAs, such as TDH.EXPORT.MEM, use a single GPA list page. The GPA list is specified by GPA_LIST_INFO, which contains the HPA of the GPA list page and the index of the first entry and last entry to be processed.

20

Interface functions that process up to 512² GPAs, such as TDH.MEM.SCAN, use multiple GPA list pages. The list of pages is specified by a page of GPA_LIST_INFO entries. The GPA list info page is specified by a PAGE_LIST_INFO, which contains the HPA of the GPA list info page and the index of the first entry and last entry to be processed.

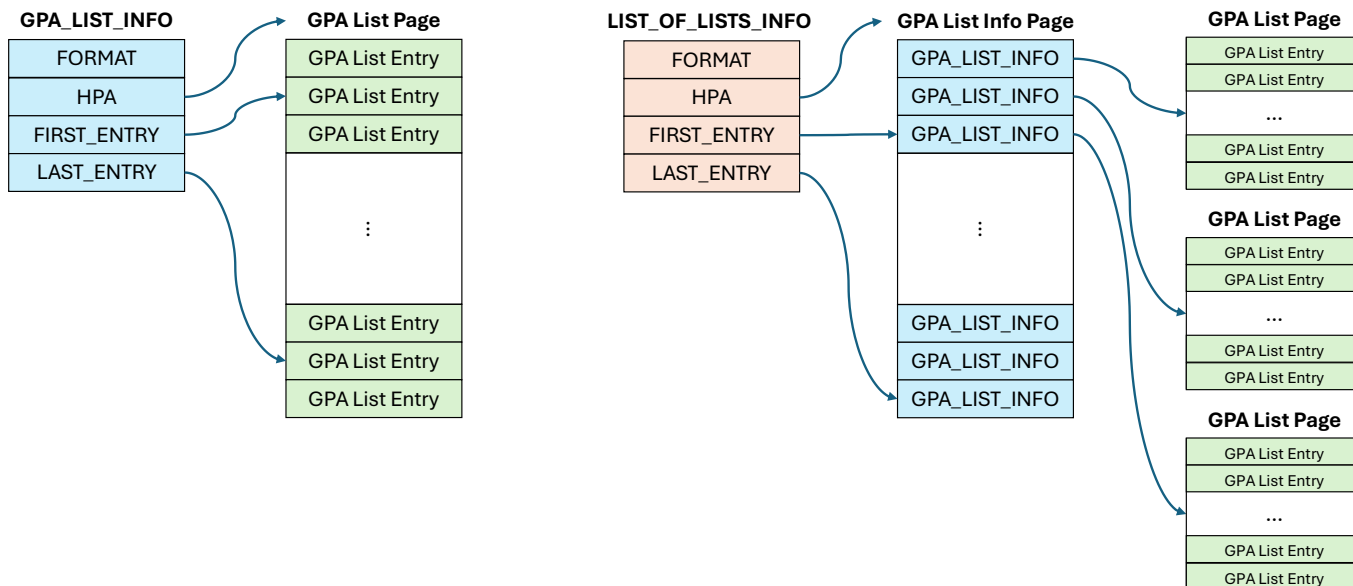


Figure 3.1: GPA List and List of Lists

3.13.2.2. 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. When used with some interface functions, it may provide list-of-lists information.

Table 3.80: GPA_LIST_INFO

| Bits | Name | Description | | |
|-------|--------------------|---|-----------------|---|
| 2:0 | FORMAT | GPA list format | | |
| | | Value | Name | Description |
| | | 0 | GPA_ONLY | A GPA list page is provided |
| | | 1 | GPA_AND_L2_ATTR | GPA list and L2 page attributes list pages are provided. This format is only used by TDH.EXPORT.MEM and TDH.IMPORT.MEM. It is mandatory for migrating partitioned TDs (which contain one or more L2 VMs). TDX Module support of this feature is enumerated by TDX_FEATURES0.PARTITIONED_TD_MIGRATION, readable by TDH.SYS.RD* (see 3.3.3.1). |
| | | 2 | LIST_OF_LISTS | A list-of-lists page is provided |
| Other | RESERVED | Reserved | | |
| 11:3 | FIRST_ENTRY | Index of the first entry of the list to be processed | | |
| 51:12 | HPA | Bits 51:12 of the host physical address (including HKID) of the GPA list page, which must be a shared HPA | | |
| 54:52 | RESERVED | Reserved: must be 0 | | |
| 63:55 | LAST_ENTRY | Index of the last entry in the GPA list | | |

The following special values indicate an empty list. This is used as the output of TDH.MEM.SCAN.COMP/RANGE.

- FIRST_ENTRY set to all-1 (511)

- LAST_ENTRY set to 0

3.13.2.3. GPA List Entry

Table 3.81 below shows the format of a GPA list entry as used. The GPA list entry format is designed so that the output of TDH.EXPORT.BLOCKW, TDH.MEM.SCAN.COMP and TDH.MEM.SCAN.RANGE can be used directly with TDH.EXPORT.MEM, and the output of TDH.EXPORT.MEM can be used directly with TDH.IMPORT.MEM.

Table 3.81: GPA List Entry Definition

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

3.13.2.4. GPA List Entry Details

3.13.2.4.1. LEVEL

- 10 LEVEL specifies the GPA mapping size for this entry, as shown in the table below.

Table 3.82: LEVEL Definition

| Level | GPA Mapping Size |
|-------|------------------|
| 0 | 4KB |
| 1 | 2MB |
| 2 | 1GB |
| 3 | Reserved |

For TDH.EXPORT.BLOCKW, TDH.EXPORT.MEM, TDH.EXPORT.RESTORE and TDH.IMPORT.MEM only level 0 (4KB) is supported.

⁹ OPERATION value for TDH.MEM.SCAN.RANGE(DSCAN) and TDH.MEM.SCAN.COMP(DCHECK) is always MIGRATE.

¹⁰ STATUS value for TDH.MEM.SCAN.RANGE(DSCAN) and TDH.MEM.SCAN.COMP(DCHECK) is always SUCCESS.

3.13.2.4.2. PENDING

Indicates that the page is Pending. Note that the actual SEPT entry state may be one of multiple PENDING* states.

3.13.2.4.3. STATE

Provides a hint about the page state.

5

Table 3.83: STATE Values Definition for TDH.MEM.SCAN*

| Value | Output | |
|-------|-------------------|--|
| | Name | Description |
| 0 | NOT_EXPORTED | Page is a candidate for initial export. |
| 1 | EXPORTED_MODIFIED | Page is a candidate for re-export due to updated content or attributes. |
| 2 | EXPORTED_BLOCKED | Page is a candidate for re-export due to being blocked. Blocking is an interim state followed by some other memory management operation; the host VMM may decide to defer the re-export. |
| 3 | EXPORTED_REMOVED | Page is a candidate for re-export due to being removed. |

3.13.2.4.4. L2_MAP

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

10

3.13.2.4.5. 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.

OPERATION Values for TDH.EXPORT.BLOCKW

15

Table 3.84: OPERATION Values Definition for TDH.EXPORT.BLOCKW

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

OPERATION Values for TDH.MEM.SCAN*

Table 3.85: OPERATION / STATE Values Definition for TDH.MEM.SCAN*

| Value | Input | | Output | |
|-------|-------|-------------|----------|-------------|
| | Name | Description | Name | Description |
| 0 | N/A | N/A | RESERVED | Reserved |
| 1 | N/A | N/A | EXPORT | Export |
| 2 | N/A | N/A | RESERVED | Reserved |
| 3 | N/A | N/A | RESERVED | Reserved |

OPERATION Values for TDH.EXPORT.MEM (For Write Blocking Based Export)

Note that on input, the MIGRATE operation is represented by two possible values. This is done for direct compatibility with the TDH.EXPORT.BLOCKW output.

Table 3.86: OPERATION Values Definition for TDH.EXPORT.MEM (Write Blocking Export)

| Value | Input | | Output | |
|-------|---------|--|-----------|---|
| | Name | Description | Name | Description |
| 0 | NOP | No operation | NOP | Not exported |
| 1 | MIGRATE | Export (may result in MIGRATE or REMIGRATE, as determined by the TDX Module) | MIGRATE | Initial export during this migration session or following a CANCEL |
| 2 | CANCEL | Cancel previous export | CANCEL | Cancellation of a previous export Not applicable for S4 hibernation. |
| 3 | MIGRATE | Export | REMIGRATE | Re-export of updated content or attributes |

5

OPERATION Values for TDH.EXPORT.MEM (For Non-Blocking Export)**Table 3.87: OPERATION Values Definition for TDH.EXPORT.MEM (Non-Blocking Export)**

| Value | Input | | Output | |
|-------|--------|--|-----------|---|
| | Name | Description | Name | Description |
| 0 | NOP | No operation | NOP | Not exported |
| 1 | EXPORT | Export request (may result in MIGRATE, REMIGRATE or CANCEL, as determined by the TDX Module) | MIGRATE | Initial export during this migration session or following a CANCEL |
| 2 | NOP | No operation | CANCEL | Cancellation of a previous export Not applicable for S4 hibernation. |
| 3 | NOP | No operation | REMIGRATE | Re-export of updated content or attributes |

OPERATION Values for TDH.IMPORT.MEM**Table 3.88: OPERATION Values Definition for TDH.IMPORT.MEM**

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

10

OPERATION Values for TDH.EXPORT.RESTORE**Table 3.89: OPERATION Values Definition for TDH.EXPORT.RESTORE**

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

3.13.2.4.6. MIG_TYPE**Table 3.90: MIG_TYPE Values Definition**

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

3.13.2.4.7. STATUS**Table 3.91: STATUS Values Definition**

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

| Value | Name | Description |
|-------|------------------------------|--|
| 16 | INVALID_MIGRATION_BUFFER_HPA | The provided migration buffer HPA is not valid |
| 17 | PAGE_DIRTY | Page not exported because the SEPT entry's Dirty bit was set |
| 31-18 | Reserved | Reserved |

3.13.2.5. TD Migration Protocol Version Compatibility

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

Table 3.92: GPA List Compatibility with TD Migration Versions

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

5 3.13.3. 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.

10 3.13.3.1. Migration Buffers List Entry

Table 3.93: Migration Buffers List Entry

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

3.13.4. Page Attributes List

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

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

3.13.5. Memory Migration Page MAC List

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

3.13.6. Non-Memory State Migration Buffers List

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

3.13.6.1. PAGE_LIST_INFO: HPA and Attributes of a Page List

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

Table 3.94: PAGE_LIST_INFO

| Bits | Name | Description |
|-------|-------------------|---|
| 11:0 | RESERVED | Reserved: must be 0 |
| 51:12 | HPA | Bits 51:12 of the host physical address (including HKID) of the page list, which must be a shared HPA |
| 54:52 | RESERVED | Reserved: must be 0 |
| 63:55 | LAST_ENTRY | Index of the last entry in the page list |

4. TD Metadata (Non-Memory State)

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

4.1. TD-Scope Metadata

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

5 4.1.1. TD-Scope Metadata Description Files

Most of the TD-scope metadata information is provided in separate files:

- **td_scope_metadata.pdf** provides a human-readable TD-scope metadata table.
- **td_scope_metadata.csv** provides the same table in a Comma-Separated Values (CSV) format. This file can be imported, if required, to tools such as MS Excel for further processing.

10 The tables below describe the information provided by those files.

In addition to the above files, **td_scope_metadata.json**, a JSON format file with the same information is provided.

Table 4.1: TD-Scope Metadata Table Description

| Column | Description |
|--------------------------------|--|
| TDX_FEATURES Enum. Bits | The set of TDX_FEATURES field bits, readable by TD*.SYS.RD*, which enumerate support for this field. One or more comma-separated bit numbers may be specified. If the value of any of those bits is, the field is supported. The special value “Always” means that the field is always supported. The special value “None” means that there is no enumeration for the support of this field. This is only used for fields that are accessible only for debuggable TDs. |
| Class | Field class, see below for details |
| Field | Field name |
| Description | Field description |
| Type | Field type |
| VM Applic. | For field arrays where each entry is associated with a VM, this column specifies VM applicability: <ul style="list-style-type: none"> • L1_AND_L2 • L1_ONLY • L2_ONLY |
| Init Value | Initial value |
| Field Size (Bytes) | Field size |
| Max Num Fields | Maximum number of fields in an array. The description text specified how the actual number of fields in enumerated. A value of “None” indicates that no maximum number is specified. |
| Num Elem. | Number of elements within each field |
| Elem. Size (Bytes) | Size of each element |
| Base FIELD_ID (Hex) | Base FIELD_ID |
| VMM Access Prod. | Host VMM access for production TDs |
| VMM Access Debug | Host VMM access for debuggable TDs |
| Guest Access | Guest TD access |
| MigTD Access | Migration TD access |
| VMM Wr Mask Prod. | Host VMM write mask for production TDs |

| Column | Description |
|--------------------------|---|
| | A bit value of 1 indicates that the corresponding field bit is writable. However, actual writability may be subject to supported features and configuration, as described in other places in this document. |
| VMM Wr Mask Debug | Host VMM write mask for debuggable TDs A bit value of 1 indicates that the corresponding field bit is writable. However, actual writability may be subject to supported features and configuration, as described in other places in this document. |
| Guest Wr Mask | Guest TD write mask A bit value of 1 indicates that the corresponding field bit is writable. However, actual writability may be subject to supported features and configuration, as described in other places in this document. |
| MigTD Wr Mask | Migration TD write mask A bit value of 1 indicates that the corresponding field bit is writable. However, actual writability may be subject to supported features and configuration, as described in other places in this document. |

4.1.2. TDR

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

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

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

TRD fields are divided into the following classes:

Table 4.2: TDR Field Classes Definition

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

4.1.3. TDCS

4.1.3.1. Overview

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.

Most TDCS fields are described in a separate JSON format file **td_scope_metadata.json**. The sections below provide additional information for specific fields.

TDCS fields are divided into the following classes:

Table 4.3: TDCS Field Classes Definition

| Field Class | Description |
|--------------------------------|--|
| TD Management | These fields are used to manage the TDCS, its descendent TD private memory pages and control structure pages. |
| TD Execution Control | Control the execution of the guest TD: some TD execution control fields are provided as an input to TDH.MNG.INIT, and some of those are included in the TDG.MR.REPORT. |
| TLB Epoch Tracking | Track the TLB epoch of the guest TD – see the [TDX Module Base Spec] for details |
| Measurement | TD measurement registers and associated fields – see the [TDX Module Base Spec] for details |
| Migration | TDCS fields that control TD migration |
| MIGSC Links | Links to Migration Stream Context pages |
| Service TD | TDCS fields that control Service TD binding and operation |
| MSR Bitmaps | MSR bitmaps that control VM exit from the guest TD on RDMSR/WRMSR are common to all TD VCPUs and thus are stored as part of TDCS. |
| Secure EPT Root Page | The root page (PML5 or PML4) of the secure EPT |
| L2 Secure EPT Root[3:1] | The root pages (PML5 or PML4) of the secure EPTs associated with L2 VM 1, 2 and 3 |

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

5 **4.1.3.2. TDCS.FEATURE_PARAVIRT_CTRL**

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

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

Table 4.4: TDCS.FEATURE_PARAVIRT_CTRL Definition

| Bits | Paravirtualized Feature Name & Applicable Linux Kernel Feature Name | Backward Compatible (TD_CTL.S.REDUCE_VE is 0) | Reduced #VE (TD_CTL.S.REDUCE_VE is 1, FEATURE_PARAVIRT_CTRL bit is 0) | Reduced #VE with Paravirtualization (TD_CTL.S.REDUCE_VE is 1, FEATURE_PARAVIRT_CTRL bit is 1) |
|------|---|---|---|---|
| 0 | CORE_CAPABILITIES (X86_FEATURE_CORE_CAPABILITIES) | Controls IA32_CORE_CAPABILITIES paravirtualization, enumerated by virtual CPUID(7,0).EDX[30] (support IA32_CORE_CAPABILITIES) | <ul style="list-style-type: none"> Virtual CPUID(7,0).EDX[30] is forced to 0. Guest TD access to applicable MSRs results in a #GP(0). | <ul style="list-style-type: none"> Virtual CPUID(7,0).EDX[30] is configured by the host VMM. Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT). |
| | | Controls Direct Cache Access paravirtualization, enumerated by virtual CPUID(1).ECX[18] (DCA) | | |

| Bits | Paravirtualized Feature Name & Applicable Linux Kernel Feature Name | Backward Compatible (TD_CTL.S.REDUCE_VE is 0) | Reduced #VE (TD_CTL.S.REDUCE_VE is 1, FEATURE_PARAVIRT_CTL.S bit is 0) | Reduced #VE with Paravirtualization (TD_CTL.S.REDUCE_VE is 1, FEATURE_PARAVIRT_CTL.S bit is 1) |
|------|---|---|---|---|
| 1 | DCA (X86_FEATURE_DCA) | <ul style="list-style-type: none"> Virtual CPUID(1).ECX[18] is configured by the host VMM. Virtual CPUID(9) results in a #VE(CONFIG_PARAVIRT). Guest TD access to applicable MSRs may result in a #GP(0) or #VE(CONFIG_PARAVIRT), depending on virtual CPUID(1).ECX[18] value. | <ul style="list-style-type: none"> Virtual CPUID(1).ECX[18] is forced to 0. Virtual CPUID(9) returns all-0. Guest TD access to applicable MSRs results in a #GP(0). | <ul style="list-style-type: none"> Virtual CPUID(1).ECX[18] is configured by the host VMM. If configured as 0: <ul style="list-style-type: none"> Virtual CPUID(9) returns all-0. Guest TD access to applicable MSRs results in a #GP(0). Else (configured as 1): <ul style="list-style-type: none"> Virtual CPUID(9) results in a #VE(CONFIG_PARAVIRT). Guest TD access to applicable MSRs results in a #VE(CONFIG_PARAVIRT). |
| 2 | EST (X86_FEATURE_EST) | Controls Enhanced Intel SpeedStep technology paravirtualization, enumerated by Virtual CPUID(1).ECX[7] (Enhanced Intel SpeedStep technology) | | |
| | | <ul style="list-style-type: none"> Virtual CPUID(1).ECX[7] is configured by the host VMM. Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT). | <ul style="list-style-type: none"> Virtual CPUID(1).ECX[7] is forced to 0. Guest TD access to applicable MSRs results in a #GP(0). | <ul style="list-style-type: none"> Virtual CPUID(1).ECX[7] is configured by the host VMM. Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT). |
| 3 | MCA (X86_FEATURE_MCA) | Controls Machine Check Architecture paravirtualization, enumerated by virtual CPUID(1).EDX[7] (Machine Check Exception) and virtual CPUID(1).EDX[14] (Machine Check Architecture) | | |
| | | <ul style="list-style-type: none"> Virtual CPUID(1).EDX[7] and virtual CPUID(1).EDX[14] are forced to 1. Guest TD access to applicable MSRs result in a #VE(CONFIG_PARAVIRT). CR4.MCE is fixed-1. Guest TD attempt to clear it result in a #VE(CONFIG_PARAVIRT). | <ul style="list-style-type: none"> Virtual CPUID(1).EDX[7] and virtual CPUID(1).EDX[14] are forced to 0. Guest TD access to applicable MSRs results in a #GP(0). CR4.MCE is initialized to 1. The guest TD may clear CR4.MCE but not set it back to 1; attempt to do so results in a #GP(0). | <ul style="list-style-type: none"> Virtual CPUID(1).EDX[7] and virtual CPUID(1).EDX[14] are configured by the host VMM. If virtual CPUID(1).EDX[14] is 0, guest TD access to applicable MSRs results in a #GP(0). Else, it results in a #VE(CONFIG_PARAVIRT). CR4.MCE is initialized to 1. If virtual CPUID(1).EDX[7] is 0, the guest TD may clear CR4.MCE but not set it back to 1; attempt to do so results in a #GP(0). Else, guest TD is allowed to modify CR4.MCE. |
| 4 | MTRR (X86_FEATURE_MTRR) | Controls Memory Type Range Registers paravirtualization, enumerated by virtual CPUID(1).EDX[12] (Memory Type Range Registers) | | |
| | | <ul style="list-style-type: none"> Virtual CPUID(1).EDX[12] is forced to 1. Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT). | <ul style="list-style-type: none"> Virtual CPUID(1).EDX[12] is forced to 0. Guest TD access to applicable MSRs results in a #GP(0). | <ul style="list-style-type: none"> Virtual CPUID(1).EDX[12] is configured by the host VMM. Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT). |
| | | Controls PCONFIG paravirtualization, enumerated by virtual CPUID(7,0).EDX[18] (PCONFIG) | | |

| Bits | Paravirtualized Feature Name & Applicable Linux Kernel Feature Name | Backward Compatible (TD_CTL.S.REDUCE_VE is 0) | Reduced #VE (TD_CTL.S.REDUCE_VE is 1, FEATURE_PARAVIRT_CTL.S bit is 0) | Reduced #VE with Paravirtualization (TD_CTL.S.REDUCE_VE is 1, FEATURE_PARAVIRT_CTL.S bit is 1) |
|------|---|---|---|---|
| 5 | PCONFIG (X86_FEATURE_PCONFIG) | <ul style="list-style-type: none"> Virtual CPUID(7,0).EDX[18] is configured by the host VMM. Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT). | <ul style="list-style-type: none"> Virtual CPUID(7,0).EDX[18] is forced to 0. Guest TD access to applicable MSRs results in a #GP(0). | <ul style="list-style-type: none"> Virtual CPUID(7,0).EDX[18] is configured by the host VMM. Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT). |
| 6 | RDT_A (X86_FEATURE_RDT_A) | Controls RDT-A paravirtualization, enumerated by virtual CPUID(7,0).EBX[15] (RDT-A) | | |
| | | <ul style="list-style-type: none"> Virtual CPUID(7,0).EBX[15] is configured by the host VMM. Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT). | <ul style="list-style-type: none"> Virtual CPUID(7,0).EBX[15] is forced to 0. Guest TD access to applicable MSRs results in a #GP(0). | <ul style="list-style-type: none"> Virtual CPUID(7,0).EBX[15] is configured by the host VMM. Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT). |
| 7 | RDT_M (X86_FEATURE_CQM) | Controls RDT-M paravirtualization, enumerated by virtual CPUID(7,0).EBX[12] (RDT-M) | | |
| | | <ul style="list-style-type: none"> Virtual CPUID(7,0).EBX[12] is configured by the host VMM. Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT). | <ul style="list-style-type: none"> Virtual CPUID(7,0).EBX[12] is forced to 0. Guest TD access to applicable MSRs results in a #GP(0). | <ul style="list-style-type: none"> Virtual CPUID(7,0).EBX[12] is configured by the host VMM. Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT). |
| 8 | ACPI (X86_FEATURE_ACPI) | Controls Thermal Monitor and Software Controlled Clock Facilities paravirtualization, enumerated by virtual CPUID(1).EDX[22] (ACPI) | | |
| | | <ul style="list-style-type: none"> Virtual CPUID(1).EDX[22] is configured by the host VMM. Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT). | <ul style="list-style-type: none"> Virtual CPUID(1).EDX[22] is forced to 0. Guest TD access to applicable MSRs results in a #GP(0). | <ul style="list-style-type: none"> Virtual CPUID(1).EDX[22] is configured by the host VMM. Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT). |
| 9 | TM2 (X86_FEATURE_TM2) | Controls MSR_THERM2_CTL paravirtualization, enumerated by virtual CPUID(1).ECX[8] (TM2) | | |
| | | <ul style="list-style-type: none"> Virtual CPUID(1).ECX[8] is configured by the host VMM. Guest TD access to MSR_THERM2_CTL may result in a #VE(CONFIG_PARAVIRT). | <ul style="list-style-type: none"> Virtual CPUID(1).ECX[8] is forced to 0. Guest TD access to MSR_THERM2_CTL results in a #GP(0). | <ul style="list-style-type: none"> Virtual CPUID(1).ECX[8] is configured by the host VMM. Guest TD access to MSR_THERM2_CTL may result in a #VE(CONFIG_PARAVIRT). |
| 10 | TME (X86_FEATURE_TME) | Controls Total Memory Encryption paravirtualization, enumerated by virtual CPUID(7,0).ECX[13] (TME_EN) | | |
| | | <ul style="list-style-type: none"> Virtual CPUID(7,0).ECX[13] is configured by the host VMM. Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT). | <ul style="list-style-type: none"> Virtual CPUID(7,0).ECX[13] is forced to 0. Guest TD access to applicable MSRs results in a #GP(0). | <ul style="list-style-type: none"> Virtual CPUID(7,0).ECX[13] is configured by the host VMM. Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT). |
| 11 | TSC_DEADLINE | Controls IA32_TSC_DEADLINE MSR paravirtualization, enumerated by virtual CPUID(1).ECX[24] (TSC deadline) | | |

| Bits | Paravirtualized Feature Name & Applicable Linux Kernel Feature Name | Backward Compatible (TD_CTL.S.REDUCE_VE is 0) | Reduced #VE (TD_CTL.S.REDUCE_VE is 1, FEATURE_PARAVIRT_CTL.S bit is 0) | Reduced #VE with Paravirtualization (TD_CTL.S.REDUCE_VE is 1, FEATURE_PARAVIRT_CTL.S bit is 1) |
|-------|---|--|---|---|
| | (X86_FEATURE_TSC_DEADLINE_TIMER) | <ul style="list-style-type: none"> Virtual CPUID(1).ECX[24] is configured by the host VMM. Guest TD access to the IA32_TSC_DEADLINE MSR may result in a #GP(0) or #VE(CONFIG_PARAVIRT), depending on virtual CPUID(1).ECX[24]. | <ul style="list-style-type: none"> Virtual CPUID(1).ECX[24] is forced to 0. Guest TD access to the IA32_TSC_DEADLINE MSR results in a #GP(0). | <ul style="list-style-type: none"> Virtual CPUID(1).ECX[24] is configured by the host VMM. Guest TD access to the IA32_TSC_DEADLINE MSR may result in a #VE(CONFIG_PARAVIRT). |
| 63:12 | RESERVED | Must be 0 | | |

4.1.3.3. TDCS.PID_MODE[4]

Unreleased Feature: This section is related to Enhanced Interrupt Virtualization, a feature which has not been released yet at the time of writing of this document. Details provided below serve as a preview and are subject to change.

Array of per-VM PID modes.

Table 4.5: TDCS.PID_MODE

| Value | Name | Description |
|-------|----------|---|
| 0 | REG_PID | Default value: Regular PID in shared memory (L1 only) For L1, a applies either when legacy interrupt virtualization is used (TDCS.INTR_CONFIG_MODE is CONFIG_LEGACY) or when enhanced interrupt virtualization has been configured (TDCS.INTR_CONFIG_MODE is CONFIG_DONE). For L2, this value indicates no PID is configured. |
| 1 | SEC_PID | Secure PID Applies only when enhanced interrupt virtualization has been configured (TDCS.INTR_CONFIG_MODE is CONFIG_DONE). |
| 2 | SHR_PID | Shared PID Applies only when enhanced interrupt virtualization has been configured (TDCS.INTR_CONFIG_MODE is CONFIG_DONE). |
| 3 | DUAL_PID | Dual PID: Secure PID and Shared PID Applies only when enhanced interrupt virtualization has been configured (TDCS.INTR_CONFIG_MODE is CONFIG_DONE). |
| Other | Reserved | Reserved |

4.1.3.4. TDCS.TD_CTL.S

10 TD_CTL.S is a bitmap of TD controls that may be modified by the guest TD (L1 only) during TD run time.

Table 4.6: TDCS.TD_CTL5 Definition

| Bits | Name | Description |
|------|---------------------------|---|
| 0 | PENDING_VE_DISABLE | <p>Controls the way guest TD access to a PENDING page is processed:</p> <p>0 (default): An EPT violation due to guest TD access to a PENDING page results in a #VE(PENDING).</p> <p>1: An EPT violation due to guest TD access to a PENDING page results in a TD exit.</p> <p>The above applies only to L1. L2 VM access to a PENDING page always results in an L2→L1 exit.</p> <p>PENDING_VE_DISABLE's initial value is copied from the TD's ATTRIBUTES.SEPT_VE_DISABLE. If the TD's CONFIG_FLAGS.FLEXIBLE_PENDING_VE is 1, the TD is allowed to modify PENDING_VE_DISABLE using TDG.VM.WR.</p> <p>Enumeration: TDX Module support of PENDING_VE_DISABLE is enumerated by TDX_FEATURES0.PENDING_EPT_VIOLATION_V2 (bit 16). If not supported, PENDING_VE_DISABLE must be 0.</p> |
| 1 | ENUM_TOPOLOGY | <p>Controls the enumeration of virtual platform topology:</p> <p>0 (default): Guest TD execution of CPUID(0xB) or CPUID(0x1F) results in a #VE(CONFIG_PARAVIRT). CPUID(1).EBX[31:24] returns the least significant 8 bits of the VCPU index. RDMSR of IA32_X2APIC_APICID (0x802) results in #VE(CONFIG_PARAVIRT).</p> <p>1: Guest TD execution of CPUID(0xB) or CPUID(0x1F) returns the virtual topology information configured by the host VMM. CPUID(1).EBX[31:24] returns the least significant 8 bits of the virtual x2APIC_ID configured by the host VMM. RDMSR of IA32_X2APIC_APICID (0x802) returns the virtual x2APIC_ID.</p> <p>ENUM_TOPOLOGY can only be set to 1 if x2APIC_ID has been properly configured with unique values for each VCPU. The guest TD can read TDCS.TOPOLOGY_ENUM_CONFIGURED using TDG.VM.RD to check that. ENUM_TOPOLOGY is implicitly set by the TDX Module if the guest TD sets REDUCE_VE.</p> <p>Enumeration: TDX Module support of ENUM_TOPOLOGY is enumerated by TDX_FEATURES0.TOPOLOGY_ENUM (bit 20). If not supported, ENUM_TOPOLOGY must be 0.</p> |
| 2 | VIRT_CPUID2 | <p>Controls the virtualization of CPUID(2):</p> <p>0 (default): Guest TD execution of CPUID(2) results in a #VE(CONFIG_PARAVIRT).</p> <p>1: Guest TD execution of CPUID(2) returns fixed values EAX=0x00FEFF01, EBX=0, ECX=0 and EDX=0, meaning "cache data is returned by CPUID leaf 0x4" and "TLB data is returned by CPUID leaf 0x18".</p> <p>VIRT_CPUID2 is implicitly set by the TDX Module if the guest TD sets REDUCE_VE.</p> <p>Enumeration: TDX Module support of VIRT_CPUID2 is enumerated by TDX_FEATURES0.CPUID2_VIRT (bit 29). If not supported, VIRT_CPUID2 must be 0.</p> |
| 3 | REDUCE_VE | <p>Allows the guest TD to control the way #VE is injected by the TDX Module on guest TD execution of CPUID, RDMSR/WRMSR and other instructions:</p> <p>0 (default): No change to default behavior.</p> |

| Bits | Name | Description |
|------|----------------------|--|
| | | <p>1: #VE injected on guest TD execution of CPUID, RDMSR/WRMSR and other instructions is greatly reduced. Some #VE injection depends on CPUID configuration of paravirtualized CPU features using TDCS.FEATURE_PARAVIRT_CTRL, see 4.1.3.2 above.</p> <p>When the guest TD sets REDUCE_VE to 1, the TDX Module also forces ENUM_TOPOLOGY and VIRT_CPUID to 1. REDUCE_VE can only be set to 1 if x2APIC_ID has been properly configured with unique values for each VCPU. The guest TD can read TDCS.TOPOLOGY_ENUM_CONFIGURED using TDG.VM.RD to check that.</p> <p>For a list of virtual CPUID values, MSRs and instructions impacted by REDUCE_VE, see Ch. 2.</p> <p>Enumeration: TDX Module support of REDUCE_VE is enumerated by TDX_FEATURES0.VE_REDUCTION (bit 30). If not supported, REDUCE_VE must be 0.</p> |
| 4 | ALLOW_HW_KEYS | <p>Controls whether a migratable TD can request a sealing key using TDG.MR.KEY.GET.</p> <p>0 (default): If the TD's ATTRIBUTES.MIGRATABLE is 1, the TD can't request a sealing key using TDG.MR.KEY.GET.</p> <p>1: The TD can request a sealing key using TDG.MR.KEY.GET regardless of ATTRIBUTES.MIGRATABLE.</p> <p>Enumeration: TDX Module supports ALLOW_HW_KEYS if both TDX_FEATURES0.TD_MIGRATION (bit 0) and TDX_FEATURES0.SEALING (bit 12) are enumerated as 1. If not supported, ALLOW_HW_KEYS must be 0.</p> |
| 62:5 | RESERVED | Must be 0 |
| 63 | LOCK | <p>Controls locking of TD-writable virtualization controls.</p> <p>0 (default): No change to default behavior.</p> <p>1: Control fields are locked and can't be modified.</p> <p>The following TD-writable control fields are impacted:</p> <ul style="list-style-type: none"> • TDCS.TD_CTL5 • TDCS.FEATURE_PARAVIRT_CTL5 • TDVPS.CPUID_SUPERVISOR_VE • TDVPS.CPUID_USER_VE • TDVPS.CPUID_CONTROL <p>Enumeration: TDX Module support of LOCK is enumerated by TDX_FEATURES0.VE_REDUCTION (bit 30). If not supported, must be 0.</p> |

4.1.3.5. TDCS.VM_CTL5

VM_CTL5 is an array of 4 64-bit control fields, with entries for L1 and each L2 VM. VM_CTL5 is writable by the host VMM using TDH.MNG.WR. It may be modified during TD run time and is migratable.

Table 4.7: TDCS.VM_CTLs Per-VM Entry Definition

| Bit | Name | Description | Default Value | TDX_FEATURES0 Enumeration Bit |
|-------|---------------------------------------|---|---------------|-------------------------------|
| 0 | EPT_VIOLATION_ON_L2_SEPT_WALK_FAILURE | If set, a TDCALL flow that encounters an L2 SEPT walk error due to a missing SEPT page causes a TD exit with an EPT violation exit reason. Else, it returns an error code to the guest. Applies only to L2 SEPTs. | 0 | Always available |
| 1 | HINT_ON_VINTR_PENDING | If set for the current VM, then on any synchronous TD exit (initiated by TDG.VP.VMCALL) from any VM, if TDVPS.VCPU_STATE_DETAILS.VINTR_PENDING for the current VM is 1, then the IMM_RESUME_HINT flag is set. Else, the IMM_RESUME_HINT flag is not impacted by this condition. | 0 | ENHANCED_INTR_STATE (bit 40) |
| 2 | HINT_ON_VINTR_IN_SERVICE | If set for the current VM, then on any synchronous TD exit (initiated by TDG.VP.VMCALL) from any VM, if TDVPS.VCPU_STATE_DETAILS.VINTR_IN_SERVICE for the current VM is 1, then the IMM_RESUME_HINT flag is set. Else, the IMM_RESUME_HINT flag is not impacted by this condition. | 0 | ENHANCED_INTR_STATE (bit 40) |
| 3 | HINT_ON_VNMI_PENDING | If set for the current VM, then on any synchronous TD exit (initiated by TDG.VP.VMCALL) from any VM, if TDVPS.VCPU_STATE_DETAILS.VNMI_PENDING for the current VM is 1, then the IMM_RESUME_HINT flag is set. Else, the IMM_RESUME_HINT flag is not impacted by this condition. | 0 | ENHANCED_INTR_STATE (bit 40) |
| Other | RESERVED | Reserved, must be 0 | 0 | N/A |

4.2. TDVPS: VCPU-Scope Metadata

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

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

4.2.1. Overview

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

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

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

TDVPS fields are divided into the following classes:

Table 4.8: 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 L1 VM architectural VMCS |
| VAPIC | The TD VCPU’s Virtual APIC page |
| VE_INFO | Holds Virtualization Exception (#VE) information |
| Guest GPR State | TD VCPU’s general-purpose register state |
| Guest MSR State | TD VCPU’s MSR state |
| Guest Extended State | TD VCPU’s extended state |
| VMCS[3:1] | VMCS pages of L2 VM 1, 2 and 3 |
| MSR Bitmaps[3:1] | MSR bitmap pages of L2 VM 1, 2 and 3 |
| MSR Bitmaps Shadow[3:1] | MSR bitmap shadow pages of L2 VM 1, 2 and 3 |

4.2.2. TDVPS (excluding TD and L2 VMCSes)

5 Most TDVPS fields are described in a separate JSON format file **vcpu_scope_metadata.json**. The sections below provide additional information for specific fields.

4.2.2.1. TDVPS Description Files

Most TDVPS fields information is provided in separate files:

- **vcpu_scope_metadata.pdf** provides a human-readable TDVPS table.
- 10 • **vcpu_scope_metadata.csv** provides the same table in a Comma-Separated Values (CSV) format. This file can be imported, if required, to tools such as MS Excel for further processing.

The tables below describe the information provided by those files.

In addition to the above files, **vcpu_scope_metadata.json**, a JSON format file with the same information is provided.

Table 4.9: TDVPS Table Description

| Column | Description |
|--------------------------------|--|
| TDX_FEATURES Enum. Bits | The set of TDX_FEATURES field bits, readable by TD*.SYS.RD*, which enumerate support for this field. One or more comma-separated bit numbers may be specified. If the value of any of those bits is, the field is supported. The special value “Always” means that the field is always supported. The special value “None” means that there is no enumeration for the support of this field. This is only used for fields that are accessible only for debuggable TDs. |
| Class | Field class, see above for details |
| Field | Field name |
| Description | Field description |
| Type | Field type |
| VM Applic. | For field arrays where each entry is associated with a VM, this column specifies VM applicability: <ul style="list-style-type: none"> • L1_AND_L2 • L1_ONLY • L2_ONLY |

| Column | Description |
|---------------------|---|
| Init Value | Initial value |
| Field Size (Bytes) | Field size |
| Max Num Fields | Maximum number of fields in an array. The description text specified how the actual number of fields in enumerated. A value of "None" indicates that no maximum number is specified. |
| Num Elem. | Number of elements within each field |
| Elem. Size (Bytes) | Size of each element |
| Base FIELD_ID (Hex) | Base FIELD_ID |
| VMM Access Prod. | Host VMM access for production TDs |
| VMM Access Debug | Host VMM access for debuggable TDs |
| Guest Access | Guest TD access |
| VMM Wr Mask Prod. | Host VMM write mask for production TDs A bit value of 1 indicates that the corresponding field bit is writable. However, actual writability may be subject to supported features and configuration, as described in other places in this document. |
| VMM Wr Mask Debug | Host VMM write mask for debuggable TDs A bit value of 1 indicates that the corresponding field bit is writable. However, actual writability may be subject to supported features and configuration, as described in other places in this document. |
| Guest Wr Mask | Guest TD write mask A bit value of 1 indicates that the corresponding field bit is writable. However, actual writability may be subject to supported features and configuration, as described in other places in this document. |

4.2.2.2. TDVPS.PIDPT_INDEX[4]

Unreleased Feature: This section is related to Enhanced Interrupt Virtualization, a feature which has not been released yet at the time of writing of this document. Details provided below serve as a preview and are subject to change.

PIDPT_INDEX is an array of 4 per-VM indices to the PIDPT entry.

- Legal values must be smaller than TDCS.PIDPT_NUM_ENTRIES.
- Index should be unique, but the TDX Module does not enforce that.
- Default value is -1, indicating a null PIDPT_INDEX; IPI to this VCPU is not enabled.

Description

L1 calls TDG.VP.WR(PIDPT_INDEX[VM]) to notify the TDX Module about the PIDPT_INDEX of the current VPCU for the requested VM. The meaning of PIDPT_INDEX is up to L1:

- For L1, it is typically either be the VCPU_INDEX (which is the default) or the virtual x2APIC_ID value
- For L2, it is typically the virtual x2APIC_ID value, same as set by L1 in the L2's Virtual APIC page

The TDX Module checks that the value is < TDCS.PIDPT_NUM_ENTRIES[VM], but doesn't check for duplicates or correctness. Note that PIDPT_NUM_ENTRIES is 0 if PIDPT is not configured.

Details

1. On import, if TDVPS.PIDPT_NUM_ENTIRES[VM]
 - 1.1. This is the import case where IPI was not enabled for this VM, so no PIDPT entries were configured. PIDPT_INDEX must be -1 (null PIDPT_INDEX).

If passed:

2. Check that value is < TDVPS.PIDPT_NUM_ENTIRES[VM]. If failed, abort with a TDX_METADATA_FIELD_VALUE_NOT_VALID error.
3. Set the TDVPS.SEC_PID[VM].NV to the value of TDCS.MAIN_NV[VM].
4. Atomically update the PIDPT entry to point to the Secure PID.

4.2.2.3. TDVPS.VCPU_STATE_DETAILS

VCPU_STATE_DETAILS is a 64-bit field that is composed of 4-bit sets. Within each set, state information is provided for L1 (VM 0) and L2 (VMs 1 through 3).

Table 4.10: TDVPS.VCPU_STATE_DETAILS

| Bits | Name | Description | Details | TDX_FEATURES0 Enumeration Bit | TDH.VP.ENTER IMM_RESUME_HINT Output Flag |
|-------|-----------------------|--|--|---|--|
| 3:0 | VINTR_PENDING[3:0] | Virtual interrupt was pending at VM exit time | VMCS.RVI[7:4] > VAPIC.VPPR[7:4] | ENHANCED_INTR_STATE (bit 40). Bit 0 (for L1) is always available | Configurable by the host VMM using TDCS.VM_CTL.S. HINT_ON_VINTR_PENDING |
| 7:4 | VINTR_IN_SERVICE[3:0] | Virtual interrupt was in service at VM exit time | VMCS.SVI > 30 | ENHANCED_INTR_STATE (bit 40) | Configurable by the host VMM using TDCS.VM_CTL.S. HINT_ON_VINTR_IN_SERVICE |
| 11:8 | VNMI_PENDING[3:0] | Virtual NMI is pending | Indicates that TDVPS.PEND_NMI, set by the host VMM, is not 0 | ENHANCED_INTR_STATE (bit 40). Only bit 8 (for L1) is available. Bits 11:9 are reserved (0). | Configurable by the host VMM using TDCS.VM_CTL.S. HINT_ON_VNMI_PENDING |
| 15:12 | ON_SET[3:0] | Outstanding interrupt notification is set | Secure PID's ON bit is set. See below for details. | ENHANCED_INTR_VIRTUALIZATION (Bit 45) | Enabled if enhanced interrupt virtualization has been configured (per VM) by TDH.INTR.CONFIG |
| Other | RESERVED | Set to 0 | | | |

10

4.2.2.3.1. ON_SET Details

The state of the ON bit of Regular and/or Shared PIDs in shared memory is not provided; the host VMM can access such PIDs directly.

Note that in some cases the TDX Module processes and merges the PID in shared memory into the Secure PID held in TDVPS – even if Secure PID is not enabled. In such cases, the ON bit of such PIDs is cleared; however, Secure PID's ON bit is set and indicated by the VCPU_STATE_DETAILS.ON_SET.

15

4.2.3. TD (L1) VMCS and L2 VMCS

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

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

20

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

4.2.3.1. Common to TD and L2 VMCS

4.2.3.1.1. TD and L2 VMCS Description Files

Most VMCS fields information is provided in separate files:

- **td_vmcs.pdf** and **l2_vmcs.pdf** provide human-readable VMCS tables.

- **td_vmcs.csv** and **l2_vmcs.csv** provide the same tables in a Comma-Separated Values (CSV) format. These files can be imported, if required, to tools such as MS Excel for further processing.

The tables below describe the information provided by those files.

In addition to the above files, **td_vmcs.json** and **l2_vmcs.json**, JSON format files with the same information are provided.

5 **Table 4.11: TD and L2 VMCS Table Description**

| Column | Description |
|----------------------------|---|
| Sub-Class | Field class |
| Field | Field name |
| Description | Field description |
| Init Value | Initial value |
| Field Size (Bytes) | Field size |
| Num Fields | Number of fields in an array |
| Base FIELD_ID (Hex) | Base FIELD_ID |
| VMM Access Prod. | Host VMM access for production TDs |
| VMM Access Debug | Host VMM access for debuggable TDs |
| L1 Access | For L2 VMCS, this field describes L1 access |
| VMM Wr Mask Prod. | Host VMM write mask for production TDs A bit value of 1 indicates that the corresponding field bit is writable. However, actual writability may be subject to supported features and configuration, as described in other places in this document. |
| VMM Wr Mask Debug | Host VMM write mask for debuggable TDs A bit value of 1 indicates that the corresponding field bit is writable. However, actual writability may be subject to supported features and configuration, as described in other places in this document. |
| L1 Wr Mask | L1 write mask (L2 VMCS only) A bit value of 1 indicates that the corresponding field bit is writable. However, actual writability may be subject to supported features and configuration, as described in other places in this document. |

4.2.3.2. TD (L1) VMCS

4.2.3.2.1. TD VMCS CR4 Guest/Host Mask

Table 4.12: TD VMCS CR4 Guest/Host Mask

| Bit | Name | Value | Description |
|-----|-------|-----------------------|---|
| 6 | MCE | 1 | Owned by the Intel TDX Module |
| 13 | VMXE | 1 | Owned by the Intel TDX Module |
| 14 | SMXE | 1 | Owned by the Intel TDX Module |
| 19 | KL | ~TDCS.ATTRIBUTES.KL | Intercept writes to CR4 if KeyLocker is not enabled |
| 22 | PKE | ~TDCS.XFAM[9] | Intercept writes to CR4 if PK is not enabled |
| 24 | PKS | ~TDCS.ATTRIBUTES.PKS | Intercept writes to CR4 if PKS is not enabled |
| 25 | UINTR | ~TDCS.XFAM[14] | Intercept writes to CR4 if ULI is not enabled |
| 27 | LASS | ~TDCS.ATTRIBUTES.LASS | Intercept writes to CR4 if LASS is not enabled |

| Bit | Name | Value | Description |
|---|------|--------------------------------|---|
| 32 | FRED | ~virt. CPUID(0x7,1).EAX[17] | Intercept writes to CR4 if FRED is not enabled Applicable to TDX Modules which support FRED. |
| Any bit set to 1 in IA32_VMX_CR4_FIXED0 (i.e., a bit whose value must be 1) | | 1 | Intercept writes of illegal values to CR4 |
| Any bit set to 0 in IA32_VMX_CR4_FIXED1 (i.e., a bit whose value must be 0) | | 1 | Intercept writes of illegal values to CR4 |
| Bits known to the Intel TDX Module as reserved (bits 63:33, 31:29, 26 and bit 15) | | 1 | Intercept writes of illegal values to CR4 |
| Other bits | | 0 | |

5. Interface Functions

5.1. How to Read the Interface Function Definitions

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

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

Table 5.1: How to Read the Operands Information Tables

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Alignment Check | Concurrency Restrictions | | |
|--|---|--|--|---|--|-----------------------------------|-----------------------------------|---|--------------|--------------|
| | | | | | | | | Resource | Contain. 2MB | Contain. 1GB |
| The operand may be specified explicitly or may be implicit | Register used as a pointer to the operand | How the operand is referenced: HPA, GPA, GPA and level or index | Resource (memory or CPU internal) for this operand | Data type of the resource, as defined in Chapter 3 or Chapter 4 | Type of memory or resource access: R, RW, or Ref | Shared, Private, Opaque or Hidden | Required alignment of the operand | Concurrency restrictions are described in the [TDX Module Base Spec]. For explicit memory accesses using HPA, there are additional concurrency restrictions on the 1GB and 2MB blocks that contain the accessed HPA. For other types of accesses, only the operand concurrency is applicable. Shared(h) and Exclusive(h) indicate shared access with host-side priority. Sh./Ex.(h) indicates either shared or exclusive access with host-side priority, depending on the platform type <ul style="list-style-type: none"> • On platforms which do not use ACT, access is shared. • On platforms which use ACT, access is exclusive. Shared(i) and Exclusive(i) indicate that the resource is implicitly restricted. | | |

5.2. Reserved Leaf Numbers

- 10 The following SEAMCALL and TDCALL leaf number ranges are reserved and will never be used by production TDX Modules:
 - 0x00F0 - 0x00FF:** Range reserved for debug builds of the TDX Module
 - 0xF000 - 0xFFFF:** Range reserved for debug builds of the TDX Module
 - 0xE000 - 0xEFFF:** Range reserved for software use, will never be used by the TDX Module
 - Other:** Ranges available for SEAMCALL and TDCALL leaf assignments

5.3. Common Algorithms Used by Multiple Interface Functions

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

5.3.1. VCPU Association with an LP

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

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

5.4. Host-Side (SEAMCALL) Interface Functions

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

5.4.1. SEAMCALL Instruction (Common)

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

Table 5.2: SEAMCALL Input Operands Definition

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

Table 5.3: SEAMCALL Output Operands Definition

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

The table below summarizes the host-side interface functions leaf numbers. Note the following:

- For support of specific interface functions by specific TDX Module releases, see the detailed description of each function.
- TDX Connect interface function leaf numbers are defined in the [TDX Connect ABI Spec].

Table 5.4: SEAMCALL Instruction Leaf Numbers Definition

| Leaf # | Interface Function Name | Description |
|--------|-------------------------|--------------------------------------|
| 0 | TDH.VP.ENTER | Enter TDX non-root operation |
| 1 | TDH.MNG.ADDCX | Add a control structure page to a TD |

| Leaf # | Interface Function Name | Description |
|--------|-------------------------|---|
| 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 |
| 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 |
| 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 WBINVD domain |
| 41 | TDH.PHYMEM.PAGE.WBINVD | Write back and invalidate all cache lines associated with the specified memory page and HKID |
| 42 | TDH.SYS.RDM | Read multiple TDX Module global-scope metadata fields |
| 43 | TDH.VP.WR | Write VCPU metadata |
| 45 | TDH.SYS.CONFIG | Globally configure the Intel TDX Module |
| 46 | TDH.MNG.RDM | Read multiple TD-scope metadata fields |
| 47 | TDH.MNG.WRM | Write multiple TD-scope metadata fields |
| 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 |
| 50 | TDH.VP.RDM | Read multiple VCPU-scope metadata fields |

| Leaf # | Interface Function Name | Description |
|--------|----------------------------|--|
| 51 | TDH.VP.WRM | Write multiple VCPU-scope metadata fields |
| 52 | TDH.SYS.SHUTDOWN | Shutdown the Intel TDX Module and prepare handoff data |
| 53 | TDH.SYS.UPDATE | Populate Intel TDX Module state from handoff data |
| 54 | TDH.SYS.S4_END | End S4 resume session |
| 58 | TDH.PHYMEM.PAMT.ADD | Add a pair of PAMT pages |
| 59 | TDH.PHYMEM.PAMT.REMOVE | Remove a pair of PAMT pages |
| 60 | TDH.EXT.INIT | Initialize the TDX Module extensions |
| 61 | TDH.EXT.MEM.ADD | Add up to 512 4KB memory pages to the TDX Module's memory pool |
| 64 | TDH.EXPORT.ABORT | Abort an export session |
| 65 | TDH.EXPORT.BLOCKW | Block a TD private page for writing |
| 66 | TDH.EXPORT.RESTORE | Restore a list of TD private 4KB pages' Secure EPT entry states after an export abort |
| 68 | TDH.EXPORT.MEM | Export a list of TD private pages contents and/or cancellation requests |
| 70 | TDH.EXPORT.PAUSE | Pause the exported TD |
| 71 | TDH.EXPORT.TRACK | End the current in-order export phase epoch and either start a new epoch or start the out-of-order export phase |
| 72 | TDH.EXPORT.STATE.IMMUTABLE | Start an export session and export the TD's immutable state |
| 73 | TDH.EXPORT.STATE.TD | Export the TD's mutable state |
| 74 | TDH.EXPORT.STATE.VP | Export a VCPU mutable state |
| 75 | TDH.EXPORT.UNBLOCKW | Unblock a page that has been blocked for writing |
| 80 | TDH.IMPORT.ABORT | Abort an import session |
| 81 | TDH.IMPORT.END | End an import session |
| 82 | TDH.IMPORT.COMMIT | Commit the import session and allow the imported TD to run |
| 83 | TDH.IMPORT.MEM | Import a list of TD private pages contents and/or cancellation requests based on a migration bundle in shared memory |
| 84 | TDH.IMPORT.TRACK | End the current in-order import phase epoch and either start a new epoch or start the out-of-order import phase |
| 85 | TDH.IMPORT.STATE.IMMUTABLE | Start an import session and import the TD's immutable state |
| 86 | TDH.IMPORT.STATE.TD | Import the TD's mutable state |
| 87 | TDH.IMPORT.STATE.VP | Import a VCPU mutable state |
| 96 | TDH.MIG.STREAM.CREATE | Create a migration stream |
| 163 | TDH.MEM.SHARED.SEPT.WR | Map/un-map shared EPT root table entries used to support trusted DMA access to TD shared GPA space. |

5.4.1.1. Instruction Description

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

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

- CPU mode is incorrect
- TDX Module has not been loaded
- 10 • TDX Module has been disabled

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

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

- 15 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.

5.4.1.2. Completion Status Codes

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

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

5

5.4.2. TDH.EXPORT.ABORT Leaf

Unreleased Feature: Some of the text in this section is related to Non-Blocking Export, a feature which has not been released yet at the time of writing of this document. Details related to that feature serve as a preview and are subject to change.

- 5 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.

5.4.2.1. Input Operands

Table 5.6: TDH.EXPORT.ABORT Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 64 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | HPA of the source TD's TDR page (HKID bits must be 0) | | |
| R8 | If an abort token is available, R8 provides the HPA and size of memory of an MBMD structure in memory, as described below. Otherwise, R8's value must be 0. | | |
| | Bits | Name | Description |
| | 51:0 | HPA | Bits 51:0 of the host physical address (including HKID bits) HPA must be aligned on 128 bytes. |
| | 63:52 | Size | Size of the memory buffer containing MBMD, in bytes The buffer size must be at least the maximum MBMD size, which is 128 bytes. See 3.13.1 for details. |
| R10 | Migration stream index: | | |
| | Bits | Name | Description |
| | 15:0 | MIGS_INDEX | Migration stream index – must be 0 |
| | 63:16 | RESERVED | Reserved: must be 0 |

10 5.4.2.2. Output Operands

Table 5.7: TDH.EXPORT.ABORT Output Operands Definition

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

5.4.2.3. Leaf Function Description

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

5.4.2.3.1. Overview

5 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.

If the TDX module is configured for non-blocking export, TDH.EXPORT.ABORT resets the internal state held by the TDX module for comprehensive memory scans of the specified TD’s GPA address space.

5.4.2.3.2. Enumeration

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

5.4.2.4. Operands Information

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

15 **Table 5.8: TDH.EXPORT.ABORT Operands Information Definition**

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

5.4.2.5. Completion Status Codes

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

| Completion Status Code | Description |
|------------------------------|-------------|
| TDX_INCORRECT_MBMD_MAC | |
| TDX_INVALID_MBMD | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |

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

5.4.3. TDH.EXPORT.BLOCKW Leaf

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

5.4.3.1. Input Operands

Table 5.10: TDH.EXPORT.BLOCKW Input Operands Definition

| Operand | Name | Description | | | | | | | | | | | | |
|---------|------------------|---|-------------|---|-------------|------|-------------|---|-------|----------------|--|-------|----------|-----------|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | | | | | | | | | | | |
| | | <table border="1"> <thead> <tr> <th>Bits</th> <th>Field</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>15:0</td> <td>Leaf Number</td> <td>Selects the SEAMCALL interface function: 65</td> </tr> <tr> <td>23:16</td> <td>Version Number</td> <td>Selects the SEAMCALL interface function version Must be 0</td> </tr> <tr> <td>63:24</td> <td>Reserved</td> <td>Must be 0</td> </tr> </tbody> </table> | Bits | Field | Description | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 65 | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 | 63:24 | Reserved | Must be 0 |
| | | Bits | Field | Description | | | | | | | | | | |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 65 | | | | | | | | | | |
| 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 | | | | | | | | | | | | |
| 63:24 | Reserved | Must be 0 | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |
| RCX | GPA_LIST_INFO | GPA_LIST_INFO: HPA of a GPA list page in shared memory, and first and last entries to process, as defined in 3.13.2 FORMAT must be GPA_ONLY. | | | | | | | | | | | | |
| RDX | TDR | HPA of the source TD's TDR page (HKID bits must be 0) | | | | | | | | | | | | |

5

5.4.3.2. Output Operands

Table 5.11: TDH.EXPORT.BLOCKW Output Operands Definition

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

5.4.3.3. Leaf Function Description

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

5.4.3.3.1. Overview

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

5.4.3.3.2. Enumeration

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

5.4.3.3.3. Export Mode

- 5 TDH.EXPORT.BLOCKW is only available if the TDX module is configured for write-blocking based export (the default). If not available, calling TDH.EXPORT.BLOCKW returns a TDX_OPERAND_INVALID(RAX) status.

5.4.3.3.4. List Entry Error

- 10 If a page can't be blocked for writing, TDH.EXPORT.BLOCKW marks its GPA list entry as unsuccessful. List processing is not aborted, it continues to the next entry, if applicable. The return status in RAX indicates the number of such cases encountered during operation in the lower 32 bits.

5.4.3.3.5. Interruptibility

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

15 5.4.3.4. Operands Information

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

Table 5.12: TDH.EXPORT.BLOCKW Memory Operands Information

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

5.4.3.5. Completion Status Codes

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

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

5.4.4. TDH.EXPORT.MEM Leaf

Unreleased Feature: Some of the text in this section is related to Non-Blocking Export, a feature which has not been released yet at the time of writing of this document. Details related to that feature serve as a preview and are subject to change.

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

5.4.4.1. Input Operands

Table 5.14: TDH.EXPORT.MEM Input Operands Definition

| Operand | Name | Description | | |
|---------|------------------|--|--|--|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 68 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 | |
| RCX | GPA_LIST_INFO | HPA of a GPA list page in shared memory, and first and last entries to process, as defined in 3.13.2 On a new invocation, FIRST_ENTRY must be 0. On a resumed invocation, FIRST_ENTRY must be the index of the next GPA list entry to export. | | |
| RDX | TDR | HPA of the source TD's TDR page (HKID bits must be 0) | | |
| R8 | MBMD | HPA and size of memory of a memory buffer to use for MBMD: | | |
| | | Bits | Name | Description |
| | | 51:0 | HPA | Bits 51:0 of the host physical address (including HKID bits) HPA must be aligned on 128 bytes. |
| | 63:52 | Size | Size of the memory buffer containing MBMD, in bytes The buffer size must be at least the maximum MBMD size, which is 128 bytes. See 3.13.1 for details. | |
| R9 | MIG_BUFF_LIST | HPA (including HKID bits) of a migration buffer list in shared memory, corresponding to the GPA list pointed by RCX – see 3.13.3. | | |
| R10 | MIG_STREAM | Migration stream and resume flag: | | |
| | | Bits | Name | Description |
| | | 15:0 | MIGS_INDEX | Migration stream index If N migration streams have been created by TDH.MIG.STREAM.CREATE, then MIGS_INDEX must be lower than N-1. |
| | | 62:16 | RESERVED | Reserved: must be 0 |
| | 63 | RESUME | 0: This is a new invocation 1: This is resumption of a previously interrupted operation | |

| Operand | Name | Description |
|---------|-------------|---|
| R11 | MAC_LIST_0 | HPA (including HKID bits) of a MAC list in shared memory, corresponding to the first 256 entries of the GPA list pointed by RCX – see 3.13.3. If GPA_LIST_INFO.FIRST_ENTRY >= 256, then MAC_LIST_0 is ignored. |
| R12 | MAC_LIST_1 | HPA (including HKID bits) of a MAC list in shared memory, corresponding to the last 256 entries of the GPA list pointed by RCX – see 3.13.3. If GPA_LIST_INFO.LAST_ENTRY < 256, then MAC_LIST_1 is ignored. |
| R14 | ATTRIB_LIST | If GPA_LIST_INFO.FORMAT is GPA_AND_L2_ATTR, then R14 contains the HPA (including HKID bits) of a page attributes list in shared memory – see 3.13.4. Else, R14 is ignored. An ATTRIB_LIST is mandatory for exporting partitioned TDs (which contain one or more L2 VMs). Enumeration: Support of this feature is enumerated by TDX_FEATURES0.PARTITIONED_TD_MIGRATION, readable by TDH.SYS.RD* (see 3.3.3.1). |

5.4.4.2. Output Operands

Table 5.15: TDH.EXPORT.MEM Output Operands Definition

| Operand | Name | Description |
|----------------------------|---------------|--|
| RAX | STATUS | SEAMCALL instruction return code, see 5.4.1 |
| RCX | GPA_LIST_INFO | Same as the input value, except that FIRST_ENTRY is updated to the index of the next entry to be processed. If all entries have been processed, FIRST_ENTRY is updated to (LAST_ENTRY + 1) Modulo 512. |
| RDX | NUM_EXPORTED | If TDH.EXPORT.MEM is successful, RDX returns the number of exported 4KB migration buffers, including: <ul style="list-style-type: none"> The GPA list page Attributes list page (if GPA_LIST_INFO.FORMAT is GPA_AND_L2_ATTR) One or two MAC pages (depending on GPA_LIST_INFO.FIRST_ENTRY and GPA_LIST_INFO.LAST_ENTRY) Up to 512 encrypted memory pages If TDH.EXPORT.MEM is not successful, RDX is unmodified. |
| AVX, AVX2 and AVX512 state | | May be reset to the architectural RESET state |
| Other | | Unmodified |

5 5.4.4.3. Leaf Function Description

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

5.4.4.3.1. Overview

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

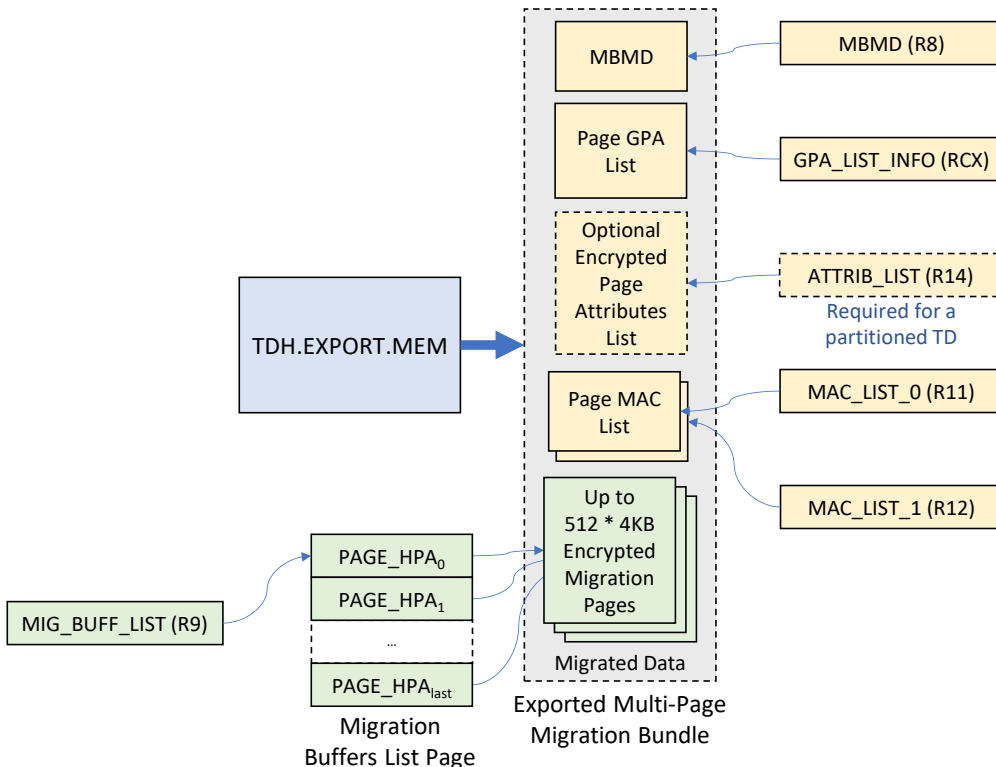


Figure 5.1: TDH.IMPORT.MEM Inputs and Outputs

5.4.4.3.2. Enumeration

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

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

Interruption due to yielding to concurrent functions that try to acquire an exclusive lock on the SEPT trees (see below) is supported if TDX_FEATURES0.NON_BLOCKING_EXPORT (bit 41) is set to 1.

5.4.4.3.3. GPA List

A GPA list is provided as an input and is updated by TDH.EXPORT.MEM. The GPA list format is described in 3.13.2. Only 4KB pages are supported.

For write-blocking based export, the requested operation for each page in the list may be one of the following:

MIGRATE: Export the page. TDH.EXPORT.MEM decides, based on the SEPT entry state, whether the operation is an export or a re-export of a previously exported page. Note that the MIGRATE operation request is represented by two possible values.

CANCEL: Cancel a previous page export.

NOP: No operation.

For non-blocking export, the requested operation for each page in the list may be one of the following:

EXPORT: Export the page. TDH.EXPORT.MEM decides, based on the SEPT entry state, whether the operation is export, re-export of a previously exported page, or cancellation of a previous export.

NOP: No operation. Note that the NOP operation request is represented by three possible values.

TDH.EXPORT.MEM updates the GPA list with the actual OPERATION code (MIGRATE, REMIGRATE, CANCEL or NOP) and STATUS; the host VMM is expected to send the GPA list as part of the migration bundle, to be imported on the destination platform by TDH.IMPORT.MEM.

5.4.4.3.4. S4 Hibernation

5 If TDH.EXPORT.MEM is called as part of an S4 hibernation, it only supports write-blocking based export (though no write blocking is actually required) and the out-of-order export phase. As a result, the GPA list may not contain a CANCEL operation. In addition, blocking and TLB tracking is not required.

5.4.4.3.5. Migration Buffers List

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

5.4.4.3.6. Write-Blocking and TLB Tracking (Write-Blocking based Export)

The following applies if the TDX module is configured for write-blocking based export:

15 If the TD may be running, the exported pages must be blocked for writing by TDH.EXPORT.BLOCKW and TLB tracked (TDH.MEM.TRACK followed by IPI to all the LPs running the TD VCPUs) to be exported. Unlike memory management operations such as TDH.MEM.PAGE.REMOVE, the TLB tracking is not page-specific; it should be done after the last TDH.EXPORT.BLOCKW of any page has been called for the current export round.

Else (e.g., the TD has been paused for export), no blocking and tracking is required.

5.4.4.3.7. Scanning and TLB Tracking (Non-Blocking Export)

The following applies if the TDX module is configured for non-blocking export:

20 If the TD may be running, the exported pages must have been scanned by TDH.MEM.SCAN(DSCAN) and TLB tracked to be exported. Unlike write-blocking based export, TLB tracking is page-specific, allowing TDH.MEM.SCAN(DSCAN) to run concurrently with TDH.EXPORT.MEM.

Else (e.g., the TD has been paused for export), no tracking is required.

5.4.4.3.8. Page-Specific Errors

25 If a page can't be exported for a reason that is specific to that page, TDH.EXPORT.MEM marks its GPA list entry as unsuccessful, but does not abort. It continues to the next entry, if applicable. Eventually, if TDH.EXPORT.MEM completes successfully, the TDX_SUCCESS return status indicates the number of such cases encountered during operation in RAX[31:0].

5.4.4.3.9. Concurrency

30 TDH.EXPORT.MEM may be called concurrently on multiple LPs (each specifying a separate migration stream) and may run concurrently with other functions (e.g., TDH.MEM.SCAN.*).

SEPT Tree Concurrency

35 TDH.EXPORT.MEM acquires a shared lock on the SEPT trees, to prevent blocking or changes to the tree structure (e.g., by TDH.MEM.RANGE.BLOCK or TDH.MEM.SEPT.REMOVE) while it runs. If supported, TDH.EXPORT.MEM may prevent starvation of concurrent memory management functions by detecting that they failed to acquire an exclusive lock on the SEPT trees. In this case, TDH.EXPORT.MEM yields and returns with a TDX_INTERRUPTED_BUSY status in RAX. The host VMM is expected to resume TDH.EXPORT.MEM. See the discussion on interruptibility below.

SEPT Entry Concurrency

40 Failure to acquire a lock on an SEPT entry is handled as a page-specific error, as described above. TDH.EXPORT.MEM skips the busy page.

5.4.4.3.10. Interruption and Resumption

TDH.EXPORT.MEM is interruptible. An interruption occurs in the following cases:

- If a pending interrupt is detected during operation, TDH.EXPORT.MEM returns with a TDX_INTERRUPTED_RESUMABLE status in RAX.

- If supported and if TDH.EXPORT.MEM detects that a concurrent function (such as TDH.MEM.PAGE.PROMOTE) has failed to acquire an exclusive lock on the SEPT trees, it may yield and return with a TDX_INTERRUPTED_BUSY status in RAX.

5 If a pending interrupt is detected during operation, TDH.EXPORT.MEM returns with a TDX_INTERRUPTED_RESUMABLE status in RAX. RCX is updated with the next list entry index to process.

The host VMM should re-invoke TDH.EXPORT.MEM after handling the interrupt, keeping the same inputs (and updated value in RCX) except setting R10.RESUME to 1. Failing to resume TDH.EXPORT.MEM and to export the generated migration bundle when it completes successfully will result in the migration protocol going out of sync; this will be detected by the destination side, resulting in an import failure.

10 The host VMM should not transmit the generated migration bundle to the destination side until TDH.EXPORT.MEM is completed successfully. Doing so will result in the migration protocol going out of sync; this will be detected by the destination side, resulting in an import failure.

5.4.4.4. Operands Information

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

Table 5.16: TDH.EXPORT.MEM Operands Information Definition

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

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

5.4.4.5. Completion Status Codes

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

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

5.4.5. TDH.EXPORT.PAUSE Leaf

Unreleased Feature: Some of the text in this section is related to Non-Blocking Export, a feature which has not been released yet at the time of writing of this document. Details related to that feature serve as a preview and are subject to change.

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

5.4.5.1. Input Operands

Table 5.18: TDH.EXPORT.PAUSE Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 70 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | HPA of Source TD TDR page (HKID bits must be 0) | | |

5.4.5.2. Output Operands

Table 5.19: TDH.EXPORT.PAUSE Output Operands Definitions

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

5.4.5.3. Leaf Function Description

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

5.4.5.3.1. Overview

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

- All TD VCPUs have stopped executing.
- No TD-specific SEAMCALL is running.

5.4.5.3.2. Enumeration

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

5.4.5.3.3. Preconditions

TDH.EXPORT.PAUSE checks the following preconditions:

- 25
- An export session must be in progress and must be in the LIVE_EXPORT state.
 - If the TDX module is configured for non-blocking export, no TD private page may be blocked.
 - If the TD is configured for TDX Connect, no TDIs may be bound.

5.4.5.4. Operands Information

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

Table 5.20: TDH.EXPORT.PAUSE Operands Information Definition

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

5

5.4.5.5. Completion Status Codes

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

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_BLOCKED_PAGES_EXIST | There are memory pages that have been blocked by TDH.MEM.RANGE.BLOCK. This is not permitted if the TDX module is configured for non-blocking export. |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | Operation is successful |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |

5.4.6. TDH.EXPORT.RESTORE Leaf

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

5.4.6.1. Input Operands

Table 5.22: TDH.EXPORT.RESTORE Input Operands Definition

| Operand | Name | Description | | |
|---------|------------------|---|----------------|--|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 66 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 | |
| RCX | GPA_LIST_INFO | GPA_LIST_INFO: HPA of a GPA list page in shared memory, and first and last entries to process, as defined in 3.13.2 FORMAT must be GPA_ONLY. | | |
| RDX | TDR | HPA of the source TD's TDR page (HKID bits must be 0) | | |

5

5.4.6.2. Output Operands

Table 5.23: TDH.EXPORT.RESTORE Output Operands Definition

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

5.4.6.3. Leaf Function Description

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

5.4.6.3.1. Overview

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

15 5.4.6.3.2. Enumeration

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

5.4.6.3.3. List Entry Error

If a page’s Secure EPT entry can’t be restored, TDH.EXPORT.RESTORE marks its GPA list entry as unsuccessful. List process is not aborted; it continues to the next entry, if applicable. The return status in RAX indicates the number of such cases encountered during operation in the lower 32 bits.

5 **5.4.6.3.4. Interruptibility**

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

5.4.6.4. Operands Information

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

Table 5.24: TDH.EXPORT.RESTORE Operands Information Definition

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

5.4.6.5. Completion Status Codes

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

| Completion Status Code | Description |
|------------------------------|-------------|
| TDX_INTERRUPTED_RESUMABLE | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | |

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

5.4.7. TDH.EXPORT.STATE.IMMUTABLE Leaf

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

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

5 5.4.7.1. Input Operands

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

| Operand | Name | Description | | |
|---------|------------------|---|--|---|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 72 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 | |
| RCX | TDR | Source TD handle and flags | | |
| | | Bits | Name | Description |
| | | 0 | EXPORT_TYPE | 0: TD Export 1: S4 Hibernation |
| | | 11:1 | Reserved | Must be 0 |
| | | 51:12 | TDR HPA | HPA[51:12] of the source TD's TDR page (HKID bits must be 0) |
| | 63:52 | Reserved | Must be 0 | |
| R8 | MBMD | HPA and size of memory of a memory buffer to use for MBMD: | | |
| | | Bits | Name | Description |
| | | 51:0 | HPA | Bits 51:0 of the host physical address (including HKID bits) HPA must be aligned on 128 bytes. |
| | 63:52 | Size | Size of the memory buffer containing MBMD, in bytes The buffer size must be at least the maximum MBMD size, which is 128 bytes. See 3.13.1 for details. | |
| R9 | PAGE_LIST_INFO | Migration buffers list information – see 3.13.6.1 See enumeration details below. | | |
| R10 | MIG_STREAM | Migration stream and resume flag: | | |
| | | Bits | Name | Description |
| | | 15:0 | MIGS_INDEX | Migration stream index – must be 0 |
| | 62:16 | RESERVED | Reserved: must be 0 | |

| Operand | Name | Description | |
|---------|------|-------------|--|
| | | 63 | RESUME |
| | | | 0: This is a new invocation 1: This is resumption of a previously interrupted operation |

5.4.7.2. Output Operands

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

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

5.4.7.3. Leaf Function Description

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

5.4.7.3.1. Overview

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

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

5.4.7.3.2. Enumeration

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

The required number of 4KB pages for exporting the immutable TD state is enumerated by NUM_IMMUTABLE_STATE_PAGES.

5.4.7.3.3. Interruptibility

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

5.4.7.3.4. Aspects of MAC Calculation Compatibility on TDX Module Update

If TDH.EXPORT.STATE.IMMUTABLE was interrupted, and later the TDX Module was updated before TDH.EXPORT.STATE.IMMUTABLE was resumed, there could be incompatibility issues with the format of the intermediate MRTD MAC context, held as part of the migration stream context (MIGSC), between the original and the new TD module.

On TDH.SYS.UPDATE, the host VMM can configure the TDX Module to detect such incompatibility. If detected, TDH.EXPORT.STATE.IMMUTABLE returns a TDX_INCOMPATIBLE_MBMD_MAC_CONTEXT status. In this case, the host VMM is expected to abort the migration session and may restart it. For details, see the [Base Spec] section titled "Compatibility Aspects of TD-Preserving Update".

If the host VMM did not configure the TDX Module to detect such incompatibility, it will not be detected by TDH.EXPORT.STATE.IMMUTABLE. However, on import, TDH.IMPORT.STATE.IMMUTABLE will detect that the MAC is incorrect and will return a TDX_INCORRECT_MBMD_MAC error.

5.4.7.4. Operands Information

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

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

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

5.4.7.5. Completion Status Codes

10

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

| Completion Status Code | Description |
|--------------------------------------|---|
| TDX_INCOMPATIBLE_MBMD_MAC_CONTEXT | |
| TDX_INTERRUPTED_RESUMABLE | |
| TDX_INVALID_RESUMPTION | |
| TDX_IOMMU_IOTLB_TRACKING_NOT_DONE | Applicable only if TDX Connect is supported |
| TDX_MAX_EXPORTS_EXCEEDED | |
| TDX_METADATA_LIST_OVERFLOW | |
| TDX_MIGRATION_SESSION_KEY_NOT_SET | |
| TDX_MIGRATION_STREAM_STATE_INCORRECT | |
| TDX_MIN_MIGS_NOT_CREATED | |

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

5.4.8. TDH.EXPORT.STATE.TD Leaf

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

5.4.8.1. Input Operands

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

| Operand | Name | Description | | |
|---------|------------------|---|--|---|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 73 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | | 63:24 | Reserved | Must be 0 |
| RCX | TDR | HPA of the source TD's TDR page (HKID bits must be 0) | | |
| R8 | MBMD | HPA and size of memory of a memory buffer to use for MBMD: | | |
| | | Bits | Name | Description |
| | | 51:0 | HPA | Bits 51:0 of the host physical address (including HKID bits) HPA must be aligned on 128 bytes. |
| | 63:52 | Size | Size of the memory buffer containing MBMD, in bytes The buffer size must be at least the maximum MBMD size, which is 128 bytes. See 3.13.1 for details. | |
| R9 | PAGE_LIST_INFO | Migration buffers list information – see 3.13.6.1 See enumeration details below. | | |
| R10 | MIG_STREAM | Migration stream and resume flag: | | |
| | | Bits | Name | Description |
| | | 15:0 | MIGS_INDEX | Migration stream index – must be 0 |
| | | 62:16 | RESERVED | Reserved: must be 0 |
| | 63 | RESUME | 0: This is a new invocation 1: This is resumption of a previously interrupted operation | |

5

5.4.8.2. Output Operands

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

| Operand | Name | Description |
|---------|--------------|---|
| RAX | STATUS | SEAMCALL instruction return code, see 5.4.1 |
| RDX | NUM_EXPORTED | Number of exported 4KB migration buffers |

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

5.4.8.3. Leaf Function Description

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

5 5.4.8.3.1. Overview

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.

10 5.4.8.3.2. Enumeration

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

The required number of 4KB pages for exporting the mutable TD state is enumerated by NUM_TD_STATE_PAGES.

15 5.4.8.3.3. Interruptibility

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

20 5.4.8.3.4. Aspects of MAC Calculation Compatibility on TDX Module Update

If TDH.EXPORT.STATE.TD was interrupted, and later the TDX Module was updated before TDH.EXPORT.STATE.TD was resumed, there could be incompatibility issues with the format of the intermediate MRTD MAC context, held as part of the migration stream context (MIGSC), between the original and the new TD module.

On TDH.SYS.UPDATE, the host VMM can configure the TDX Module to detect such incompatibility. If detected, TDH.EXPORT.STATE.TD returns a TDX_INCOMPATIBLE_MBMD_MAC_CONTEXT status. In this case, the host VMM is expected to abort the migration session and may restart it. For details, see the [Base Spec] section titled "Compatibility Aspects of TD-Preserving Update".

25 If the host VMM did not configure the TDX Module to detect such incompatibility, it will not be detected by TDH.EXPORT.STATE.TD. However, on import, TDH.IMPORT.STATE.TD will detect that the MAC is incorrect and will return a TDX_INCORRECT_MBMD_MAC error.

30 5.4.8.4. Operands Information

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

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

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|---------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | TDR page | TDR | R | Opaque | 4KB | Shared | Shared | Shared |
| Explicit | R8 | HPA | Memory to use for MBMD | MBMD | RW | Shared | 128B | None | None | None |

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

5.4.8.5. Completion Status Codes

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

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

5.4.9. TDH.EXPORT.STATE.VP Leaf

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

5.4.9.1. Input Operands

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

| Operand | Name | Description | | |
|---------|------------------|---|--|--|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 74 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 | |
| RCX | TDVPR | HPA of the source TD VCPU's TDVPR page (HKID bits must be 0) | | |
| R8 | MBMD | HPA and size of memory of a memory buffer to use for MBMD: | | |
| | | Bits | Name | Description |
| | | 51:0 | HPA | Bits 51:0 of the host physical address (including HKID bits) HPA must be aligned on 128 bytes. |
| | 63:52 | Size | Size of the memory buffer containing MBMD, in bytes The buffer size must be at least the maximum MBMD size, which is 128 bytes. See 3.13.1 for details. | |
| R9 | PAGE_LIST_INFO | Migration buffers list information – see 3.13.6.1 See enumeration details below. | | |
| R10 | MIG_STREAM | Migration stream and resume flag: | | |
| | | Bits | Name | Description |
| | | 15:0 | MIGS_INDEX | Migration stream index If N migration streams have been created by TDH.MIG.STREAM.CREATE, then MIGS_INDEX must be lower than N-1. |
| | | 62:16 | RESERVED | Reserved: must be 0 |
| | 63 | RESUME | 0: This is a new invocation 1: This is resumption of a previously interrupted operation | |

5

5.4.9.2. Output Operands

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

| Operand | Name | Description |
|---------|--------|---|
| RAX | STATUS | SEAMCALL instruction return code, see 5.4.1 |

| Operand | Name | Description |
|----------------------------|--------------|---|
| RDX | NUM_EXPORTED | Number of exported 4KB migration buffers |
| AVX, AVX2 and AVX512 state | | May be reset to the architectural RESET state |
| Other | | Unmodified |

5.4.9.3. Leaf Function Description

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

5 5.4.9.3.1. Overview

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.

5.4.9.3.2. Enumeration

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

The required number of 4KB pages for exporting the mutable VCPU state is enumerated by NUM_VP_STATE_PAGES.

5.4.9.3.3. Interruptibility

- 15 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.

5.4.9.3.4. Aspects of MAC Calculation Compatibility on TDX Module Update

- 20 If TDH.EXPORT.STATE.VP was interrupted, and later the TDX Module was updated before TDH.EXPORT.STATE.VP was resumed, there could be incompatibility issues with the format of the intermediate MRTD MAC context, held as part of the migration stream context (MIGSC), between the original and the new TD module.

On TDH.SYS.UPDATE, the host VMM can configure the TDX Module to detect such incompatibility. If detected, TDH.EXPORT.STATE.VP returns a TDX_INCOMPATIBLE_MBMD_MAC_CONTEXT status. In this case, the host VMM is expected to abort the migration session and may restart it. For details, see the [Base Spec] section titled "Compatibility Aspects of TD-Preserving Update".

- 25 If the host VMM did not configure the TDX Module to detect such incompatibility, it will not be detected by TDH.EXPORT.STATE.VP. However, on import, TDH.IMPORT.STATE.VP will detect that the MAC is incorrect and will return a TDX_INCORRECT_MBMD_MAC error.

5.4.9.3.5. VCPU Association

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

5.4.9.4. Operands Information

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

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

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

5.4.9.5. Completion Status Codes

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

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

| Completion Status Code | Description |
|----------------------------|--|
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |
| TDX_VCPU_ALREADY_EXPORTED | |
| TDX_VCPU_STATE_INCORRECT | The VCPU is expected to be in the VCPU_READY state. For details, see the [Base Spec] discussion of VPCU transitions. |

5.4.10. TDH.EXPORT.TRACK Leaf

Unreleased Feature: Some of the text in this section is related to Non-Blocking Export, a feature which has not been released yet at the time of writing of this document. Details related to that feature serve as a preview and are subject to change.

- 5 TDH.EXPORT.TRACK ends the current export epoch and starts a new one. It either starts a new in-order phase epoch or starts the out-of-order phase. In both cases, TDH.EXPORT.TRACK generates an epoch token to be exported to the destination platform.

5.4.10.1. Input Operands

Table 5.38: TDH.EXPORT.TRACK Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 71 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | HPA of the source TD's TDR page (HKID bits must be 0) | | |
| R8 | HPA and size of memory of a memory buffer to use for MBMD: | | |
| | Bits | Name | Description |
| | 51:0 | HPA | Bits 51:0 of the host physical address (including HKID bits) HPA must be aligned on 128 bytes. |
| | 63:52 | Size | Size of the memory buffer containing MBMD, in bytes The buffer size must be at least the maximum MBMD size, which is 128 bytes. See 3.13.1 for details. |
| 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, and a start token should be generated |

10

5.4.10.2. Output Operands

Table 5.39: TDH.EXPORT.TRACK Output Operands Definition

| Operand | Description |
|---------|---|
| RAX | SEAMCALL instruction return code, see 5.4.1 |

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

5.4.10.3. Leaf Function Description

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

5 5.4.10.3.1. Overview

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.

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

5.4.10.3.2. Enumeration

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

15 5.4.10.3.3. Epoch Token Generation

TDH.EXPORT.TRACK generates an epoch token, to be exported. The epoch token, once successfully imported on the destination side, indicates that a new migration epoch has started.

5.4.10.3.4. End of the In-Order Phase and Start Token Generation

20 An R10.IN_ORDER_DONE value of 1 indicates that TDH.EXPORT.TRACK is requested to end the in-order export phase, start the out-of-order phase and generate a start token to be exported. A start token is a private case of an epoch token. Once successfully imported on the destination side, the start token enables the host VMM to commit the migration and start running the TD on the destination.

TDH.EXPORT.TRACK checks the completeness of memory export:

- TDH.EXPORT.TRACK checks that the TD has been paused by TDH.EXPORT.PAUSE.
- 25 • If the TDX module is configured for non-blocking export, TDH.EXPORT.TRACK checks that a comprehensive scan was successfully done using TDH.MEM.SCAN.COMP(DCHECK).
- TDH.EXPORT.TRACK checks that no TD private memory exported so far needs to be re-exported.
- If either the TD is configured for TDX Connect or the TDX module does not support post-copy (this is the case if the TDX module is configured for non-blocking export), TDH.EXPORT.TRACK checks that all the TD private memory has
- 30 been exported.

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

5.4.10.4. Operands Information

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

Table 5.40: TDH.EXPORT.TRACK Operands Information Definition

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

5.4.10.5. Completion Status Codes

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

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

5.4.11. TDH.EXPORT.UNBLOCKW Leaf

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

5.4.11.1. Input Operands

Table 5.42: TDH.EXPORT.UNBLOCKW Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|---|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 75 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | EPT mapping information: | | |
| | Bits | Name | Description |
| | 2:0 | Level | Level of the Secure EPT entry that maps the page to be blocked for writing – see 3.6.1: must be 0 (4KB) |
| | 11:3 | Reserved | Reserved: must be 0 |
| | 51:12 | GPA | Bits 51:12 of the GPA to be unblocked for writing |
| | 63:52 | Reserved | Reserved: must be 0 |
| RDX | Host physical address of the parent TDR page (HKID bits must be 0) | | |

5

5.4.11.2. Output Operands

Table 5.43: TDH.EXPORT.UNBLOCKW Output Operands Definition

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

5.4.11.3. Leaf Function Description

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

5.4.11.3.1. Overview

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

5.4.11.3.2. Enumeration

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

5.4.11.3.3. Export Mode

TDH.EXPORT.UNBLOCKW is only available if the TDX module is configured for write-blocking based export (the default). If not available, calling TDH.EXPORT.UNBLOCKW returns a TDX_OPERAND_INVALID(RAX) status.

5.4.11.4. Operands Information

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

Table 5.44: TDH.EXPORT.UNBLOCKW Operands Information Definition

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

5.4.11.5. Completion Status Codes

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

| Completion Status Code | Description |
|-------------------------------|-------------|
| TDX_EPT_ENTRY_STATE_INCORRECT | |
| TDX_EPT_WALK_FAILED | |

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

5.4.12. TDH.EXT.INIT Leaf

Unreleased Feature: This section is related to TDX Module Extensions, a feature which has not been released yet at the time of writing of this document. Details provided below serve as a preview and are subject to change.

5 TDH.EXT.INIT initializes TDX Module extension.

5.4.12.1. Input Operands

Table 5.46: TDH.EXT.INIT Input Operands Definition

| Operand | Name | Description | | |
|---------|------------------|---|----------------|--|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 60 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 | |

5.4.12.2. Output Operands

10 Table 5.47: TDH.EXT.INIT Output Operands Definition

| Operand | Name | Description |
|---------|--------|---|
| RAX | STATUS | SEAMCALL instruction return code, see 5.4.1 |
| Other | | Unmodified |

5.4.12.3. Leaf Function Description

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

15 5.4.12.3.1. Overview

TDH.EXT.INIT initializes the TDX Module extension.

5.4.12.3.2. Enumeration

Availability of TDH.EXT.INIT is enumerated by TDX_FEATURES0.NRX (bit 39), readable by TDH.SYS.RD*. If not supported, TDH.EXT.INIT returns a TDX_OPERAND_INVALID(RAX) status.

20 5.4.12.3.3. Interruptibility

TDH.EXT.INIT is interruptible. If a pending interrupt is detected during operation, TDH.EXT.INIT returns with a TDX_INTERRUPTED_RESUMABLE status in RAX. The host VMM may re-invoke TDH.EXT.INIT after handling the interrupt, keeping the same inputs.

5.4.12.4. Operands Information

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

Table 5.48: TDH.EXT.INIT Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|-------------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Implicit | N/A | N/A | TDX Module extension state | N/A | RW | Hidden | N/A | Exclusive | None | None |
| Implicit | N/A | N/A | NRX TDs' TDR | TDR | RW | Hidden | N/A | Exclusive(i) | None | None |
| Implicit | N/A | N/A | All other NRX TDs' state | N/A | RW | Hidden | N/A | Exclusive(i) | None | None |

5.4.12.5. Completion Status Codes

Table 5.49: TDH.EXT.INIT Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------------|---|
| TDX_EXT_FATAL_ERROR | |
| TDX_EXT_MEMORY_POOL_REQUIRED | |
| TDX_INCONSISTENT_MSR | IA32_TSC_ADJUST - MSR value is not the same as capture during the TDX Module initialization |
| TDX_INTERRUPTED_RESUMABLE | |
| TDX_INVALID_RESUMPTION | |
| TDX_LIMIT_CPUID_MAXVAL_SET | |
| TDX_OPERAND_BUSY | |
| TDX_SUCCESS | |
| TDX_SYS_BUSY | |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TSC_ROLLBACK | |

5

5.4.13. TDH.EXT.MEM.ADD Leaf

Unreleased Feature: This section is related to TDX Module Extensions, a feature which has not been released yet at the time of writing of this document. Details provided below serve as a preview and are subject to change.

- 5 TDH.EXT.MEM.ADD adds up to 512 4KB memory pages to the TDX Module's memory pool.

5.4.13.1. Input Operands

Table 5.50: TDH.EXT.MEM.ADD Input Operands Definition

| Operand | Name | Description | | |
|---------|------------------|---|----------------|--|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 61 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | | 63:24 | Reserved | Must be 0 |
| RCX | PAGE_LIST_INFO | Reference to a list of memory pages to add, in HPA_LIST_INFO format – See 3.5.3 | | |

5.4.13.2. Output Operands

10

Table 5.51: TDH.EXT.MEM.ADD Output Operands Definition

| Operand | Name | Description |
|---------|----------------|---|
| RAX | STATUS | SEAMCALL instruction return code, see 5.4.1 |
| RCX | PAGE_LIST_INFO | Same as the input value, except that FIRST_ENTRY is updated to the index of the next entry to be processed. If all entries have been processed, FIRST_ENTRY is updated to (LAST_ENTRY + 1) Modulo 512. |
| RDX | RESERVED | Set to 0 |
| Other | | Unmodified |

5.4.13.3. Leaf Function Description

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

15

5.4.13.3.1. Overview

TDH.EXT.MEM.ADD adds up to 512 4KB pages to the TDX Module's memory pool.

5.4.13.3.2. Enumeration

Availability of TDH.EXT.MEM.ADD is enumerated by TDX_FEATURES0.NRX (bit 39), readable by TDH.SYS.RD*. If not supported, calling TDH.EXT.MEM.ADD returns a TDX_OPERAND_INVALID(RAX) status.

The required number of memory pool pages is provided by MEMORY_POOL_REQUIRED_PAGES, readable by TDH.SYS.RD.

5 5.4.13.3.3. Interruptibility

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

5.4.13.4. Operands Information

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

Table 5.52: TDH.EXT.MEM.ADD Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|--|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | List page | HPA list | R | Shared | 4KB | None | None | None |
| Explicit | N/A | HPA | Pool pages | Blob | RW | Private | 4KB | None | None | None |
| Explicit | N/A | HPA | Memory pool buffer pages (via page list) | Blob | RW | Shared | 4KB | Exclusive | Shared | Shared |
| Implicit | N/A | N/A | NRX state | N/A | R | Hidden | N/A | Shared | None | None |

5.4.13.5. Completion Status Codes

15 **Table 5.53: TDH.EXT.MEM.ADD Completion Status Codes (Returned in RAX) Definition**

| Completion Status Code | Description |
|---------------------------------|-------------|
| TDX_INTERRUPTED_RESUMABLE | |
| TDX_INVALID_RESUMPTION | |
| TDX_SUCCESS | |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_EXT_MEMORY_POOL_FULL | |
| TDX_EXT_MEMORY_POOL_NOT_PENDING | |
| TDX_SYS_BUSY | |
| TDX_OPERAND_INVALID | |
| TDX_PAGE_METADATA_INCORRECT | |
| TDX_EXT_MEMORY_POOL_OVERFLOW | |

5.4.14. 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.

5.4.14.1. Input Operands

Table 5.54: TDH.IMPORT.ABORT Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 80 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | HPA of the destination 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) HPA must be aligned on 128 bytes. |
| | 63:52 | Size | Size of the memory buffer containing MBMD, in bytes The buffer size must be at least the maximum MBMD size, which is 128 bytes. See 3.13.1 for details. |
| R10 | Migration stream index – must be 0 | | |

5.4.14.2. Output Operands

Table 5.55: TDH.IMPORT.ABORT Output Operands Definition

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

5.4.14.3. Leaf Function Description

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

5.4.14.3.1. Overview

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.

5 5.4.14.3.2. Enumeration

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

5.4.14.4. Operands Information

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

Table 5.56: TDH.IMPORT.ABORT Operands Information Definition

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

5.4.14.5. Completion Status Codes

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

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

5.4.15. TDH.IMPORT.COMMIT Leaf

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

5.4.15.1. Input Operands

Table 5.58: TDH.IMPORT.COMMIT Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 82 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | HPA of the destination TD's TDR page (HKID bits must be 0) | | |

5

5.4.15.2. Output Operands

Table 5.59: TDH.IMPORT.COMMIT Output Operands Definition

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

5.4.15.3. Leaf Function Description

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

5.4.15.3.1. Overview

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

5.4.15.3.2. Enumeration

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

5.4.15.4. Operands Information

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

Table 5.60: TDH.IMPORT.COMMIT Operands Information Definition

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

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|----------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | 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 |

5.4.15.5. Completion Status Codes

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

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

5.4.16. TDH.IMPORT.END Leaf

End an import session.

5.4.16.1. Input Operands

Table 5.62: TDH.IMPORT.END Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 81 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | HPA of the destination TD's TDR page (HKID bits must be 0) | | |

5

5.4.16.2. Output Operands

Table 5.63: TDH.IMPORT.END Output Operands Definition

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

5.4.16.3. Leaf Function Description

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

5.4.16.3.1. Overview

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

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

5.4.16.3.2. Enumeration

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

5.4.16.4. Operands Information

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

Table 5.64: TDH.IMPORT.END Operands Information Definition

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

5.4.16.5. Completion Status Codes

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

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

5

5.4.17. 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.

5.4.17.1. Input Operands

5

Table 5.66: TDH.IMPORT.MEM Input Operands Definition

| Operand | Name | Description | | |
|---------|------------------|--|--|--|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 83 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 | |
| RCX | GPA_LIST_INFO | HPA of a GPA list page in shared memory, and first and last entries to process, as defined in 3.13.2 On a new invocation, FIRST_ENTRY must be 0. On a resumed invocation, FIRST_ENTRY must be the index of the next GPA list entry to export. | | |
| RDX | TDR | HPA of the destination TD's TDR page (HKID bits must be 0) | | |
| R8 | MBMD | HPA and size of memory of an MBMD structure in memory: | | |
| | | Bits | Name | Description |
| | | 51:0 | HPA | Bits 51:0 of the host physical address (including HKID bits) HPA must be aligned on 128 bytes. |
| | 63:52 | Size | Size of the memory buffer containing MBMD, in bytes The buffer size must be at least the maximum MBMD size, which is 128 bytes. See 3.13.1 for details. | |
| R9 | MIG_BUFF_LIST | HPA (including HKID bits) of a migration buffer list in shared memory, corresponding to the GPA list pointed by RCX – see 3.13.3. No migration buffers are required for PENDING pages and for migration cancellation requests. The list entries for such pages are skipped. | | |
| R10 | MIG_STREAM | Migration stream and resume flag: | | |
| | | Bits | Name | Description |
| | | 15:0 | MIGS_INDEX | Migration stream index If N migration streams have been created by TDH.MIG.STREAM.CREATE, then MIGS_INDEX must be lower than N-1. |
| | | 62:16 | RESERVED | Reserved: must be 0 |
| | 63 | RESUME | 0: This is a new invocation 1: This is resumption of a previously interrupted operation | |

| Operand | Name | Description |
|---------|-------------|--|
| R11 | MAC_LIST_0 | HPA (including HKID bits) of a MAC list in shared memory, corresponding to the first 256 entries of the GPA list pointed by RCX – see 3.13.3. If GPA_LIST_INFO.FIRST_ENTRY >= 256, then MAC_LIST_0 is ignored. |
| R12 | MAC_LIST_1 | HPA (including HKID bits) of a MAC list in shared memory, corresponding to the last 256 entries of the GPA list pointed by RCX – see 3.13.3. If GPA_LIST_INFO.LAST_ENTRY < 256, then MAC_LIST_1 is ignored. |
| R13 | PAGE_LIST | HPA (including HKID bits) of a destination page list in shared memory. See the description below for details. |
| R14 | ATTRIB_LIST | If GPA_LIST_INFO.FORMAT is GPA_AND_L2_ATTR, then R14 contains the HPA (including HKID bits) of a page L2 attributes list in shared memory – see 3.13.4. Else, R14 is ignored. An ATTRIB_LIST is mandatory for importing partitioned TDs (which contain one or more L2 VMs). Enumeration: Support of this feature is enumerated by TDX_FEATURES0.PARTITIONED_TD_MIGRATION, readable by TDH.SYS.RD* (see 3.3.3.1). |

5.4.17.2. Output Operands

Table 5.67: TDH.IMPORT.MEM Output Operands Definition

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

5 5.4.17.3. Leaf Function Description

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

5.4.17.3.1. Overview

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

For each page in the migration bundle's GPA list, the requested operation may either be to import the page, to re-import a newer version of the page (after a previous import) or to cancel a previous page import. It is also possible to skip entries in the list by requesting no operation for specific entries. The GPA list format is described in 3.13.2.

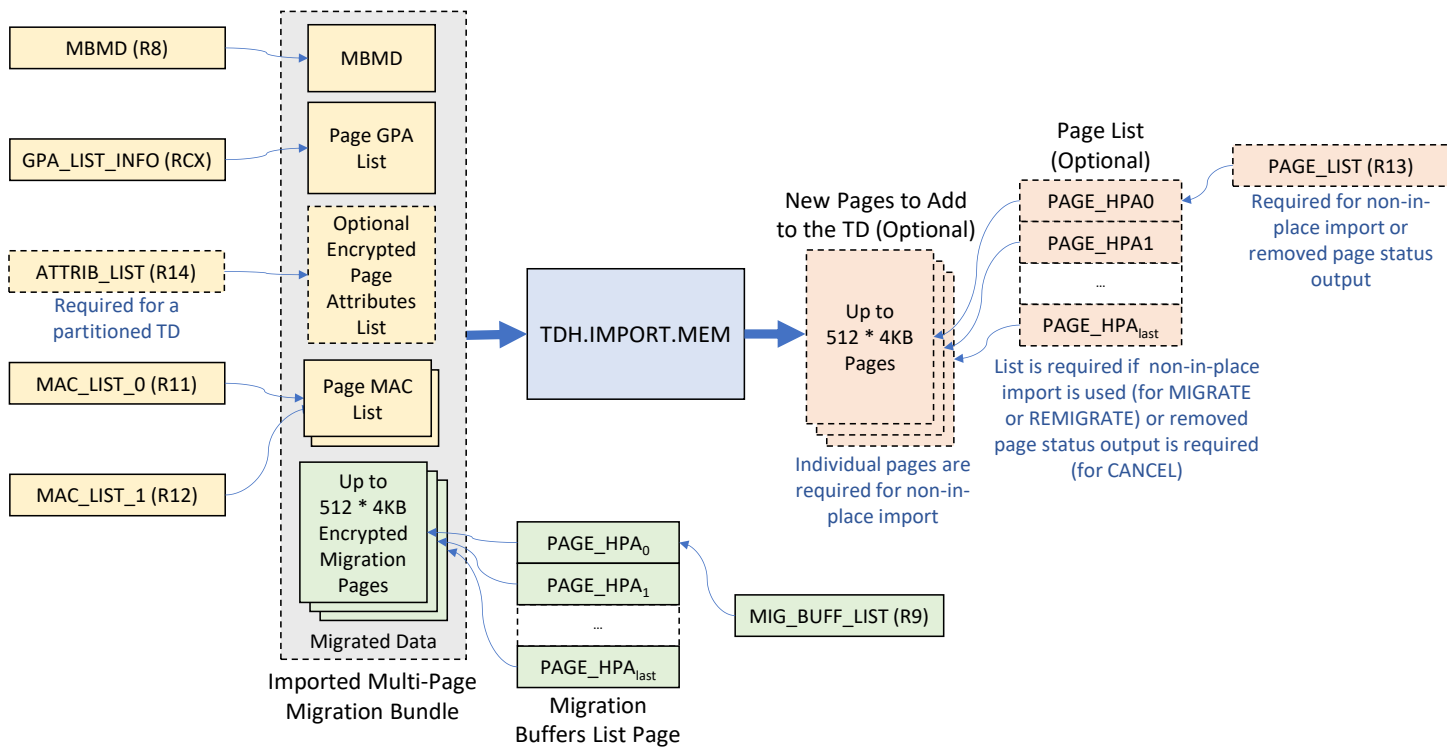


Figure 5.2: TDH.IMPORT.MEM Inputs and Outputs

5.4.17.3.2. Enumeration

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

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

TDX_FEATURES0.IMPORT_PAGE_STATUS (bit 44) enumerates update of page list entries to indicate status (see below).

5.4.17.3.3. Page List

If a page list is provided, i.e., R13 contains a valid shared HPA of a page containing page list entry, then each entry in the list is processed as follows:

A page list entry (see the format in 3.11.1) is used by TDH.IMPORT.MEM on input as follows:

- If the INVALID bit (63) is 0, the entry should contain a valid shared HPA of a page that can be converted to a TD private page (HKID bits must be 0). TDH.IMPORT.MEM uses this page for a MIGRATE operation.
- Else (the INVALID bit is 1), the entry doesn't point to a page. For a MIGRATE operation, TDH.IMPORT.MEM uses the applicable migration buffer for in-place import (see below). To facilitate parsing of the list entry on output (see below) it is recommended that the host VMM will set such entries to NULL_PA (all 1's).

A page list entry is updated by TDH.IMPORT.MEM on output as follows:

- If an entry was skipped due to a NOP operation or due to an error, the INVALID bit (63) is set to 1.

The following applies only if TDX_FEATURES0.IMPORT_PAGE_STATUS (bit 44) is enumerated as 1:

- If a page was provided on input (i.e., the INVALID bit was 0) but was not used by TDH.IMPORT.MEM as a TD private page, the INVALID bit is set to 1.
- If the import operation was CANCEL and it successfully removed a TD private page, and a page was not provided on input (the INVALID bit was 1), the page list entry is updated as follows:
 - Bits 51:0 contain the removed page HPA (HKID bits are 0).
 - The REMOVED bit (61) is set to 1 to indicate a removed page.
 - If the TDX Module is configured for Dynamic PAMT, the PAMT_REMOVAL_HINT bit (62) provides a hint that the PAMT pages mapping the removed page are empty and can be removed.
 - The INVALID bit (63) is cleared to 0.

If a page list is not provided, i.e., R13 contains NULL_PA (all 1's), then in-place import must be used for MIGRATE operations. Other import operations (NOP, REMIGRATE, CANCEL) never require a new TD private page.

5.4.17.3.4. Re-Import

Re-import is only allowed during the in-order import phase. The imported pages replace an older version of the same pages, if 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, then if the TDX module supports page release by the guest TD, as enumerated by TDX_FEATURES0.PAGE_RELEASE, it may be overwritten by a new version that is either MAPPED or PENDING. Else, it may only be overwritten by a new 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.

5.4.17.3.5. 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, as described above. 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.

5.4.17.3.6. 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 has 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.

5.4.17.3.7. S4 Resumption

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

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

5.4.17.3.8. Dynamic PAMT

If the TDX Module is configured for dynamic PAMT, the PAMT hierarchy can be built on demand. If a missing PAMT page pair is detected during operation, TDH.IMPORT.MEM returns with a TDX_INTERRUPTED_PAMT status in RAX. RCX is updated with the list entry for which the error happened. The host VMM is expected to add the missing PAMT page pair using TDH.PHYMEM.PAMT.ADD, then re-invoke TDH.IMPORT.MEM, keeping the same inputs as the original invocation except using the updated RCX value and setting R10.RESUME to 1.

5.4.17.3.9. Interruptibility

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

5.4.17.3.10. Removed Page Initialization

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

5.4.17.4. Operands Information

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

Table 5.68: TDH.IMPORT.MEM Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------|-------------|--|---------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ¹¹ |
| Explicit | RCX | HPA | GPA List page | GPA_LIST | RW | Shared | 4KB | None | None | None | None |
| Explicit | RDX | HPA | TDR page | TDR | RW | Opaque | 4KB | Shared | Shared | Shared | None |
| Explicit | R8 | HPA | MBMD | MBMD | R | Shared | 128B | None | None | None | None |
| Explicit | R9 | HPA | Migration buffer list | PAGE_LIST | R | Shared | 4KB | None | None | None | None |
| Explicit | R10 | Index | Mig. Stream context | Mig. Stream context | RW | Opaque | N/A | Exclusive | N/A | N/A | N/A |
| Explicit | R11 | HPA | MAC list page 1 | MAC list | R | Shared | 4KB | None | None | None | None |
| Explicit | R12 | HPA | MAC list page 2 | MAC list | R | Shared | 4KB | None | None | None | None |
| Explicit | R13 | HPA | Destination page list | Blob | RW | Private | 4KB | Exclusive | Shared | Shared | None |
| Explicit | R14 | HPA | L2 attributes list page | L2 page attributes | R | Shared | 4KB | None | None | None | None |
| Explicit | N/A | GPA | TD private pages (via GPA list) | Blob | None | Private | 4KB | None | None | None | None |
| Explicit | N/A | HPA | Migration buffer pages (via page list) | Blob | RW | Shared | 4KB | None | None | None | None |
| Explicit | N/A | HPA | Destination pages (via page list) | Blob | RW | Private | 4KB | Exclusive | Shared | Shared | Exclusive |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A | N/A |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared | N/A | N/A | N/A |
| Implicit | N/A | N/A | Migration context | N/A | RW | Opaque | N/A | Shared | N/A | N/A | N/A |
| Implicit | N/A | GPA | L1 Secure EPT tree | N/A | RW | Private | N/A | Shared | N/A | N/A | N/A |
| Implicit | N/A | GPA | L1 Secure EPT entries | SEPT Entry | RW | Private | N/A | Exclusive | N/A | N/A | N/A |
| Implicit | N/A | GPA | L2 Secure EPT trees | N/A | RW | Private | N/A | Shared(i) | N/A | N/A | N/A |

¹¹ ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------|-------------|--------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ¹¹ |
| Implicit | N/A | GPA | L2 Secure EPT entries | SEPT Entry | RW | Private | N/A | Exclusive(i) | N/A | N/A | N/A |

5.4.17.5. Completion Status Codes

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

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

5.4.18. TDH.IMPORT.STATE.IMMUTABLE Leaf

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

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

5.4.18.1. Input Operands

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

| Operand | Name | Description | | |
|---------|------------------|---|--|---|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 85 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 | |
| RCX | TDR | Destination TD handle and flags | | |
| | | Bits | Name | Description |
| | | 0 | Import Type | 0: TD Import 1: S4 Resumption |
| | | 11:1 | Reserved | Must be 0 |
| | | 51:12 | TDR HPA | HPA[51:12] of the destination TD's TDR page (HKID bits must be 0) |
| | 63:52 | Reserved | Must be 0 | |
| R8 | MBMD | HPA and size of memory of an MBMD structure in memory: | | |
| | | Bits | Name | Description |
| | | 51:0 | HPA | Bits 51:0 of the host physical address (including HKID bits) HPA must be aligned on 128 bytes. |
| | 63:52 | Size | Size of the memory buffer containing MBMD, in bytes The buffer size must be at least the maximum MBMD size, which is 128 bytes. See 3.13.1 for details. | |
| R9 | PAGE_LIST_INFO | Migration buffers list information – see 3.13.6.1 | | |
| R10 | MIG_STREAM | Migration stream and resume flag: | | |
| | | Bits | Name | Description |
| | | 15:0 | MIGS_INDEX | Migration stream index – must be 0 |
| | | 62:16 | RESERVED | Reserved: must be 0 |
| | 63 | RESUME | 0: This is a new invocation 1: This is resumption of a previously interrupted operation | |

5.4.18.2. Output Operands

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

| Operand | Name | Description |
|----------------------------|----------------------|--|
| RAX | STATUS | SEAMCALL instruction return code, see 5.4.1 |
| RCX | EXTENDED_ERROR_INFO1 | In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESUMABLE, RCX is unmodified. In case of an error related to non-memory state field import, as indicated by RAX, RCX contains the offending field identifier. See the status codes table below. In other cases, RCX returns 0. |
| RDX | EXTENDED_ERROR_INFO2 | In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESUMABLE, RDX is unmodified. In other cases, RDX returns 0. |
| AVX, AVX2 and AVX512 state | | May be reset to the architectural RESET state |
| Other | | Unmodified |

5.4.18.3. Leaf Function Description

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

5.4.18.3.1. Overview

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

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

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

15 5.4.18.3.2. Enumeration

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

5.4.18.3.3. Interruptibility

- 20 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.

5.4.18.3.4. Aspects of MAC Calculation Compatibility on TDX Module Update

25 If TDH.IMPORT.STATE.IMMUTABLE was interrupted, and later the TDX Module was updated before TDH.IMPORT.STATE.IMMUTABLE was resumed, there could be incompatibility issues with the format of the intermediate MRTD MAC context, held as part of the migration stream context (MIGSC), between the original and the new TD module.

On TDH.SYS.UPDATE, the host VMM can configure the TDX Module to detect such incompatibility. If detected, TDH.IMPORT.STATE.IMMUTABLE returns a TDX_INCOMPATIBLE_MBMD_MAC_CONTEXT status. In this case, the host VMM is expected to abort the migration session and may restart it. For details, see the [Base Spec] section titled "Compatibility Aspects of TD-Preserving Update".

If the host VMM did not configure the TDX Module to detect such incompatibility, it will not be detected by TDH.IMPORT.STATE.IMMUTABLE. However, TDH.IMPORT.STATE.IMMUTABLE will detect that the MAC is incorrect and will return a TDX_INCORRECT_MBMD_MAC error.

5.4.18.3.5. Import Abort

5 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.

5.4.18.3.6. S4 Resumption Abort

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

5.4.18.4. Operands Information

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

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

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

15

5.4.18.5. Completion Status Codes

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

| Completion Status Code | Description |
|-----------------------------------|-------------|
| TDX_INCOMPATIBLE_MBMD_MAC_CONTEXT | |
| TDX_INCORRECT_MBMD_MAC | |
| TDX_INTERRUPTED_RESUMABLE | |
| TDX_INVALID_MBMD | |
| TDX_INVALID_METADATA_LIST_HEADER | |

| Completion Status Code | Description |
|--------------------------------------|--|
| TDX_INVALID_RESUMPTION | |
| TDX_METADATA_FIELD_ID_INCORRECT | Field ID or sequence header is returned in RCX |
| TDX_METADATA_FIELD_NOT_WRITABLE | Field ID is returned in RCX |
| TDX_METADATA_FIELD_VALUE_NOT_VALID | Field ID is returned in RCX |
| TDX_METADATA_LIST_OVERFLOW | |
| TDX_MIGRATION_SESSION_KEY_NOT_SET | |
| TDX_MIGRATION_STREAM_STATE_INCORRECT | |
| TDX_NUM_MIGS_HIGHER_THAN_CREATED | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_REQUIRED_METADATA_FIELD_MISSING | Required field ID is returned in RCX |
| TDX_RND_NO_ENTROPY | Failed to generate a random migration encryption key. This is typically caused by an entropy error of the CPU's random number generator, and may be impacted by RDSEED, RDRAND or PCONFIG executing on other LPs. The operation should be retried. |
| TDX_SERVTD_EXT_MISMATCH | Rebind SERVTD_EXT hash did not match |
| TDX_SUCCESS | Operation is successful |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |
| TDX_VIRTUAL_MSR_VALUE_NOT_VALID | |

5.4.19. TDH.IMPORT.STATE.TD Leaf

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

5.4.19.1. Input Operands

Table 5.74: TDH.IMPORT.STATE.TD Input Operands Definition

| Operand | Name | Description | | |
|---------|------------------|---|----------------|---|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 86 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | | 63:24 | Reserved | Must be 0 |
| RCX | TDR | HPA of the destination TD TDR page (HKID bits must be 0) | | |
| R8 | MBMD | HPA and size of memory of an MBMD structure in memory: | | |
| | | Bits | Name | Description |
| | | 51:0 | HPA | Bits 51:0 of the host physical address (including HKID bits) HPA must be aligned on 128 bytes. |
| 63:52 | Size | Size of the memory buffer containing MBMD, in bytes The buffer size must be at least the maximum MBMD size, which is 128 bytes. See 3.13.1 for details. | | |
| R9 | PAGE_LIST_INFO | Migration buffers list information – see 3.13.6.1 | | |
| R10 | MIG_STREAM | Migration stream and resume flag: | | |
| | | Bits | Name | Description |
| | | 15:0 | MIGS_INDEX | Migration stream index – must be 0 |
| | | 62:16 | RESERVED | Reserved: must be 0 |
| | | 63 | RESUME | 0: This is a new invocation 1: This is resumption of a previously interrupted operation |

5

5.4.19.2. Output Operands

Table 5.75: TDH.IMPORT.STATE.TD Output Operands Definition

| Operand | Name | Description |
|---------|----------------------|--|
| RAX | STATUS | SEAMCALL instruction return code, see 5.4.1 |
| RCX | EXTENDED_ERROR_INFO1 | In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESUMABLE, RCX is unmodified. In case of an error related to non-memory state field import, as indicated by RAX, RCX contains the offending field identifier. See the status codes table below. |

| Operand | Name | Description |
|----------------------------|----------------------|---|
| | | In other cases, RCX returns 0. |
| RDX | EXTENDED_ERROR_INFO2 | In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESUMABLE, RDX is unmodified. In other cases, RDX returns 0. |
| AVX, AVX2 and AVX512 state | | May be reset to the architectural RESET state |
| Other | | Unmodified |

5.4.19.3. Leaf Function Description

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

5 5.4.19.3.1. Overview

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

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

10 5.4.19.3.2. Enumeration

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

15 5.4.19.3.3. Interruptibility

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

20 5.4.19.3.4. Import Abort

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

25 5.4.19.3.5. Aspects of MAC Calculation Compatibility on TDX Module Update

If TDH.IMPORT.STATE.TD was interrupted, and later the TDX Module was updated before TDH.IMPORT.STATE.TD was resumed, there could be incompatibility issues with the format of the intermediate MRTD MAC context, held as part of the migration stream context (MIGSC), between the original and the new TD module.

25 On TDH.SYS.UPDATE, the host VMM can configure the TDX Module to detect such incompatibility. If detected, TDH.IMPORT.STATE.TD returns a TDX_INCOMPATIBLE_MBMD_MAC_CONTEXT status. In this case, the host VMM is expected to abort the migration session and may restart it. For details, see the [Base Spec] section titled “Compatibility Aspects of TD-Preserving Update”.

30 If the host VMM did not configure the TDX Module to detect such incompatibility, it will not be detected by TDH.IMPORT.STATE.TD. However, TDH.IMPORT.STATE.TD will detect that the MAC is incorrect and will return a TDX_INCORRECT_MBMD_MAC error.

5.4.19.3.6. S4 Resumption Abort

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

5.4.19.4. Operands Information

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

Table 5.76: TDH.IMPORT.STATE.VP Operands Information Definition

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

5

5.4.19.5. Completion Status Codes

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

| Completion Status Code | Description |
|-----------------------------------|-------------|
| TDX_INCOMPATIBLE_MBMD_MAC_CONTEXT | |
| TDX_INTERRUPTED_RESUMABLE | |
| TDX_INCORRECT_MBMD_MAC | |
| TDX_INTERRUPTED_RESUMABLE | |
| TDX_INVALID_MBMD | |
| TDX_INVALID_METADATA_LIST_HEADER | |
| TDX_INVALID_RESUMPTION | |

| Completion Status Code | Description |
|--------------------------------------|--|
| TDX_METADATA_FIELD_ID_INCORRECT | Field ID or sequence header is returned in RCX |
| TDX_METADATA_FIELD_NOT_WRITABLE | Field ID is returned in RCX |
| TDX_METADATA_FIELD_VALUE_NOT_VALID | Field ID is returned in RCX |
| TDX_METADATA_LIST_OVERFLOW | |
| TDX_MIGRATION_STREAM_STATE_INCORRECT | |
| TDX_REQUIRED_METADATA_FIELD_MISSING | Required field ID is returned in RCX |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |

5.4.20. TDH.IMPORT.STATE.VP Leaf

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

5.4.20.1. Input Operands

Table 5.78: TDH.IMPORT.STATE.VP Input Operands Definition

| Operand | | Description | | |
|---------|------------------|--|----------------|--|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 87 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | | 63:24 | Reserved | Must be 0 |
| RCX | TDVPR | HPA of the destination TD VCPU's TDVPR page (HKID bits must be 0) | | |
| R8 | MBMD | HPA and size of memory of an MBMD structure in memory: | | |
| | | Bits | Name | Description |
| | | 51:0 | HPA | Bits 51:0 of the host physical address (including HKID bits) HPA must be aligned on 128 bytes. |
| 63:52 | Size | Size of the memory buffer containing MBMD, in bytes The buffer size must be at least the maximum MBMD size, which is 128 bytes. See 3.13.1 for details. | | |
| R9 | PAGE_LIST_INFO | Migration buffers list information – see 3.13.6.1 | | |
| R10 | MIG_STREAM | Migration stream and resume flag: | | |
| | | Bits | Name | Description |
| | | 15:0 | MIGS_INDEX | Migration stream index If N migration streams have been created by TDH.MIG.STREAM.CREATE, then MIGS_INDEX must be lower than N-1. |
| | | 62:16 | RESERVED | Reserved: must be 0 |
| | | 63 | RESUME | 0: This is a new invocation 1: This is resumption of a previously interrupted operation |

5

5.4.20.2. Output Operands

Table 5.79: TDH.IMPORT.STATE.VP Output Operands Definition

| Operand | Name | Description |
|---------|----------------------|---|
| RAX | STATUS | SEAMCALL instruction return code, see 5.4.1 |
| RCX | EXTENDED_ERROR_INFO1 | In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESUMABLE, RCX is unmodified. |

| Operand | Name | Description |
|----------------------------|----------------------|---|
| | | In case of an error related to non-memory state field import, as indicated by RAX, RCX contains the offending field identifier. See the status codes table below. In other cases, RCX returns 0. |
| RDX | EXTENDED_ERROR_INFO2 | In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESUMABLE, RDX is unmodified. In other cases, RDX returns 0. |
| AVX, AVX2 and AVX512 state | | May be reset to the architectural RESET state |
| Other | | Unmodified |

5.4.20.3. Leaf Function Description

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

5 5.4.20.3.1. Overview

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

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

10 5.4.20.3.2. Enumeration

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

15 5.4.20.3.3. Interruptibility

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

20 5.4.20.3.4. Aspects of MAC Calculation Compatibility on TDX Module Update

If TDH.IMPORT.STATE.VP was interrupted, and later the TDX Module was updated before TDH.IMPORT.STATE.VP was resumed, there could be incompatibility issues with the format of the intermediate MRTD MAC context, held as part of the migration stream context (MIGSC), between the original and the new TD module.

On TDH.SYS.UPDATE, the host VMM can configure the TDX Module to detect such incompatibility. If detected, TDH.IMPORT.STATE.VP returns a TDX_INCOMPATIBLE_MBMD_MAC_CONTEXT status. In this case, the host VMM is expected to abort the migration session and may restart it. For details, see the [Base Spec] section titled “Compatibility Aspects of TD-Preserving Update”.

25 If the host VMM did not configure the TDX Module to detect such incompatibility, it will not be detected by TDH.IMPORT.STATE.VP. However, TDH.IMPORT.STATE.VP will detect that the MAC is incorrect and will return a TDX_INCORRECT_MBMD_MAC error.

30 5.4.20.3.5. Import Abort

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

5.4.20.3.6. S4 Resumption Abort

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

5.4.20.3.7. VCPU Association

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

5.4.20.4. Operands Information

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

10 **Table 5.80: TDH.IMPORT.STATE.VP Operands Information Definition**

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

5.4.20.5. Completion Status Codes

Table 5.81: TDH.IMPORT.STATE.VP Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-----------------------------------|-------------|
| TDX_ALL_VCPUS_IMPORTED | |
| TDX_INCOMPATIBLE_MBMD_MAC_CONTEXT | |
| TDX_INTERRUPTED_RESUMABLE | |
| TDX_INTERRUPTED_RESUMABLE | |
| TDX_INCORRECT_MBMD_MAC | |
| TDX_INTERRUPTED_RESUMABLE | |
| TDX_INVALID_MBMD | |
| TDX_INVALID_METADATA_LIST_HEADER | |

| Completion Status Code | Description |
|-------------------------------------|---|
| TDX_INVALID_RESUMPTION | |
| TDX_METADATA_FIELD_ID_INCORRECT | Field ID or sequence header is returned in RCX |
| TDX_METADATA_FIELD_NOT_WRITABLE | Field ID is returned in RCX |
| TDX_METADATA_FIELD_VALUE_NOT_VALID | Field ID is returned in RCX |
| TDX_METADATA_LIST_OVERFLOW | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_REQUIRED_METADATA_FIELD_MISSING | Required field ID is returned in RCX |
| TDX_SUCCESS | Operation is successful |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |
| TDX_VCPU_STATE_INCORRECT | If this is the initial call to TDH.IMPORT.STATE.VP for this VCPU, the VCPU is expected to be in the VCPU_READY state. If this is a resumption of an interrupted call, the VCPU is expected to be in the VCPU_IMPORT state. For details, see the [Base Spec] discussion of VPCU transitions. |
| TDX_X2APIC_ID_NOT_UNIQUE | |

5.4.21. 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.

5.4.21.1. Input Operands

5

Table 5.82: TDH.IMPORT.TRACK Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 84 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | HPA of the destination 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) HPA must be aligned on 128 bytes. |
| | 63:52 | Size | Size of the memory buffer containing MBMD, in bytes The buffer size must be at least the maximum MBMD size, which is 128 bytes. See 3.13.1 for details. |
| R10 | Migration stream index – must be 0 | | |

5.4.21.2. Output Operands

Table 5.83: TDH.IMPORT.TRACK Output Operands Definition

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

10 5.4.21.3. Leaf Function Description

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

5.4.21.3.1. Overview

15

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.

5.4.21.3.2. Enumeration

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

5.4.21.3.3. Import Abort

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

5.4.21.3.4. S4 Resumption

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

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

5.4.21.4. Operands Information

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

Table 5.84: TDH.IMPORT.TRACK Operands Information Definition

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

5.4.21.5. Completion Status Codes

Table 5.85: TDH.IMPORT.TRACK Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|--------------------------------------|-------------|
| TDX_INCORRECT_MBMD_MAC | |
| TDX_INVALID_MBMD | |
| TDX_MIGRATION_STREAM_STATE_INCORRECT | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | |

| | |
|-------------------------------------|-------------------------|
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SOME_VCPUS_NOT_MIGRATED | |
| TDX_SUCCESS | Operation is successful |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |

5.4.22. TDH.INTR.CONFIG Leaf

Unreleased Feature: This section is related to Enhanced Interrupt Virtualization, a feature which has not been released yet at the time of writing of this document. Details provided below serve as a preview and are subject to change.

- 5 Allocate and initialize interrupt-related TD-scope metadata for one specific L1 or L2 VM within a TD.

5.4.22.1. Input Operands

Table 5.86: TDH.INTR.CONFIG Input Operands Definition

| Operand | Description | | | | |
|---------|---|---------------------|--|---------|-----------------------------------|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | | | |
| | Bits | Field | Description | | |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 62 | | |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 | | |
| | 63:24 | Reserved | Must be 0 | | |
| RCX | TD handle and flags: | | | | |
| | Bits | Name | Description | | |
| | 11:0 | Reserved | Reserved: must be 0 | | |
| | 51:12 | TDR_HPA | Bits 51:12 of the host physical address of the parent TDR page (HKID bits must be 0) | | |
| | 53:52 | VM | Virtual machine index | | |
| 63:54 | Reserved | Reserved: must be 0 | | | |
| RDX | PIDPT (Posted Interrupt Descriptor Pointers Table) information: | | | | |
| | Bits | Name | Description | | |
| | 11:0 | PIDPT_NUM_PAGES | The number of 4KB contiguous physical pages which will be used as PIDPT If PIDPT_NUM_PAGES is 0, no PIDPT is allocated and IPI virtualization is not supported. PID_MODE (see below) may be either REG_PID (for L1) or SHR_PID. Else, PID_MODE must be either SEC_PID or DUAL_PID. | | |
| | 51:12 | PIDPT_HPA | Bits 51:12 of the host physical address (HKID bits must be 0) of multiple 4KB contiguous physical pages which will be used as PIDPT If PIDPT_NUM_PAGES is 0, PIDPT_HPA must be 0. | | |
| 63:52 | Reserved | Reserved: must be 0 | | | |
| R8 | Posted interrupts configuration: | | | | |
| | 1:0 | PID_MODE | PID mode: | | |
| | | | Value | Name | Meaning |
| | | | 0 | REG_PID | Regular PID – allowed only for L1 |

| Operand | Description | | | | |
|---------|-------------|----------------|--|--|-------------------------------------|
| | | | 1 | SEC_PID | Secure PID |
| | | | 2 | SHR_PID | Shared PID |
| | | | 3 | DUAL_PID | Dual PID: Secure PID and Shared PID |
| | | | Other | Reserved | Reserved |
| | 6:2 | Reserved | Reserved: must be 0 | | |
| | 7 | MAIN_NV_SHARED | 0 | MAIN_NV value, if not 0, is unique among all MAIN_NVs and SHR_NVs of a VM of this TD | |
| | | | 1 | MAIN_NV value is assigned to at least one other MAIN_NV or SHR_NV of a VM of this TD | |
| | 15:8 | MAIN_NV | Main PID notification vector If PID_MODE is REG_PID, MAIN_NV must either be 0 (indicating no notification vector) or between 31 and 255. Else, if PID_MODE is SEC_PID or DUAL_PID, MAIN_NV must be between 31 and 255. Else, MAIN_NV must be 0. | | |
| | 23:16 | SHR_NV | Shared PID notification vector If PID_MODE is SHR_PID or DUAL_PID, SHR_NV must either be 0 (indicating no notification vector) or between 31 and 255. Else, SHR_NV must be 0. | | |
| | 63:24 | Reserved | Reserved: must be 0 | | |

5.4.22.2. Output Operands

Table 5.87: TDH.INTR.CONFIG Output Operands Definition

| Operand | Description | | |
|---------|--|-----------------|---|
| RAX | SEAMCALL instruction return code – see 5.4.1 | | |
| RDX | Updated PIDPT information: | | |
| | Bits | Name | Description |
| | 11:0 | PIDPT_NUM_PAGES | The remaining number of 4KB contiguous physical pages to be used as PIDPT |
| | 51:12 | PIDPT_HPA | Updated bits 51:12 of the host physical address (HKID bits must be 0) of multiple 4KB contiguous physical pages to be used as PIDPT |
| | 63:52 | Reserved | Reserved: must be 0 |
| Other | Unmodified | | |

5 5.4.22.3. Leaf Function Description

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

5.4.22.3.1. Overview

TDH.INTR.CONFIG allocates and configures enhance interrupt virtualization resources and metadata for a specific VM (L1 or L2) within a TD.

5.4.22.3.2. Enumeration

5 Support of TDH.INTR.CONFIG is enumerated by TDX_FEATURES0.ENHANCED_INTR_VIRTUALIZATION (bit 45), readable by TDH.SYS.RD*.

5.4.22.3.3. Legacy Compatibility

10 TDH.INTR.CONFIG is only allowed for L1 if the legacy method of posted interrupt configuration, by setting the TD VMCS' pin-based execution controls' "process posted interrupts" bit using TDH.VP.WR, has not been used. Once THD.INTR.CONFIG for L1 completes successfully, only the "posted interrupt descriptor address" VMCS field can be written.

5.4.22.3.4. Configuration During TD Build and TD Run Time

TDH.INIT.CONFIG may be called at any time after TD initialization (TDH.MNG.INIT), with the following restrictions:

- TDH.INTR.CONFIG may only be completed successfully once per L1 and each L2 VM.
- 15 • TDH.INTR.CONFIG may not be called for an L2 VM once that VM was entered for the first time.

L1 configuration becomes effective on the TD entry following TDH.INIT.CONFIG successful completion. If done after TD build, when L1 may already be running, the host VMM may send IPIs to the TD's VCPUs to help ensure the configuration is effective.

5.4.22.3.5. Configuration or Reconfiguration During or After TD Migration

20 Most posted interrupt resources and metadata allocated and configured by TDH.INTR.CONFIG are not migrated. If TDH.INTR.CONFIG was called on the source platform for a certain VM (L1 or L2) before export, the host VMM on the destination platform must call TDH.INTR.CONFIG to reallocate and reconfigure the posted interrupt resources for that VM. This must be done following TD state import by TDH.IMPORT.STATE.TD, and before VCPU state is imported by TDH.IMPORT.STATE.VP.

25 If TDH.INTR.CONFIG was not called on the source platform for a certain VM (L1 or L2) before export, the host VMM on the destination platform may call TDH.INTR.CONFIG for that VM after import is committed (by TDH.IMPORT, with the same restrictions as discussed in 5.4.22.3.4 above).

5.4.22.3.6. PID Configurations for Posted Interrupts and IPI Virtualization

30 The table below summarizes the available PID configurations for posted interrupts and IPI virtualization. For details, see the [Interrupt Virtualization Spec].

Table 5.88: PID Configurations

| Name | Configuration | Posted Interrupts | IPI Virtualization |
|----------|--|-------------------------------|--------------------|
| N/A | No PID (TDH.INTR.CONFIG not called) | Not enabled | Not enabled |
| REG_PID | Regular PID in Shared Memory (L1 only) | Enabled, not filtered | Not enabled |
| SHR_PID | Shared PID Only | Enabled, filtered by PIR_MASK | Not enabled |
| SEC_PID | Secure PID Only | Not enabled (except for IPI) | Enabled |
| DUAL_PID | Secure PID + Shared PID | Enabled, filtered by PIR_MASK | Enabled |

5.4.22.3.7. Notification Vectors

35 Notification vector MAIN_NV and SHR_NV may be configured with unique, non-unique or null values. This trades performance and functionality vs. physical interrupt vector space consumption. The host VMM must specify, using the MAIN_NV_SHARED input flag, whether the configured MAIN_NV value is unique or not. If specified as a unique value, the TDX Module checks that all other MAIN_NV or SHR_NV values configured for this TD are different.

For details, see the section titled "Notification Vector Uniqueness and its Implication on Performance and Functionality" in the [Interrupt Virtualization Spec].

5.4.22.3.8. Interruptibility

TDH.INTR.CONFIG is interruptible. If a pending interrupt is detected during operation, TDH.INTR.CONFIG returns with a TDX_INTERRUPTED_RESUMABLE status in RAX. The PIDPT information in RCX is updated to reflect the operation so far.

TDH.INTR.CONFIG is designed to be invoked in a loop until all required PIDPT pages have been added:

- 5 1. Call TDH.INTR.CONFIG.
2. While RAX indicates TDX_INTERRUPTED_RESUMABLE:
 - 2.1. Call TDH.INTR.CONFIG with the GPR values as returned by the previous call.
 - 2.2. If an error indication other than TDX_INTERRUPTED_RESUMABLE is returned, abort.

5.4.22.3.9. Error Handling

- 10 If TDH.INTR.CONFIG returns with an error status, it may be called again. This is especially important for transient errors (e.g., TDX_OPERNAD_BUSY) or errors requiring some host VMM operation (e.g., missing PAMT pages when Dynamic PAMT is used).

5.4.22.4. Operands Information

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

Table 5.89: TDH.INTR.CONFIG Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------|-------------|----------------|---------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ¹² |
| Explicit | RCX | HPA | TDR page | TDR | RW | Opaque | 4KB | Shared | Shared | Shared | None |
| Explicit | RDX | HPA | PIDPT Pages | PIDPT | RW | Opaque | 4KB | Exclusive | Shared | Shared | None |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A | None |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Exclusive | N/A | N/A | None |

5.4.22.5. Completion Status Codes

Table 5.90: TDH.INTR.CONFIG Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|---|-------------|
| TDX_ENHANCED_INTR_VIRT_CONFIGURED | |
| TDX_INTR_VIRT_CONFIG_ALREADY_DONE | |
| TDX_LEGACY_INTR_VIRT_CONFIGURED | |
| TDX_PIDPT_SIZE_TOO_SMALL | |
| TDX_PID_MODE_IMPORT_MISMATCH | |
| TDX_SUCCESS | |
| TDX_VCPUS_ALREADY_IMPORTED | |
| Unreleased Feature: This list is not comprehensive | |

20

¹² ACT is implemented only on client platforms. This is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

5.4.23. 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.

5.4.23.1. Input Operands

Table 5.91: TDH.MEM.PAGE.ADD Input Operands Definition

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

5.4.23.2. Output Operands

Table 5.92: TDH.MEM.PAGE.ADD Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| RCX | <p>Extended error information part 1</p> <p>In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2</p> <p>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.</p> <p>In other cases, RCX returns 0.</p> |

| Operand | Description |
|---------|---|
| RDX | Extended error information part 2 In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2 In other cases, RDX returns 0. |
| Other | Unmodified |

5.4.23.3. Leaf Function Description

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

5 5.4.23.3.1. Overview

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.

5.4.23.3.2. In-Place Add

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

5.4.23.3.3. Dynamic PAMT

15 If the TDX Module is configured for dynamic PAMT, the PAMT hierarchy can be built on demand. A TDX_MISSING_PAMT_PAGE_PAIR status indicates that a PAMT page pair is missing. The host VMM may add it using TDH.PHYMEM.PAMT.ADD and retry the operation.

5.4.23.3.4. TD Measurement Register (MRTD) Extension

TDH.MEM.PAGE.ADD extends the TD's 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.

5.4.23.3.5. MRTD Extension Compatibility Aspects

25 If MRTD was initialized by TDH.MNG.INIT, and later the TDX Module was updated before the current TDH.MEM.PAGE.ADD was called, there could be incompatibility issues with the intermediate MRTD context format between the original and the new TD module. On TDH.SYS.UPDATE, the host VMM can configure the TDX Module to detect such incompatibility. If detected, TDH.MEM.PAGE.ADD returns a TDX_INCOMPATIBLE_MRTD_CONTEXT status. In this case, the host VMM is expected to tear down the TD and rebuild it. For details, see the [Base Spec] section titled "Compatibility Aspects of TD-Preserving Update".

5.4.23.4. Operands Information

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

Table 5.93: TDH.MEM.PAGE.ADD Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------|-------------|--|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ¹³ |
| Explicit | RCX | GPA | TD private page (GPA) ¹⁴ | Blob | RW | Private | 4KB | N/A | N/A | N/A | N/A |
| Explicit | RDX | HPA | TDR page | Blob | RW | Opaque | 4KB | Exclusive | Shared | Shared | None |
| Explicit | R8 | HPA | TD private page (HPA) ¹⁴ | Blob | RW | Private | 4KB | Exclusive | Shared | Shared | Exclusive |
| Explicit | R9 | HPA | Source page | Blob | R | Shared | 4KB | None | None | None | None |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Exclusive(i) | N/A | N/A | None |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Exclusive(i) | N/A | N/A | None |
| Implicit | N/A | GPA | Secure EPT tree | N/A | RW | Private | N/A | Exclusive(i) | N/A | N/A | None |
| Implicit | N/A | GPA | Secure EPT entry | SEPT Entry | RW | Private | N/A | Exclusive(i) | N/A | N/A | None |

5.4.23.5. Completion Status Codes

Table 5.94: TDH.MEM.PAGE.ADD Completion Status Codes (Returned in RAX) Definition

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

5

¹³ ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

¹⁴ RCX and R8 denote the same TD private page operand, using HPA and GPA respectively

5.4.24. TDH.MEM.PAGE.AUG Leaf

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

5.4.24.1. Input Operands

Table 5.95: TDH.MEM.PAGE.AUG Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|---|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 6 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | EPT mapping information: | | |
| | Bits | Name | Description |
| | 2:0 | Level | Level of the EPT entry that will map the new page – see 3.6.1: must be 0 (4KB) or 1 (2MB) |
| | 11:3 | Reserved | Reserved: must be 0 |
| | 51:12 | GPA | Bits 51:12 of the guest physical address to be mapped for the new Secure EPT page |
| | 63:52 | Reserved | Reserved: must be 0 |
| RDX | Host physical address of the parent TDR page (HKID bits must be 0) | | |
| R8 | Host physical address of the target page to be added to the TD (HKID bits must be 0) | | |

5

5.4.24.2. Output Operands

Table 5.96: TDH.MEM.PAGE.AUG Output Operands Definition

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

| Operand | Description |
|---------|-------------|
| Other | Unmodified |

5.4.24.3. Leaf Function Description

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

5.4.24.3.1. Overview

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.

5.4.24.3.2. Dynamic PAMT

If the TDX Module is configured for dynamic PAMT, the PAMT hierarchy can be built on demand. A TDX_MISSING_PAMT_PAGE_PAIR status indicates that a PAMT page pair is missing (applies only for 4KB pages). The host VMM may add it using TDH.PHYMEM.PAMT.ADD and retry the operation.

5.4.24.4. Operands Information

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

Table 5.97: TDH.MEM.PAGE.AUG Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------|-------------|--|------------------|--------|---------------------|----------------------------------|--------------------------|----------------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ¹⁵ |
| Explicit | RCX | GPA | TD private page (GPA) ¹⁶ | Blob | None | Private | 2 ^{12+9*Level} Bytes | N/A | N/A | N/A | N/A |
| Explicit | RDX | HPA | TDR page | TDR | RW | Opaque | 4KB | Shared | Shared | Shared | None |
| Explicit | R8 | HPA | TD private page (HPA) ¹⁶ | Blob | None | Private | 2 ^{12+9*Level} Bytes | Exclusive | Shared ¹⁷ | Shared | Exclusive |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A | N/A |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared | N/A | N/A | N/A |
| Implicit | N/A | GPA | Secure EPT tree | N/A | RW | Private | N/A | Shared | N/A | N/A | N/A |
| Implicit | N/A | GPA | Secure EPT entry | SEPT Entry | RW | Private | N/A | Exclusive | N/A | N/A | N/A |

¹⁵ ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

¹⁶ RCX and R8 denote the same TD private page operand, using HPA and GPA respectively

¹⁷ Applicable for 4KB pages only

5.4.24.5. *Completion Status Codes*

Table 5.98: TDH.MEM.PAGE.AUG Completion Status Codes (Returned in RAX) Definition

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

5.4.25. TDH.MEM.PAGE.DEMOTE Leaf

Unreleased Feature: Some of the text in this section is related to Non-Blocking Export, a feature which has not been released yet at the time of writing of this document. Details related to that feature serve as a preview and are subject to change.

5 TDH.MEM.PAGE.DEMOTE splits a single 2MB page into 512 4KB pages, or a single 1GB page into 512 2MB pages. The leaf SEPT entry that mapped the large page changes to a non-leaf entry pointing to a new SEPT page which maps the 512 small pages.

If the TDX Module is configured to use Dynamic PAMT and the large page level is 1 (2MB), the leaf PAMT entry that was associated with the large page’s HPA changes to a non-leaf PAMT entry pointing to a pair of PAMT pages that are associated with the 512 small pages.

10

5.4.25.1. Input Operands

Table 5.99: TDH.MEM.PAGE.DEMOTE Input Operands Definition

| Operand | Description | | | | |
|---------|---|---------------------|---|---|-------------|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | | | |
| | Bits | Field | Description | | |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 15 | | |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 | | |
| | 63:24 | Reserved | Must be 0 | | |
| RCX | EPT mapping information: | | | | |
| | Bits | Name | Description | | |
| | 2:0 | Level | Level of the Secure EPT entry that maps the large page to be split: either 1 (2MB) or 2 (1GB) – see 3.6.1 | | |
| | 11:3 | Reserved | Reserved: must be 0 | | |
| | 51:12 | GPA | Bits 51:12 of the guest physical address of the 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 | TD handle and flags: | | | | |
| | Bits | Name | Description | | |
| | 0 | L2_SEPT_ADD_MODE | New L2 SEPT pages addition mode: | | |
| | | | Value | Name | Description |
| 0 | | | DENSE | New L2 SEPT pages are added, if provided by R9, R10 or R11. | |

| Operand | Description | | | | |
|---------|--|----------|---|--------|---|
| | | | 1 | SPARSE | New L2 SEPT pages are added, if provided by R9, R10 or R11, but only for L2 VMs where the L1 VMM has created a page alias (using TDG.MEM.PAGE.ATTR.WR). |
| | | | In both cases, a new L2 SEPT must be provided for L2 VMs where a page alias exists. This bit is ignored if the TD has no L2 VMs. | | |
| | 11:1 | Reserved | Reserved: must be 0 | | |
| | 51:12 | TDR_HPA | Bits 51:12 of the host physical address of the parent TDR page (HKID bits must be 0) | | |
| | 63:52 | Reserved | Reserved: must be 0 | | |
| R8 | Host physical address of the new L1 Secure EPT page to be added to the TD (HKID bits must be 0) | | | | |
| R9 | <p>If the number of L2 VMs is ≥ 1, R9 contains the host physical address of the new Secure EPT page to be added to L2 VM #1's SEPT tree (HKID bits must be 0).</p> <p>Else (the number of L2 VMs is 0), R9 is ignored.</p> <p>If the value of R9 is NULL_PA (-1), no new SEPT page is added to L2 VM #1's SEPT. If the demoted TD private page has an L2 page alias for L2 VM #1, this is an error.</p> <p>Else, bit 63 of R9 is ignored. If L2_SEPT_ADD_MODE is 1, the new SEPT page is only used if the demoted TD private page has an L2 page alias for L2 VM #1. Else, the new SEPT page is always used.</p> | | | | |
| R10 | <p>If the number of L2 VMs is ≥ 2, R10 contains the host physical address of the new Secure EPT page to be added to L2 VM #2's SEPT tree (HKID bits must be 0).</p> <p>Else (the number of L2 VMs is 0 or 1), R10 is ignored.</p> <p>If the value of R10 is NULL_PA (-1), no new SEPT page is added to L2 VM #2's SEPT. If the demoted TD private page has an L2 page alias for L2 VM #2, this is an error.</p> <p>Else, bit 63 of R10 is ignored. If L2_SEPT_ADD_MODE is 1, the new SEPT page is only used if the demoted TD private page has an L2 page alias for L2 VM #2. Else, the new SEPT page is always used.</p> | | | | |
| R11 | <p>If the number of L2 VMs is ≥ 3, R11 contains the host physical address of the new Secure EPT page to be added to L2 VM #3's SEPT tree (HKID bits must be 0).</p> <p>Else (the number of L2 VMs is 0, 1 or 2), R11 is ignored.</p> <p>If the value of R11 is NULL_PA (-1), no new SEPT page is added to L2 VM #3's SEPT. If the demoted TD private page has an L2 page alias for L2 VM #3, this is an error.</p> <p>Else, bit 63 of R11 is ignored. If L2_SEPT_ADD_MODE is 1, the new SEPT page is only used if the demoted TD private page has an L2 page alias for L2 VM #3. Else, the new SEPT page is always used.</p> | | | | |
| R12 | <p>If the TDX Module is configured to use Dynamic PAMT and the large page level is 1 (2MB), R12 contains the host physical address of a new PAMT page (HKID bits must be 0).</p> <p>Else, R12 is ignored.</p> | | | | |
| R13 | <p>If the TDX Module is configured to use Dynamic PAMT and the large page level is 1 (2MB), R13 contains the host physical address of a new PAMT page (HKID bits must be 0).</p> <p>Else, R13 is ignored.</p> | | | | |

5.4.25.2. Output Operands**Table 5.100: TDH.MEM.PAGE.DEMOTE Output Operands Definition**

| Operand | Description |
|---------|---|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| RCX | In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESTARTABLE, RCX is unmodified. Else, RCX returns extended error information part 1. In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page; it may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. In other cases, RCX returns 0. |
| RDX | In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESTARTABLE, RDX is unmodified. Else, RDX returns extended error information part 2. In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2 In other cases, RDX returns 0. |
| R9 | If TDH.MEM.PAGE.DEMOTE terminated successfully, the number of L2 VMs is ≥ 1 and the page whose HPA was provided in R9 was not used as an SEPT page for any reason, R9 is updated with bit 63 set to 1. Else, R9 is unmodified. |
| R10 | If TDH.MEM.PAGE.DEMOTE terminated successfully, the number of L2 VMs is ≥ 2 and the page whose HPA was provided in R10 was not used as an SEPT page for any reason, R10 is updated with bit 63 set to 1. Else, R10 is unmodified. |
| R11 | If TDH.MEM.PAGE.DEMOTE terminated successfully, the number of L2 VMs is ≥ 3 and the page whose HPA was provided in R11 was not used as an SEPT page for any reason, R11 is updated with bit 63 set to 1. Else, R11 is unmodified. |
| Other | Unmodified |

5.4.25.3. Leaf Function Description

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

5.4.25.3.1. Overview

TDH.MEM.PAGE.DEMOTE splits a large TD private page (2MB or 1GB) into 512 small pages (4KB or 2MB, respectively) and adds a new Secure EPT page to map those small pages. If the large page is mapped in any L2 SEPTs, TDH.MEM.PAGE.DEMOTE splits those mapping and adds new L2 Secure EPT pages to map the demoted page.

5.4.25.3.2. Enumeration

TDH.MEM.PAGE.DEMOTE support of non-blocking mapping resize is enumerated by TDX_FEATURES0.NON_BLOCKING_RESIZE (bit 35), readable using TDH.SYS.RD*.

Support of dynamic PAMT is enumerated by TDX_FEATURES0.DYNAMIC_PAMT (bit 36).

Support of no interruption if the TD is not partitioned in enumerated by TDX_FEATURES0.ENHANCED_DEMOTE_INTERRUPTIBILITY (bit 51).

5.4.25.3.3. Blocking and TLB Tracking

If the TDX Module supports non-blocking mapping resize, no blocking and tracking of the demoted page is required.

5 Else, if the TD may be running, the demoted page must be blocked and TLB tracked. Else (e.g., TDH.MR.FINALIZE has not yet been executed, or the TD has been paused for export), no blocking and tracking is required.

5.4.25.3.4. New SEPT Pages

10 If the TDX Module supports non-blocking mapping resize, then the new SEPT pages specified by R8, R9, R10 and R11 may be either 4KB physical pages not currently used by TDX, or they may be PT_TR pages (i.e., former SEPT pages converted by TDH.MEM.PAGE.PROMOTE) owned by the specified TD. TDH.MEM.PAGE.DEMOTE checks PT_TR pages for TLB tracking.

Otherwise (non-blocking mapping resize is not supported), the new SEPT pages must be 4KB physical pages not currently used by TDX.

5.4.25.3.5. L2 SEPT Population

15 TDH.MEM.PAGE.DEMOTE supports multiple host VMM policies for populating the L2 SEPT trees.

- Dense mode is when the host VMM maintains an L2 SEPT page for each L1 SEPT page. This mode is selected by setting L2_SEPT_ADD_MODE to 0.
- Sparse mode is when the host VMM only maintains an L2 SEPT page for a certain L1 SEPT page on demand, i.e., when there's a need to map a TD private page in an L2 VM's GPA space. This mode is selected by setting L2_SEPT_ADD_MODE to 1. The host VMM provides new SEPT pages, but they are only used if a page alias exists for the relevant L2 VM.

The host VMM may also choose to maintain L2 SEPT trees only for a subset of the L2 VMs (e.g., if a TD is created with a certain number of L2 VMs but not all of them are currently in use). The host VMM can do so by providing NULL_PA as the new SEPT page(s) HPA.

5.4.25.3.6. Dynamic PAMT

25 If the TDX Module is configured for dynamic PAMT, the PAMT hierarchy can be built on demand. A TDX_MISSING_PAMT_PAGE_PAIR status indicates that a PAMT page pair is missing for one of the added SEPT pages. The host VMM may add it using TDH.PHYMEM.PAMT.ADD and retry the operation.

30 Note that this does not apply to the demoted page itself. TDH.MEM.PAGE.DEMOTE always requires a pair of PAMT pages to be provided (in R12 and R13) if the page is demoted to 512 4KB pages.

5.4.25.3.7. SEPT HPA Arguments on Input and Output

The table below shows the values of the SEPT HPA arguments in R9 – R11 when no error occurs (RAX returns TDX_SUCCESS).

Table 5.101: Meaning of TDH.MEM.PAGE.DEMOTE's SEPT HPA Arguments on Input and Output (No Error)

| Value | R9 – R11 on Input | R9 – R11 on Output |
|--|---|--|
| NULL_PA (-1) | No new SEPT page to add | Unmodified |
| Bits 62:0: Valid HPA, HKID bits are 0 Bit 63: 0 | Bits 62:0: HPA of new SEPT page to add Bit 63: Ignored | TDH.MEM.PAGE.DEMOTE terminated successfully and the new SEPT page has been added |
| Bits 62:0: Valid HPA, HKID bits are 0 Bit 63: 1 | | TDH.MEM.PAGE.DEMOTE terminated successfully and the new SEPT page has not been added |

5.4.25.3.8. Interaction with Non-Blocking Export

If the page state before the demote operation was MAPPED and a non-blocking export session is in the LIVE_EXPORT phase, TDH.MEM.PAGE.DEMOTE sets all the small pages' Dirty bits. Otherwise, it clears them.

5.4.25.3.9. Interruptibility

If the TD is not partitioned (i.e., it has been configured with no L2 VMs), and the TDX Module enumerates TDX_FEATURES0.ENHANCED_DEMOTE_INTERRUPTIBILITY as 1, TDH.MEM.PAGE.DEMOTE is not interruptible.

Else, TDH.MEM.PAGE.DEMOTE is interruptible but not resumable. If a pending interrupt is detected during operation, TDH.MEM.PAGE.DEMOTE returns with a TDX_INTERRUPTED_RESTARTABLE status in RAX. No demote operation is done and no output operands except RAX are modified.

In such cases, TDH.MEM.PAGE.DEMOTE should be invoked in a loop until it terminates successfully. The host VMM should be designed to avoid cases where interrupt storms prevent successful completion of TDH.MEM.PAGE.DEMOTE.

5.4.25.4. Operands Information

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

Table 5.102: TDH.MEM.PAGE.DEMOTE Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource Name | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|--------------------|---------------------|-------------------------------------|------------------|--------|---------------------|----------------------------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ¹⁸ |
| Explicit | RCX | GPA and Level | TD private page to split | Blob | None | Private | $2^{12+9*\text{level}}$ bytes | Exclusive | None | None | None |
| Explicit | RDX | HPA | TDR page | TDR | RW | Opaque | 4KB | Shared | Shared | Shared | None |
| Explicit | R8 | HPA | New L1 Secure EPT page | SEPT_PAGE | RW | Private | 4KB | Exclusive | Shared | Shared | Exclusive |
| Explicit | R9, R10, R11 | HPA | New L2 Secure EPT pages | SEPT_PAGE | RW | Private | 4KB | Exclusive | Shared | Shared | Exclusive |
| Explicit | R12, R13 | HPA | New PAMT pages ¹⁹ | PAMT_PAGE | RW | Opaque | 4KB | Exclusive | Shared | Shared | Exclusive |
| Implicit | N/A | HPA | HPA range to split ²⁰ | N/A | RW | Opaque | N/A | Exclusive ²¹ | None | None | None |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A | None |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared | N/A | N/A | None |
| Implicit | N/A | GPA | L1 Secure EPT Tree | N/A | RW | Private | N/A | Shared | N/A | N/A | None |
| Implicit | N/A | GPA | L1 Secure EPT entry | SEPT Entry | RW | Private | N/A | Exclusive | N/A | N/A | None |
| Implicit | N/A | GPA | L2 Secure EPT Trees | N/A | RW | Private | N/A | Shared(i) | N/A | N/A | None |

¹⁸ ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

¹⁹ Only if the TDX Module is configured to use Dynamic PAMT and the large page level is 1 (2MB).

²⁰ Only if the TDX Module is configured to use Dynamic PAMT and the large page level is 1 (2MB).

²¹ Concurrency is enforced by locking the parent **non-leaf** PAMT entry in the PAMT tree, which points to the new page.

| Explicit/ Implicit | Reg. | Ref Type | Resource Name | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------|-------------|--------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ¹⁸ |
| Implicit | N/A | GPA | L2 Secure EPT entries | SEPT Entry | RW | Private | N/A | Exclusive(i) | N/A | N/A | None |

5.4.25.5. Completion Status Codes

Table 5.103: TDH.MEM.PAGE.DEMOTE Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_EPT_ENTRY_STATE_INCORRECT | |
| TDX_EPT_WALK_FAILED | |
| TDX_GPA_RANGE_NOT_BLOCKED | |
| TDX_INTERRUPTED_RESTARTABLE | TDH.MEM.PAGE.DEMOTE's operation has been interrupted by an external event; it may be restarted (from its beginning) by calling it again. |
| TDX_L2_SEPT_PAGE_NOT_PROVIDED | |
| TDX_L2_SEPT_WALK_FAILED | |
| TDX_MISSING_PAMT_PAGE_PAIR | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_PAGE_NOT_FREE | |
| TDX_SUCCESS | TDH.MEM.PAGE.DEMOTE is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |
| TDX_TLB_TRACKING_NOT_DONE | |

5.4.26. TDH.MEM.PAGE.PROMOTE Leaf

TDH.MEM.PAGE.PROMOTE merges 512 consecutive 4KB pages into a single 2MB page, or 512 consecutive 2MB pages into a single 1GB page. The L1 and any L2 SEPT pages which mapped the 512 small pages are removed, and the non-leaf SEPT entries that mapped the whole large GPA range change to leaf entries.

- 5 If the TDX Module is configured to use Dynamic PAMT and the merged page level is 1 (2MB), the pair of PAMT pages that were associated with the 512 small pages is removed, and the non-leaf PAMT entry that was associated with the whole large HPA range changes to a leaf entry.

5.4.26.1. Input Operands

Table 5.104: TDH.MEM.PAGE.PROMOTE Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|---|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 23 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Versions 0 and 1 are supported. See enumeration details below. |
| | 63:24 | Reserved | Must be 0 |
| RCX | EPT mapping information: | | |
| | Bits | Name | Description |
| | 2:0 | Level | Level of the Secure EPT entry that will map the merged large page: either 1 (2MB) or 2 (1GB) (see 3.6.1) |
| | 11:3 | Reserved | Reserved: must be 0 |
| | 51:12 | GPA | Bits 51:12 of the guest physical address of the merged large page Depending on the level, the following least significant bits must be 0: Level 1 (2MB): Bits 20:12 Level 2 (1GB): Bits 29:12 |
| | 63:52 | Reserved | Reserved: must be 0 |
| RDX | TD handle and flags: | | |
| | Bits | Name | Description |
| | 0 | NO_TRACK | Large GPA range blocking and TLB tracking mode: 0: The merged large GPA range is checked for blocking and TLB tracking, and SEPT pages are removed. 1: The merged large GPA range is not checked for blocking and TLB tracking, and SEPT pages are converted to PT_TR pages. Enumeration: Support of NO_TRACK value of 1 is enumerated by TDX_FEATURES0.NON_BLOCKING_RESIZE (bit 35), readable using TDH.SYS.RD*. |
| | 11:1 | Reserved | Reserved: must be 0 |
| | 51:12 | TDR_HPA | Bits 51:12 of the host physical address of the parent TDR page (HKID bits must be 0) |

| Operand | Description | | |
|---------|-------------|----------|---------------------|
| | 63:52 | Reserved | Reserved: must be 0 |

5.4.26.2. Output Operands

Table 5.105: TDH.MEM.PAGE.PROMOTE Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| RCX | <p>If TDH.MEM.PAGE.PROMOTE succeeded:</p> <ul style="list-style-type: none"> If the specified value of NO_TRACK was 0, RCX returns the HPA of the removed L1 SEPT page (HKID bits are set to 0). If supported by the TDX Module has been configured for Dynamic PAMT, then bit 62 serves as a dynamic PAMT removal hint; it indicates that all entries in the PAMT page pair mapping the removed L1 SEPT page are free. Else (NO_TRACK was 1), RCX returns the HPA of the L1 SEPT page converted to PT_TR (HKID bits are set to 0). <p>In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESTARTABLE, RCX is unmodified.</p> <p>Else, RCX returns extended error information part 1.</p> <p>In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2</p> <p>The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page; it may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX.</p> <p>In other cases, RCX returns 0.</p> |
| RDX | <p>In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESTARTABLE, RDX is unmodified.</p> <p>Else, RDX returns extended error information part 2.</p> <p>In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2</p> <p>In other cases, RDX returns 0.</p> |
| R9 | <p>If TDH.MEM.PAGE.PROMOTE version is 1 or higher:</p> <ul style="list-style-type: none"> If the specified value of NO_TRACK was 0 and L2 VM #1's L2 SEPT page has been removed, R9 returns the HPA of that L2 SEPT page (HKID bits are set to 0). If the TDX Module has been configured for Dynamic PAMT, then bit 62 serves as a dynamic PAMT removal hint; it indicates that all entries in the PAMT page pair mapping the removed L2 #1 SEPT page are free. Else, if NO_TRACK was 1 and L2 VM #1's L2 SEPT page has been converted to PT_TR, R9 returns the HPA of that L2 SEPT page (HKID bits are set to 0). Else, R9 returns NULL_PA (-1). <p>Else (TDH.MEM.PAGE.PROMOTE version is 0), R9 is unmodified.</p> |

| Operand | Description |
|---------|---|
| R10 | <p>If TDH.MEM.PAGE.PROMOTE version is 1 or higher:</p> <ul style="list-style-type: none"> If the specified value of NO_TRACK was 0 and L2 VM #2's L2 SEPT page has been removed, R10 returns the HPA of that L2 SEPT page (HKID bits are set to 0). <p>If the TDX Module has been configured for Dynamic PAMT, then bit 62 serves as a dynamic PAMT removal hint; it indicates that all entries in the PAMT page pair mapping the removed L2 #2 SEPT page are free.</p> <ul style="list-style-type: none"> Else, if NO_TRACK was 1 and L2 VM #2's L2 SEPT page has been converted to PT_TR, R10 returns the HPA of that L2 SEPT page (HKID bits are set to 0). Else, R10 returns NULL_PA (-1). <p>Else (TDH.MEM.PAGE.PROMOTE version is 0), R10 is unmodified.</p> |
| R11 | <p>If TDH.MEM.PAGE.PROMOTE version is 1 or higher:</p> <ul style="list-style-type: none"> If the specified value of NO_TRACK was 0 and L2 VM #3's L2 SEPT page has been removed, R11 returns the HPA of that L2 SEPT page (HKID bits are set to 0). <p>If the TDX Module has been configured for Dynamic PAMT, then bit 62 serves as a dynamic PAMT removal hint; it indicates that all entries in the PAMT page pair mapping the removed L2 #3 SEPT page are free.</p> <ul style="list-style-type: none"> Else, if NO_TRACK was 1 and L2 VM #3's L2 SEPT page has been converted to PT_TR, R11 returns the HPA of that L2 SEPT page (HKID bits are set to 0). Else, R11 returns NULL_PA (-1). <p>Else (TDH.MEM.PAGE.PROMOTE version is 0), R11 is unmodified.</p> |
| R12 | <p>If the TDX Module is configured to use Dynamic PAMT and the merged page level is 1 (2MB), and TDH.MEM.PAGE.PROMOTE succeeded, R12 returns the HPA of the removed PAMT page for 4KB pages 0 through 255 (HKID bits are set to 0).</p> <p>Else, R12 is unmodified.</p> |
| R13 | <p>If the TDX Module is configured to use Dynamic PAMT and the merged page level is 1 (2MB), and TDH.MEM.PAGE.PROMOTE succeeded, R13 returns the HPA of the removed PAMT page for 4KB pages 256 through 511 (HKID bits are set to 0).</p> <p>Else, R13 is unmodified.</p> |
| Other | Unmodified |

5.4.26.3. Leaf Function Description

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

5 5.4.26.3.1. Overview

TDH.MEM.PAGE.PROMOTE merges 512 private pages, which are consecutive both in the HPA space and in the GPA space. The L1 SEPT page and all existing L2 SEPT pages at the requested GPA and level are removed.

All merged private pages must have the same Secure EPT leaf entry attributes and state, which must be either MAPPED or PENDING. All merged private pages must have the same set of L2 mappings and L2 attributes.

10 5.4.26.3.2. Enumeration

Availability of TDH.MEM.PAGE.PROMOTE version 1 is enumerated by TDX_FEATURES0.TD_PARTITIONING (bit 7), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.MEM.PAGE.PROMOTE with a version number higher than 0 returns a TDX_OPERAND_INVALID(RAX) status.

Support of NO_TRACK value of 1 is enumerated by TDX_FEATURES0.NON_BLOCKING_RESIZE (bit 35).

15 Support of dynamic PAMT is enumerated by TDX_FEATURES0.DYNAMIC_PAMT (bit 36).

5.4.26.3.3. Blocking and TLB Tracking and the NO_TRACK Input Flag

If NO_TRACK is supported and its value is 1, the promoted GPA range is not checked for blocking and TLB tracking. The SEPT pages mapping that range are not removed; instead, they are converted to PT_TR pages which can later be either tracked and reclaimed or used as new SEPT pages for TDH.MEM.PAGE.DEMOTE for the current TD.

5 Else (NO_TRACK is 0), if the TD may be running, the promoted GPA range must be blocked and TLB tracked. Else (e.g., TDH.MR.FINALIZE has not yet been executed, or the TD has been paused for export), no blocking and tracking is required.

5.4.26.3.4. Dynamic PAMT

10 If the TDX Module is configured to use Dynamic PAMT and the merged page level is 1 (2MB), the pair of PAMT pages that were associated with the 512 small pages is removed, and the non-leaf PAMT entry that was associated with the whole large HPA range changes to a leaf entry.

If the TDX Module is configured for dynamic PAMT, bit 62 of RCX, R9, R10 and R11 are returned as hints indicating that all entries in the PAMT pages mapping the respective removed SEPT pages are free, and they can be removed using TDH.PHYMEM.PAMT.REMOVE.

15 **Note:** If the same PAMT pages map more than one of the removed SEPT pages, bit 62 may indicate so in only one of RCX, R9, R10 and R11.

5.4.26.3.5. Interruptibility

TDH.MEM.PAGE.PROMOTE is interruptible but not resumable. If a pending interrupt is detected during operation, TDH.MEM.PAGE.PROMOTE returns with a TDX_INTERRUPTED_RESTARTABLE status in RAX. No promote operation is done and no output operands except RAX are modified.

20 In such cases, TDH.MEM.PAGE.PROMOTE should be invoked in a loop until it terminates successfully. The host VMM should be designed to avoid cases where interrupt storms prevent successful completion of TDH.MEM.PAGE.PROMOTE.

5.4.26.3.6. Removed Page Initialization

On platforms which do not use ACT, after the SEPT pages have been removed, the host VMM should initialize their content before they are reused as non-private pages, as described in the [Base Spec].

5.4.26.3.7. SEPT Page HPA Inputs and Outputs

25 The table below shows the values of the SEPT HPA output arguments in RCX and R9 – R11 when version 1 or higher is selected and no error occurs (RAX returns TDX_SUCCESS).

Table 5.106: Meaning of TDH.MEM.PAGE.PROMOTE's SEPT HPA Arguments on Output (No Error)

| Value | RCX on Output | R9 – R11 on Output |
|--------------|--|--|
| NULL_PA (-1) | N/A | No removed or converted SEPT page for this L2 VM |
| Valid HPA | L1 SEPT page HPA If NO_TRACK value is 0, the page is removed. If supported and Dynamic PAMT is used, bit 62 is a PAMT page pair removal hint. Else (NO_TRACK is supported and its value is 1), the page is not removed; it is converted to PT_TR. | L2 SEPT page HPA If NO_TRACK value is 0, the page is removed. If supported and Dynamic PAMT is used, bit 62 is a PAMT page pair removal hint. Else (NO_TRACK is supported and its value is 1), the page is not removed; it is converted to PT_TR. |

30 5.4.26.4. Operands Information

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

Table 5.107: TDH.MEM.PAGE.PROMOTE Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------|---------------------|---|------------------|--------|---------------------|----------------------------------|----------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ²² |
| Explicit | RCX | GPA and Level | Removed Secure EPT page | SEPT_PAGE | R | Private | 2 ^{12+9*Level} Bytes | Exclusive | None | None | Exclusive |
| Explicit | RDX | HPA | TDR page | TDR | RW | Opaque | 4KB | Shared | Shared | Shared | None |
| Implicit | N/A | HPA | Merged HPA range | Blob | None | Private | N/A | Exclusive | None | None | None |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A | None |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared | N/A | N/A | None |
| Implicit | N/A | GPA | L1 Secure EPT Tree | N/A | RW | Private | N/A | Exclusive(h) | N/A | N/A | None |
| Implicit | N/A | GPA | Large page L1 Secure EPT entry | SEPT Entry | RW | Private | N/A | Exclusive | N/A | N/A | None |
| Implicit | N/A | GPA | Small pages L1 Secure EPT entries | SEPT Entry | RW | Private | N/A | Exclusive | N/A | N/A | None |
| Implicit | N/A | GPA | L2 Secure EPT Trees | N/A | RW | Private | N/A | Shared(i) | N/A | N/A | None |
| Implicit | N/A | GPA | L2 Large page Secure EPT entry | SEPT Entry | RW | Private | N/A | Exclusive(i) | N/A | N/A | None |
| Implicit | N/A | GPA | L2 Small pages Secure EPT entries | SEPT Entry | RW | Private | N/A | Exclusive(i) | N/A | N/A | None |
| Implicit | N/A | HPA | Removed PAMT pages ²³ | PAMT_PAGE | RW | Opaque | 4KB | Exclusive(i) ²⁴ | None | None | Exclusive |

5.4.26.5. Completion Status Codes

Table 5.108: TDH.MEM.PAGE.PROMOTE Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------------------|-------------|
| TDX_EPT_ENTRY_STATE_INCORRECT | |
| TDX_EPT_INVALID_PROMOTE_CONDITIONS | |
| TDX_EPT_WALK_FAILED | |

²² ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

²³ Applicable only if the TDX Module is configured to use dynamic PAMT and the merged page level is 1 (2MB).

²⁴ Concurrency is enforced by locking the parent leaf PAMT entry, which maps the new page.

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_INTERRUPTED_RESTARTABLE | TDH.MEM.PAGE.PROMOTE's operation has been interrupted by an external event; it may be restarted (from its beginning) by calling it again. |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.MEM.PAGE.PROMOTE is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |
| TDX_TLB_TRACKING_NOT_DONE | |

5.4.27. TDH.MEM.PAGE.RELOCATE Leaf

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

5.4.27.1. Input Operands

Table 5.109: TDH.MEM.PAGE.RELOCATE Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 5 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | EPT mapping information: | | |
| | Bits | Name | Description |
| | 2:0 | Level | The level of the Secure EPT entry that maps the private page to be relocated, must be 0 (i.e., 4KB) (see 3.6.1). |
| | 11:3 | Reserved | Reserved: must be 0 |
| | 51:12 | GPA | Bits 51:12 of the guest physical address of the private page to be relocated |
| | 63:52 | Reserved | Reserved: must be 0 |
| RDX | Host physical address of the parent TDR page (HKID bits must be 0) | | |
| R8 | Host physical address of the relocated page target (HKID bits must be 0) | | |

5

5.4.27.2. Output Operands

Table 5.110: TDH.MEM.PAGE.RELOCATE Output Operands Definition

| Operand | Description |
|---------|---|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| RCX | <p>If TDH.MEM.PAGE.RELOCATE succeeded:</p> <ul style="list-style-type: none"> RCX returns the HPA of the old physical page that has been removed (HKID bits are set to 0). If the TDX Module has been configured for Dynamic PAMT, then bit 62 serves as a dynamic PAMT removal hint; it indicates that all entries in the PAMT page pair mapping the removed old physical page are free. <p>In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2</p> <p>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.</p> <p>In other cases, RCX returns 0.</p> |
| RDX | Extended error information part 2 |

| Operand | Description |
|---------|--|
| | In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2 In other cases, RDX returns 0. |
| Other | Unmodified |

5.4.27.3. Leaf Function Description

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

5 **5.4.27.3.1. Overview**

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.

5.4.27.3.2. Blocking and TLB Tracking

10 If the TD may be running, the relocated page must be blocked and TLB tracked. Else (e.g., TDH.MR.FINALIZE has not yet been executed, or the TD has been paused for export), no blocking and tracking is required.

5.4.27.3.3. Dynamic PAMT

15 If the TDX Module is configured for dynamic PAMT, the PAMT hierarchy can be built on demand. A TDH_MISSING_PAMT_PAGE_PAIR status indicates that a PAMT page pair is missing. The host VMM may add it using TDH.PHYMEM.PAMT.ADD and retry the operation. In addition, RCX[62] is returned as a hint indicating that all entries in the PAMT page pair mapping the removed old physical page are free, and they can be removed using TDH.PHYMEM.PAMT.REMOVE.

5.4.27.3.4. Removed Page Initialization

20 On platforms which do not use ACT, after the page has been relocated, the host VMM should initialize its content before it is reused as a non-private page, as described in the [Base Spec].

5.4.27.4. Operands Information

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

Table 5.111: TDH.MEM.PAGE.RELOCATE Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------|---------------------|-------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ²⁵ |
| Explicit | RCX | GPA and Level | TD private page | Blob | R | Private | 4KB | Exclusive | None | None | Exclusive |
| Explicit | RDX | HPA | TDR page | TDR | R | Opaque | 4KB | Shared | Shared | Shared | None |
| Explicit | R8 | HPA | Target physical page | Blob | RW | Private | 4KB | Exclusive | Shared | Shared | Exclusive |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A | None |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared | N/A | N/A | None |

²⁵ ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------|-------------|--------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ²⁵ |
| Implicit | N/A | GPA | L1 Secure EPT tree | N/A | RW | Private | N/A | Shared | N/A | N/A | None |
| Implicit | N/A | GPA | L1 Secure EPT entry | SEPT Entry | RW | Private | N/A | Exclusive | N/A | N/A | None |
| Implicit | N/A | GPA | L2 Secure EPT trees | N/A | RW | Private | N/A | Shared(i) | N/A | N/A | None |
| Implicit | N/A | GPA | L2 Secure EPT entries | SEPT Entry | RW | Private | N/A | Exclusive(i) | N/A | N/A | None |

5.4.27.5. Completion Status Codes

Table 5.112: 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_MISSING_PAMT_PAGE_PAIR | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.MEM.PAGE.RELOCATE is successful |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |
| TDX_TLB_TRACKING_NOT_DONE | |

5.4.28. TDH.MEM.PAGE.REMOVE Leaf

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

5.4.28.1. Input Operands**Table 5.113: TDH.MEM.PAGE.REMOVE Input Operands Definition**

| Operand | Description | | |
|---------|--|----------------|---|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 29 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version If the TDX Module supports ACT, version may be 0 or 1. Else, version must be 0. See enumeration details below. |
| | 63:24 | Reserved | Must be 0 |
| RCX | EPT mapping information: | | |
| | Bits | Name | Description |
| | 2:0 | Level | Level of the Secure EPT entry that maps the private page to be removed: either 0 (4KB), 1 (2MB) or 2 (1GB) – see 3.6.1. |
| | 11:3 | Reserved | Reserved: must be 0 |
| | 51:12 | GPA | Bits 51:12 of the guest physical address of the private page to be removed |
| | 63:52 | Reserved | Reserved: must be 0 |
| RDX | Host physical address of the parent TDR page (HKID bits must be 0) | | |

5

5.4.28.2. Output Operands**Table 5.114: TDH.MEM.PAGE.REMOVE Output Operands Definition**

| Operand | Description |
|---------|---|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| RCX | <p>If TDH.MEM.PAGE.REMOVE succeeded:</p> <ul style="list-style-type: none"> RCX returns the HPA of the removed page (HKID bits are set to 0). If the TDX Module has been configured for Dynamic PAMT, then bit 62 serves as a dynamic PAMT removal hint; it indicates that all entries in the PAMT page pair mapping the removed page are free. This only applies to 4KB pages. <p>In case of EPT walk error, RCX returns the Secure EPT entry architectural content where the error was detected – see 3.6.2. The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page and may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX.</p> <p>In other cases, RCX returns 0.</p> |
| RDX | <p>Extended error information part 2</p> <p>In case of EPT walk error, RDX returns the Secure EPT entry level and state where the error was detected – see 3.6.2</p> |

| Operand | Description |
|---------|--------------------------------|
| | In other cases, RDX returns 0. |
| Other | Unmodified |

5.4.28.3. Leaf Function Description

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

5 **5.4.28.3.1. Overview**

TDH.MEM.PAGE.REMOVE removes a 4KB, 2MB or 1GB private page from the TD’s Secure EPT tree (marks the SEPT entry as FREE). If the page is mapped in any L2 Secure EPT, the applicable L2 SEPT entries are marked as L2_FREE. On successful operation, TDH.MEM.PAGE.REMOVE marks the physical page as free in PAMT.

5.4.28.3.2. Enumeration

10 Availability of TDH.MEM.PAGE.REMOVE version 1 is enumerated by TDX_FEATURES0.ACT (bit 14), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.MEM.PAGE.REMOVE with version higher than 0 returns a TDX_OPERAND_INVALID(RAX) status.

5.4.28.3.3. Blocking and TLB Tracking

15 If the TD may be running, the removed page must be blocked and TLB tracked. Else (e.g., TDH.MR.FINALIZE has not yet been executed, or the TD has been paused for export), no blocking and tracking is required.

5.4.28.3.4. Dynamic PAMT

If the TDX Module is configured for dynamic PAMT and the removed page size is 4KB, RCX[62] is returned as a hint indicating that all entries in the PAMT page pair mapping the removed page are free, and they can be removed using TDH.PHYMEM.PAMT.REMOVE.

20 **5.4.28.3.5. Removed Page Initialization**

On platforms which do not use ACT, after the page has been removed, the host VMM should initialize its content before it is reused as non-private pages, as described in the [Base Spec].

5.4.28.3.6. Interruptibility

25 If called with version higher than 0, TDH.MEM.PAGE.REMOVE is interruptible and resumable. If a pending interrupt is detected during operation, TDH.MEM.PAGE.REMOVE returns with a TDX_INTERRUPTED_RESUMABLE status in RAX. No output operands except RAX are modified.

5.4.28.4. Operands Information

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

30 **Table 5.115: TDH.MEM.PAGE.REMOVE Operands Information Definition**

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------|---------------------|-----------------|------------------|--------|---------------------|----------------------------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ²⁶ |
| Explicit | RCX | GPA and Level | TD private page | Blob | R | Private | 2 ^{12+9*Level} Bytes | Exclusive | None | None | Exclusive |

²⁶ ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------|-------------|--------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ²⁶ |
| Explicit | RDX | HPA | TDR page | TDR | RW | Opaque | 4KB | Shared | Shared | Shared | None |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A | None |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared | N/A | N/A | None |
| Implicit | N/A | GPA | L1 Secure EPT tree | N/A | RW | Private | N/A | Shared | N/A | N/A | None |
| Implicit | N/A | GPA | L1 Secure EPT entry | SEPT Entry | RW | Private | N/A | Exclusive | N/A | N/A | None |
| Implicit | N/A | GPA | L2 Secure EPT trees | N/A | RW | Private | N/A | Shared(i) | N/A | N/A | None |
| Implicit | N/A | GPA | L2 Secure EPT entries | SEPT Entry | RW | Private | N/A | Exclusive(i) | N/A | N/A | None |

5.4.28.5. Completion Status Codes

Table 5.116: TDH.PHYMEM.PAGE.REMOVE Completion Status Codes (Returned in RAX) Definition

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

5.4.29. TDH.MEM.RANGE.BLOCK Leaf

Unreleased Feature: Some of the text in this section is related to Non-Blocking Export, a feature which has not been released yet at the time of writing of this document. Details related to that feature serve as a preview and are subject to change.

- 5 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.

5.4.29.1. Input Operands

Table 5.117: TDH.MEM.RANGE.BLOCK Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 7 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | EPT mapping information: | | |
| | Bits | Name | Description |
| | 2:0 | Level | Level of the Secure EPT entry that maps the GPA range to be blocked – see 3.6.1 Level must be 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) | | |

10 5.4.29.2. Output Operands

Table 5.118: TDH.MEM.RANGE.BLOCK Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |

| Operand | Description |
|---------|--|
| RCX | <p>Extended error information part 1</p> <p>In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2</p> <p>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.</p> <p>In other cases, RCX returns 0.</p> |
| RDX | <p>Extended error information part 2</p> <p>In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2</p> <p>In other cases, RDX returns 0.</p> |
| Other | Unmodified |

5.4.29.3. Leaf Function Description

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

5 5.4.29.3.1. Overview

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.

5.4.29.3.2. Interaction with TD Migration

10 If the TDX module is configured for write-blocking based export, TDH.MEM.RANGE.BLOCK cannot be used to block a page that has been exported. To block an exported page, the host VMM must first cancel the page export by calling TDH.EXPORT.MEM with a CANCEL operation for that page.

15 Else (the TDX module is configured for non-blocking export), TDH.MEM.RANGE.BLOCK cannot block any memory while an export session is in the blackout phase, i.e., after TDH.EXPORT.PAUSE is called and before either TDH.EXPORT.TRACK(DONE) or TDH.EXPORT.ABORT is called.

5.4.29.3.3. Interaction with TDX Connect

If the TD is configured with TDX Connect enabled, the following conditions apply:

- If the GPA range is mapped by a leaf SEPT entry, TDH.MEM.RANGE.BLOCK of MAPPED pages is only allowed if no TDIs are attached. TDH.MEM.RANGE.BLOCK of PENDING pages is allowed unconditionally.
- 20 • Else (the GPA range is mapped by a non-leaf SEPT entry), TDH.MEM.RANGE.BLOCK is only allowed if either no TDIs are attached, or if all 512 entries of the SEPT child page are FREE.

5.4.29.4. Operands Information

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

Table 5.119: TDH.MEM.RANGE.BLOCK Operands Information Definition

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

5.4.29.5. Completion Status Codes

Table 5.120: TDH.MEM.RANGE.BLOCK Completion Status Codes (Returned in RAX) Definition

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

5.4.30. 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.

5.4.30.1. Input Operands**Table 5.121: TDH.MEM.RANGE.UNBLOCK Input Operands Definition**

| Operand | Description | | |
|---------|--|---------------------|---|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 39 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | EPT mapping information: | | |
| | Bits | Name | Description |
| | 2:0 | Level | Level of the Secure EPT entry that maps the GPA range to be unblocked – see 3.6.1 Level must be 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) | | |

5.4.30.2. Output Operands**Table 5.122: TDH.MEM.RANGE.UNBLOCK Output Operands Definition**

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |

| Operand | Description |
|---------|--|
| RCX | <p>Extended error information part 1</p> <p>In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2</p> <p>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.</p> <p>In other cases, RCX returns 0.</p> |
| RDX | <p>Extended error information part 2</p> <p>In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2</p> <p>In other cases, RDX returns 0.</p> |
| Other | Unmodified |

5.4.30.3. Leaf Function Description

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

5 **5.4.30.3.1. Overview**

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).

5.4.30.3.2. Blocking and TLB Tracking

10 If the TD may be running, the unblocked GPA range must be blocked and TLB tracked. Else (e.g., TDH.MR.FINALIZE has not yet been executed, or the TD has been paused for export), no blocking and tracking is required.

5.4.30.4. Operands Information

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

15 **Table 5.123: TDH.MEM.RANGE.UNBLOCK Operands Information Definition**

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

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|--------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Implicit | N/A | GPA | L2 Secure EPT tree | N/A | RW | Private | N/A | Exclusive(i) | N/A | N/A |
| Implicit | N/A | GPA | L2 Secure EPT entries | SEPT Entry | RW | Private | N/A | Exclusive(i) | N/A | N/A |

5.4.30.5. Completion Status Codes

Table 5.124: TDH.MEM.RANGE.UNBLOCK Completion Status Codes (Returned in RAX) Definition

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

5.4.31. TDH.MEM.RD Leaf

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

5.4.31.1. Input Operands

Table 5.125: TDH.MEM.RD Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 12 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | The guest physical address of a naturally aligned 8-byte chunk of a guest TD private page | | |
| RDX | Host physical address of the parent TDR page (HKID bits must be 0) | | |

5

5.4.31.2. Output Operands

Table 5.126: TDH.MEM.RD Output Operands Definition

| Operand | Description |
|---------|---|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| RCX | Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page and may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. In other cases, RCX returns 0. |
| RDX | Extended error information part 2 In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2 In other cases, RDX returns 0. |
| R8 | Content of the memory chunk In case of an error, as indicated by RAX, R8 returns 0 |
| Other | Unmodified |

5.4.31.3. Leaf Function Description

10 **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.

5.4.31.4. Operands Information

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

Table 5.127: TDH.MEM.RD Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|----------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | GPA | TD private memory | Blob | R | Private | 8B | None | None | None |
| Explicit | RDX | HPA | TDR page | TDR | R | Opaque | 4KB | Shared | Shared | Shared |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | R | Opaque | N/A | Shared | N/A | N/A |
| Implicit | N/A | GPA | Secure EPT tree | N/A | R | Private | N/A | Shared | N/A | N/A |
| Implicit | N/A | GPA | Secure EPT entry | SEPT Entry | R | Private | N/A | Exclusive | N/A | N/A |

5

5.4.31.5. Completion Status Codes

Table 5.128: TDH.MEM.RD Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|-------------|
| TDX_EPT_ENTRY_NOT_PRESENT | |
| TDX_EPT_ENTRY_STATE_INCORRECT | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TD_NON_DEBUG | |
| TDX_TDCS_NOT_ALLOCATED | |

5.4.32. TDH.MEM.SCAN.COMP/RANGE – Common

Unreleased Feature: TDH.MEM.SCAN.* is a set of interface functions that will be provided with Non-Blocking Export, a feature which has not been released yet at the time of writing of this document. Details related to that feature serve as a preview and are subject to change.

5 This section contains definitions that are common to TDH.MEM.SCAN.COMP and TDH.MEM.SCAN.RANGE.

5.4.32.1. GPA List-of-Lists Processing

5.4.32.1.1. Overview

GPA list-of-lists is defined in 3.13.2. It enables the host VMM to specify up to 262144 (i.e., 512²) GPA list entries into which the scan results are written. The figure below shows how it is used by TDH.MEM.SCAN.COMP and TDH.MEM.SCAN.RANGE.

10

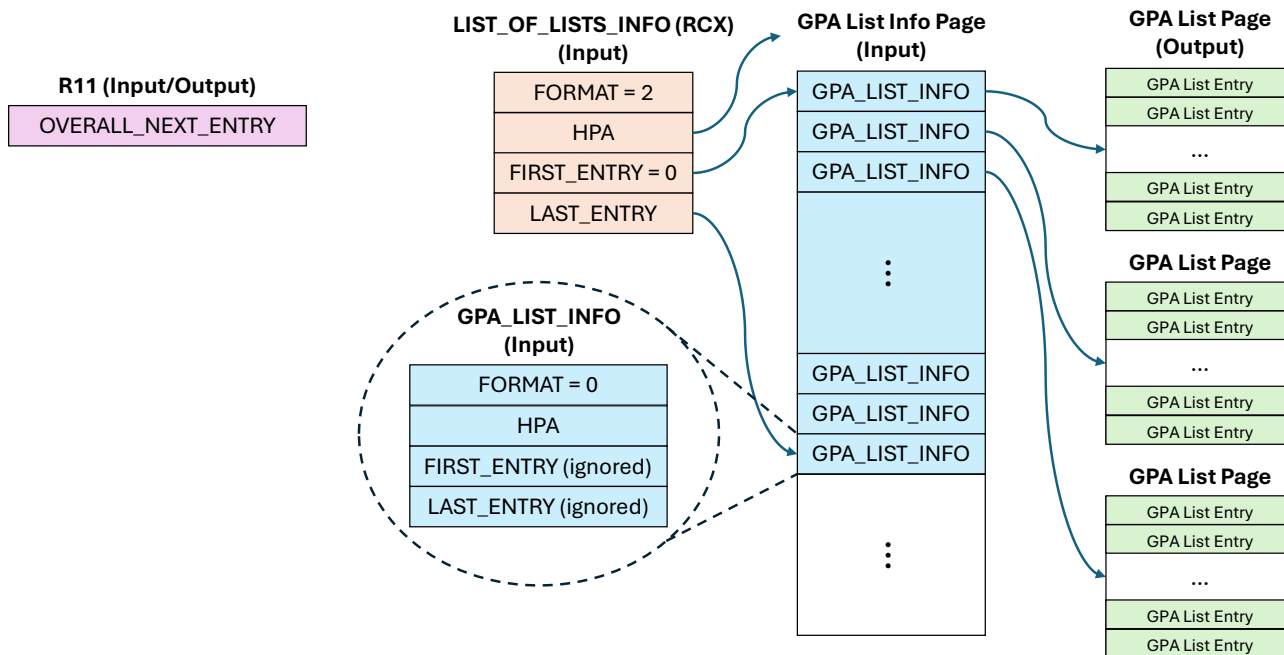


Figure 5.3: GPA List of Lists as Used by TDH.MEM.SCAN.COMP/RANGE

The host VMM can allocate up to 512 4KB GPA List pages, referenced by a single 4KB GPA List Info Page. Each GPA List page contains 512 entries.

15 5.4.32.1.2. LIST_OF_LISTS_INFO (Input)

The host VMM sets the LIST_OF_LISTS_INFO input (in RCX) as follows:

- HPA contains bits 51:12 of the GPA List Info page shared HPA (including HKID).
- FIRST_ENTRY must be 0.
- LAST_ENTRY contains the index of the last valid entry in the GPA List Info page.
- FORMAT must be LIST_OF_LISTS (2).

20

LIST_OF_LISTS_INFO is not modified by TDH.MEM.SCAN.COMP/RANGE.

5.4.32.1.3. GPA_LIST_INFO Entries (Input)

The host VMM sets each GPA_LIST_INFO entry in the GPA List Info page as follows:

- HPA contains bits 51:12 of the GPA List page shared HPA (including HKID).
- FIRST_ENTRY and LAST_ENTRY are ignored by TDH.MEM.SCAN.COMP/RANGE.
- FORMAT must be GPA_ONLY (0).

25

GPA_LIST_INFO entries are not modified by TDH.MEM.SCAN.COMP/RANGE.

5.4.32.1.4. GPA List Pages (Output)

TDH.MEM.SCAN.COMP/RANGE writes entries in the GPA list pages, starting with the next available entry.

5.4.32.1.5. OVERALL_NEXT_ENTRY (Input/Output)

OVERALL_NEXT_ENTRY, provided as an input in R11, specifies the next overall (between 0 and 262143) index of the GPA List entry to be written by TDH.MEM.SCAN.COMP/RANGE. On output, R11 is updated by TDH.MEM.SCAN.COMP/RANGE to the index of the next available entry. Thus, the output value can be used as-is as the input for the next call to TDH.MEM.SCAN.COMP/RANGE, to continue filling the GPA list-of-lists. See below for details.

Given OVERALL_NEXT_ENTRY, the following can be calculated:

- The next available GPA_LIST_INFO entry index in the GPA List Info page, which is also the index of the GPA List Page that is being filled, can be calculated as $OVERALL_NEXT_ENTRY / 512$.
- All previous GPA List Pages are already filled with 512 GPA List Entries.
- In the GPA List Page that being filled, the next available GPA List Entry index can be calculated as $OVERALL_NEXT_ENTRY \% 512$.

On input, OVERALL_NEXT_ENTRY's value must be between 0 and $512 * (LIST_OF_LISTS_INFO.LAST_ENTRY + 1) - 1$.

Note: On a list full condition (status is TDX_INTERRUPTED_LIST_FULL), OVERALL_NEXT_ENTRY returns with a value that is higher than the last available entry.

5.4.32.1.6. Providing an Empty GPA List-of-Lists

To start with a new, empty GPA list-of-lists, the host VMM sets OVERALL_NEXT_ENTRY to 0.

This typically needs to be done after a previous GPA list-of-lists was completely filled, i.e., the previous call to TDH.MEM.SCAN.COMP/RANGE returned with and TDX_INTERRUPTED_LIST_FULL status (see below).

5.4.32.1.7. Continuing a Partially Filled GPA List-of-Lists

TDH.MEM.SCAN.COMP/RANGE may be called to continue writing to a partially filled GPA list-of-lists, with the input LIST_OF_LISTS_INFO and OVERALL_NEXT_ENTRY values as returned by the previous call to TDH.MEM.SCAN.COMP/RANGE.

Specifically, this can be done when resuming TDH.MEM.SCAN.COMP/RANGE after an interruption, i.e., the returned status was TDX_INTERRUPTED_RESUMABLE or TDX_INTERRUPTED_BUSY.

5.4.32.1.8. List Full Interruption and Resumption

A list full condition happens when all the available GPA List entries have been written by TDH.MEM.SCAN.COMP/RANGE, and there is a need to write another entry. In this case, TDH.MEM.SCAN.COMP/RANGE returns with a TDX_INTERRUPTED_LIST_FULL status. The host VMM is expected to provide a GPA list-of-lists with available entries and resume TDH.MEM.SCAN.COMP/RANGE (with the RESUME flag set to 1).

5.4.33. TDH.MEM.SCAN.COMP Leaf

Unreleased Feature: TDH.MEM.SCAN.* is a set of interface functions that will be provided with Non-Blocking Export, a feature which has not been released yet at the time of writing of this document. Details related to that feature serve as a preview and are subject to change.

- 5 Do a comprehensive scan of the TD's private GPA space and perform the requested operation.

5.4.33.1. Input Operands

Table 5.129: TDH.MEM.SCAN.COMP Input Operands Definition

| Operand | Name | Description | | |
|---------|--------------------|--|----------------|---|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 93 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | | 63:24 | Reserved | Must be 0 |
| RCX | LIST_OF_LISTS_INFO | Information for GPA list-of-lists in shared memory. For details, see the description below, in 5.4.32.1 and in 3.13.2. <ul style="list-style-type: none"> FIRST_ENTRY must be 0. LAST_ENTRY contains the index of the last valid entry in the GPA List Info page. FORMAT must be LIST_OF_LISTS (2). | | |
| RDX | TDR | HPA of the source TD's TDR page (HKID bits must be 0) | | |
| R8 | CONTROLS | Controls fields: | | |
| | | Bits | Field | Description |
| | | 7:0 | OPERATION | Identifies the requested operation – see the table below Enumeration: See below for details. |
| | | 15:8 | QUALIFIER | Additional qualification of the requested operation |
| | | 31:16 | RESERVED | Reserved, must be 0 |
| | | 47:32 | CONTEXT_ID | Identifies the context of this invocation CONTEXT_ID must be between 0 and NUM_MEM_SCAN_CONTEXTS - 1. See enumeration details below. Each concurrent invocation of TDH.MEM.SCAN.COMP must be provided with a unique CONTEXT_ID. |
| 55:48 | RANGE_ID | Identifies the GPA range (previously configured by TDH.SCAN.CONFIG) for this invocation RANGE_ID must be between 0 and NUM_RANGES - 1 (NUM_RANGES is configured by TDH.SCAN.CONFIG). | | |

| Operand | Name | Description |
|---------|--------------------|---|
| | | 62:56 RESERVED Reserved, must be 0 |
| | | 63 RESUME 0: This is a new invocation of TDH.MEM.SCAN.COMP 1: This is resumption of a previously interrupted TDH.MEM.SCAN.COMP operation |
| R11 | OVERALL_NEXT_ENTRY | The next overall index of the GPA List entry to be written by TDH.MEM.SCAN.COMP. For details, see the description below and in 5.4.32.1. Must be between 0 and $512 * (\text{LIST_OF_LISTS_INFO.LAST_ENTRY} + 1) - 1$. |

5.4.33.2. Output Operands

Table 5.130: TDH.MEM.SCAN.COMP Output Operands Definition

| Operand | Name | Description |
|---------|--------------------|---|
| RAX | STATUS | SEAMCALL instruction return code, see 5.4.1 |
| R11 | OVERALL_NEXT_ENTRY | The overall index of the next available GPA List entry. For details, see the description below and in 5.4.32.1. |
| Other | | Unmodified |

5.4.33.3. Leaf Function Description

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

5.4.33.3.1. Overview

TDH.MEM.SCAN.COMP scans the whole TD's GPA range and performs the requested operation. Most operations fill page information in the returned GPA lists.

5.4.33.3.2. Enumeration

TDH.MEM.SCAN.COMP is supported if any of its operations are enumerated as available. See the operation modes table below for details. If not supported, calling TDH.MEM.SCAN.COMP returns a TDX_OPERAND_INVALID(RAX) status.

The supported number of memory scan contexts is enumerated by NUM_MEM_SCAN_CONTEXTS.

5.4.33.3.3. Operation Modes and Qualifiers

TDH.MEM.SCAN.COMP may support multiple operation modes as detailed below. Operation modes may have qualifiers.

Table 5.131: TDH.MEM.SCAN.COMP Operation Modes

| OPERATION Value | Name | Description |
|-----------------|--------|---|
| 1 | DCHECK | Do a comprehensive scan of the TD's GPA range for a final list of export candidates. DCHECK is used in the final round of export, after the TD has been paused and as a prerequisite to exporting the up-to-date memory image before calling TDH.EXPORT.TRACK(DONE). DCHECK is only available if the TDX module is configured for non-blocking export. An export session must be in progress, and the TD must have been paused by TDH.EXPORT.PAUSE. |

| OPERATION Value | Name | Description |
|-----------------|----------|--|
| | | <p>Enumeration: DCHECK support is enumerated by TDX_FEATURES0.NON_BLOCKING_EXPORT (bit 41), readable by TDH.SYS.RD* (see 3.3.3.1).</p> <p>DCHECK has two sub modes, selected by the QUALIFIER input:</p> <p>0: EXPORT: Scan and return a GPA list of memory pages that either have not been exported or need to be re-exported.</p> <p>1: REEXPORT: Scan and return a GPA list of memory pages that need to be re-exported. Pages that have not been exported are still counted by UNEXPORT_COUNT. This mode is useful for implementing post-copy (if supported).</p> <p>Other: Reserved</p> |
| Other | RESERVED | Reserved |

5.4.33.3.4. Comprehensive Scan Details

The whole private GPA space is scanned by multiple, possibly concurrent, invocations of TDH.MEM.SCAN.COMP. While a comprehensive scan is in progress, no changes are allowed to the TD's GPA space structure.

5 5.4.33.3.4.1. Multiple GPA Ranges

To allow optimization for NUMA configurations, multiple GPA ranges may be configured by the host VMM using TDH.MEM.SCAN.CONFIG.

10 The RANGE_ID input parameter identifies the GPA range for the current invocation. RANGE_ID must be between 0 and NUM_RANGES - 1 (NUM_RANGES is configured by TDH.SCAN.CONFIG). The host VMM can invoke concurrent scanning TDH.MEM.SCAN.COMP threads of each GPA range, typically executed on LPs that are close (in the NUMA sense) to the memory range being scanned.

5.4.33.3.4.2. Scan Concurrency within a GPA Range

Within each GPA range, the host VMM can invoke multiple concurrent scanning TDH.MEM.SCAN.COMP threads. The threads balance the work among themselves.

- 15
1. When invoked, TDH.MEM.SCAN.COMP allocates a sub-range to scan. The sub-range size is configured by the host VMM using TDH.MEM.SCAN.CONFIG.
 2. When done with that sub-range, if there is still unscanned memory in the range, TDH.MEM.SCAN.COMP allocates a new sub-range to scan and loops to step 1 above.
 3. If there is no more memory to scan, TDH.MEM.SCAN.COMP returns to the host VMM. If there are no more concurrent scans running in the current range, it indicates a successful scan of the range.
- 20

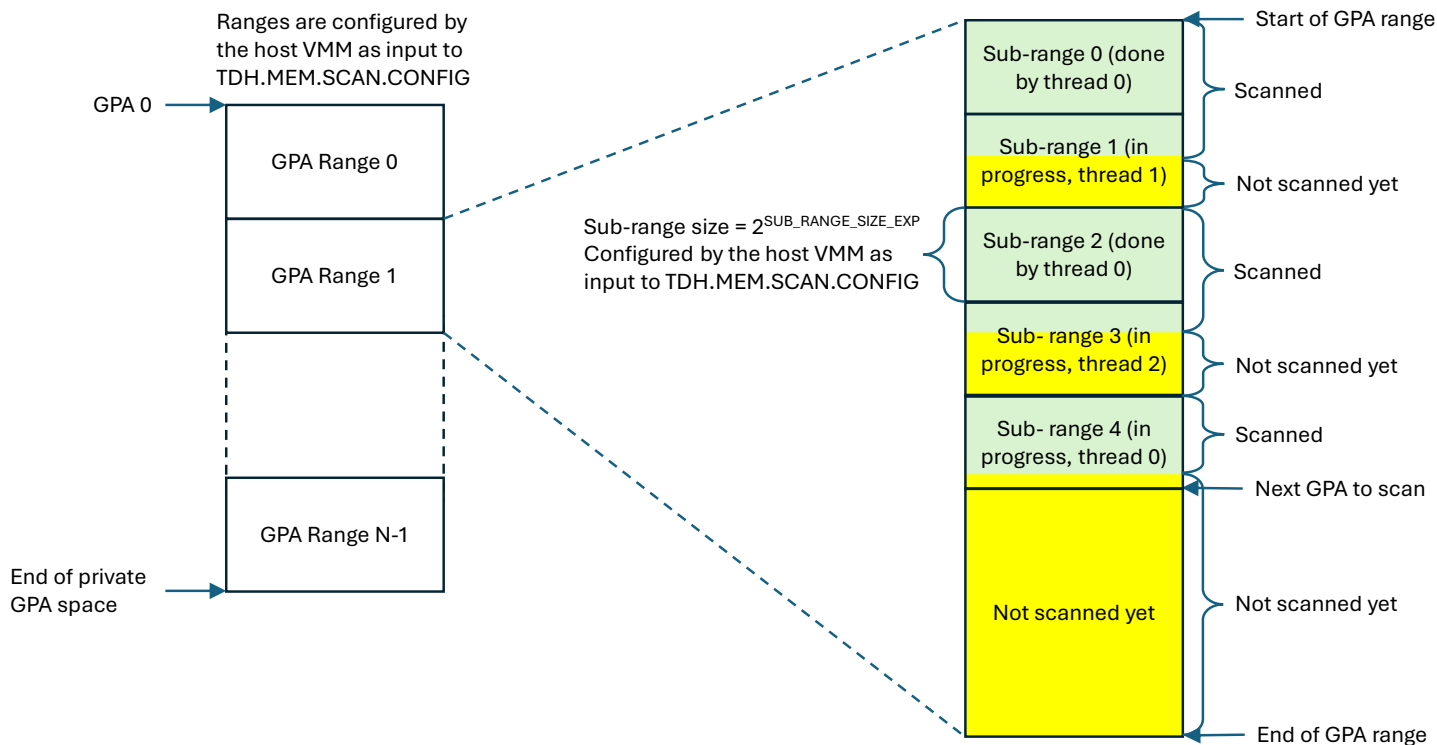


Figure 5.4: Concurrent Scan within a GPA Range

Notes:

- Each concurrent TDH.MEM.SCAN.COMP invocation must be provided with a unique CONTEXT_ID.
- The sub-range size and number of concurrent threads per range can be tuned for best performance, possibly taking into account the guest TD’s memory allocation distributing within the GPA range.

5.4.33.3.4.3. Comprehensive Scan Start

A new comprehensive scan starts when the host VMM calls TDH.MEM.SCAN.COMP to perform a comprehensive scan, and either of the following conditions is true:

- This is the first invocation of TDH.MEM.SCAN.COMP since the TD was created or imported, or
- This is the first invocation of TDH.MEM.SCAN.COMP since TDH.MEM.SCAN.RESET was called.

5.4.33.3.4.4. Comprehensive Scan Context

To support multiple concurrent comprehensive scan threads, the TDX module holds multiple instances of scan contexts per TD, allocated by TDH.MEM.SCAN.CONFIG. The CONTEXT_ID parameter is used as a handle for the internally held context, to allow resumption after an incomplete operation or an interruption. It must be between 0 and NUM_MEM_SCAN_CONTEXTS - 1. See enumeration details above.

5.4.33.3.4.5. Comprehensive Scan Success

If TDH.MEM.SCAN.COMP executes successfully, the return status in RAX indicates the following:

- TDX_SUCCESS** The current invocation of TDH.MEM.SCAN.COMP completed successfully, but there are concurrent instances of TDH.MEM.SCAN.COMP still scanning the current GPA range.
- TDX_MEM_RANGE_SCAN_SUCCESS** Comprehensive scan of the current GPA range completed successfully, but there are concurrent instances of TDH.MEM.SCAN.COMP threads still scanning other GPA ranges. This status is returned at most once per GPA range.
- TDX_MEM_SCAN_SUCCESS** Comprehensive scan of the entire GPA space completed successfully. This status is returned at most once per comprehensive scan.

5.4.33.3.4.6. Comprehensive Scan Failure

There might be cases where a TDH.MEM.SCAN.COMP instance fails in a way that fails the whole comprehensive scan. For example, this happens when a blocked SEPT entry that is discovered by TDH.MEM.SCAN.COMP(DCHECK). In such

cases, TDH.MEM.SCAN.COMP returns a TDX_MEM_SCAN_FAILED_BLOCKED_RANGE status. Concurrently executing TDH.MEM.SCAN.COMP instances consequently also fail, returning a TDX_MEM_SCAN_FAILED(OTHER_THREAD_FAILED) status.

After a comprehensive scan failed, the host VMM may start a new comprehensive scan. To do that, it first needs to call TDH.COMP.SCAN.RESET to reset the internal comprehensive scan state.

5.4.33.3.4.7. Backward Compatibility

TDH.MEM.SCAN.COMP supports scanning the memory of a TD that was created by an older TDX module that didn't support TDH.MEM.SCAN.COMP, and the TDX module was later updated using a TD-preserving update. This scan may be somewhat less efficient since some information collected during the TD lifetime may not exist.

5.4.33.3.5. Concurrency

TDH.MEM.SCAN.COMP may be called concurrently on multiple LPs and may run concurrently with other functions (e.g., TDH.EXPORT.MEM). This is especially important for the DCHECK operation, which is done during the export blackout time and thus should be completed as fast as possible.

5.4.33.3.5.1. Memory Page Concurrency

For the DCHECK operation, concurrently changing page state in a way that impacts export correctness is prevented by the architectural restrictions on the allowed memory management operations.

5.4.33.3.5.2. SEPT Tree Concurrency

TDH.MEM.SCAN.COMP acquires a shared lock on the SEPT trees, to prevent blocking or changes to the tree structure (e.g., by TDH.MEM.RANGE.BLOCK or TDH.MEM.SEPT.REMOVE) while TDH.MEM.SCAN.COMP is running. However, this lock does not prevent SEPT tree structure modification when TDH.MEM.SCAN is not executing (e.g., it was interrupted).

For the DCHECK operation, the non-blocking export architecture prevents SEPT tree structure change while the comprehensive scan is in progress, even if no instance of TDH.MEM.SCAN.COMP is running, by not allowing the applicable memory management functions to run.

5.4.33.3.5.3. SEPT Entry Concurrency

If TDH.MEM.SCAN.COMP fails to acquire a lock on an SEPT entry, it returns with a TDX_INTERRUPTED_BUSY status in RAX. The host VMM is expected to resume TDH.MEM.SCAN.COMP; see the discussion on interruptibility below.

5.4.33.3.6. GPA List-of-Lists Processing

Processing of the GPA list-of-list by TDH.MEM.SCAN.COMP and by TDH.MEM.SCAN.RANGE is similar; thus, it is described in 5.4.32.1.

5.4.33.3.7. Interruption and Resumption

TDH.MEM.SCAN.COMP is interruptible and resumable. An interruption occurs in the following cases:

- If a pending interrupt is detected during operation, TDH.MEM.SCAN.COMP returns with a TDX_INTERRUPTED_RESUMABLE status in RAX.
- If TDH.MEM.SCAN.COMP fails to acquire a lock on an SEPT entry, it returns with a TDX_INTERRUPTED_BUSY Status in RAX.
- If TDH.MEM.SCAN.COMP detects that a concurrent function (such as TDH.EXPORT.ABORT) has failed to acquire an exclusive lock on the comprehensive memory scan state, it yields and returns with a TDX_INTERRUPTED_BUSY status in RAX.
- A GPA list-of-lists full condition has been detected. In this case, TDH.MEM.SCAN.COMP returns with a TDX_INTERRUPTED_LIST_FULL status.

In all cases, OVERALL_NEXT_ENTRY (in R11) is updated with the overall index of the next list entry to process. Except for the GPA list-of-lists full case, the host VMM may re-invoke TDH.MEM.SCAN.COMP soon after handling the interrupt. To resume after a GPA list-of-lists full case, the host VMM must provide a GPA list-of-lists with available entries. See 5.4.32.1 for details.

If an interrupted TDH.MEM.SCAN.COMP is resumed after an error was detected by a TDH.MEM.SCAN.COMP running in another context, it immediately returns with a TDX_MEM_SCAN_FAILED_OTHER_THREAD status.

5.4.33.4. Operands Information

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

Table 5.132: TDH.MEM.SCAN.COMP Memory Operands Information

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|------------------|---|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | GPA list info page | HPA | R | Shared | 4KB | None | None | None |
| Explicit | N/A | N/A | GPA list pages (via GPA list info page) | HPA | RW | Shared | 4KB | None | None | None |
| Explicit | RDX | HPA | TDR page | TDR | R | Opaque | 4KB | Shared | Shared | Shared |
| Explicit | R8 | Unsigned Integer | Comprehensive scan context | N/A | 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 | R | Opaque | N/A | Shared | N/A | N/A |
| Implicit | N/A | GPA | L1 Secure EPT tree | N/A | RW | Private | N/A | Shared | N/A | N/A |
| Implicit | N/A | GPA | L1 Secure EPT entry | SEPT Entry | RW | Private | N/A | Exclusive(h) | N/A | N/A |
| Implicit | N/A | GPA | L2 Secure EPT trees | N/A | RW | Private | N/A | Shared(i) | N/A | N/A |
| Implicit | N/A | GPA | L2 Secure EPT entries | SEPT Entry | RW | Private | N/A | Exclusive(i) | N/A | N/A |
| Implicit | N/A | N/A | Comprehensive scan state | N/A | RW | Opaque | N/A | Shared | N/A | N/A |

5

5.4.33.5. Completion Status Codes

Table 5.133: TDH.MEM.SCAN.COMP Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-----------------------------------|--|
| TDX_BLOCKED_PAGES_EXIST | There are memory pages that have been blocked by TDH.MEM.RANGE.BLOCK. This is not permitted if the TDX module is configured for non-blocking export. |
| TDX_MEM_RANGE_SCAN_SUCCESS | |
| TDX_MEM_SCAN_FAILED_BLOCKED_RANGE | |
| TDX_MEM_SCAN_FAILED_OTHER_THREAD | |
| TDX_MEM_SCAN_SUCCESS | |
| TDX_INTERRUPTED_BUSY | |
| TDX_INTERRUPTED_LIST_FULL | The output GPA lists are full. |

| Completion Status Code | Description |
|------------------------------|--|
| TDX_INTERRUPTED_RESUMABLE | |
| TDX_INVALID_RESUMPTION | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | Operation is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |
| TDX_TDCS_PAGES_REQUIRED | |

5.4.34. TDH.MEM.SCAN.CONFIG Leaf

Unreleased Feature: TDH.MEM.SCAN.* is a set of interface functions that will be provided with Non-Blocking Export, a feature which has not been released yet at the time of writing of this document. Details related to that feature serve as a preview and are subject to change.

- 5 Configure memory scan and add control structure pages.

5.4.34.1. Input Operands

Table 5.134: TDH.MEM.SCAN.CONFIG Input Operands Definition

| Operand | | Description | | |
|---------|------------------|--|----------------|---|
| RAX | LEAF_AND_VERSION | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 94 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 | |
| RCX | RANGE_LIST_INFO | The shared physical address (including HKID) of a of a page where memory scan control page will be created | | |
| | | Bits | Field | Description |
| | | 8:0 | NUM_RANGES | Number of GPA ranges Must be between 1 and MAX_MEM_SCAN_RANGES, readable by TDH.SYS.RD*. |
| | | 11:9 | RESERVED | Must be 0 |
| | | 51:12 | RANGE_LIST_HPA | Shared HPA (Incl. HKID) of a range list page |
| | 63:52 | RESERVED | Must be 0 | |
| RDX | TD_HANDLE | The physical address of the owner TDR page (HKID bits must be 0) | | |
| R8 | CX0_HPA | The physical address where memory scan control page 0 will be created (HKID bits must be 0) | | |
| R9 | CX1_HPA | The physical address where memory scan control page 1 will be created (HKID bits must be 0) A valid HPA is required if the value of MEM_SCAN_CONTROL_PAGES, readable by TDH.SYS.RD*, is 2 or higher. Otherwise, R9 is ignored. | | |
| R10 | CX2_HPA | The physical address where memory scan control page 2 will be created (HKID bits must be 0) A valid HPA is required if the value of MEM_SCAN_CONTROL_PAGES, readable by TDH.SYS.RD*, is 3 or higher. Otherwise, R10 is ignored. | | |
| R11 | CX3_HPA | The physical address where memory scan control page 3 will be created (HKID bits must be 0) A valid HPA is required if the value of MEM_SCAN_CONTROL_PAGES, readable by TDH.SYS.RD*, is 4 or higher. Otherwise, R11 is ignored. | | |

5.4.34.2. Output Operands

Table 5.135: TDH.MEM.SCAN.CONFIG Output Operands Definition

| Operand | Name | Description |
|---------|--------|--|
| RAX | Status | SEAMCALL instruction return code – see 5.4.1 |
| Other | | Unmodified |

5.4.34.3. Leaf Function Description

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

TDH.MEM.SCAN.CONFIG adds a page where scan control information used by TDH.MEM.SCAN will reside. This function can be invoked at any time after the TDCS pages have been allocated.

TDH.MEM.SCAN.CONFIG can only be successfully invoked if no comprehensive memory scan session is in progress.

10 5.4.34.3.1. Enumeration

TDH.MEM.SCAN.CONFIG is supported if TDH.MEM.SCAN is supported. If not supported, calling TDH.MEM.SCAN.CONFIG returns a TDX_OPERAND_INVALID(RAX) status.

The required number of memory scan control pages is enumerated by NUM_MEM_SCAN_CONTROL_PAGES, readable by TDH.SYS.RD*.

- 15 The maximum number of GPA ranges is enumerated by MAX_MEM_SCAN_RANGES.

5.4.34.3.2. Range List Page

The range list page holds a list of GPA ranges, to be used by TDH.MEM.SCAN. The list is formatted as an array of 64-bit entries, with entry N specifying the start of a GPA range N. The number of valid entries is specified by RCX.NUM_RANGES. See the description of TDH.MEM.SCAN.COMP in 5.4.33.3.4 for details.

20 **Table 5.136: Range List Entry**

| Bits | Name | Description |
|-------|--------------------|--|
| 20:0 | RESERVED | Must be 0 |
| 50:21 | RANGE_START | Bits 50:21 of the range start GPA RANGE_START must be a valid private GPA, lower than the maximum valid private GPA allowed for the TD – see the [TDX Module Base Spec] section on GPA Space Size Configuration and Virtualization. RANGE_START must be aligned on SUB_RANGE_SIZE, i.e., all bits lower than SUB_RANGE_SIZE_EXP must be 0. RANGE_START of entry 0 must be 0. |
| 51 | RESERVED | Must be 0 |
| 57:52 | SUB_RANGE_SIZE_EXP | Specifies the sub-range size to be used by TDH.MEM.SCAN.COMP within the current range: $\text{SUB_RANGE_SIZE} = 2^{\text{SUB_RANGE_SIZE_EXP}}$ SUB_RANGE_SIZE_EXP must be at least 21 (SUB_RANGE_SIZE of 2MB). Sub ranges are used by TDH.MEM.SCAN for multithreaded scanning. A SUB_RANGE_SIZE value equal or higher than the range size forces the range scan to be single threaded. |
| 63:58 | RESERVED | Must be 0 |

5.4.34.3.3. Dynamic PAMT

If the TDX module is configured for dynamic PAMT, the PAMT hierarchy can be built on demand. A `TDX_MISSING_PAMT_PAGE_PAIR` status indicates that a PAMT page pair is missing for a control page. The host VMM may add it using `TDH.PHYMEM.PAMT.ADD` and retry the operation.

5 5.4.34.4. Operands Information

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

Table 5.137: TDH.MEM.SCAN.CONFIG Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|---------------------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | Range list page | N/A | R | Shared | 4KB | N/A | N/A | N/A |
| Explicit | RDX | HPA | TDR page | TDR | RW | Opaque | 4KB | Shared | Shared | Shared |
| Explicit | R8 | HPA | Control page 0 | N/A | RW | Opaque | 4KB | Exclusive | Shared | Shared |
| Explicit | R9 | HPA | Control page 1 | N/A | RW | Opaque | 4KB | Exclusive | Shared | Shared |
| Explicit | R10 | HPA | Control page 2 | N/A | RW | Opaque | 4KB | Exclusive | Shared | Shared |
| Explicit | R11 | HPA | Control page 3 | N/A | RW | Opaque | 4KB | Exclusive | 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(h) | N/A | N/A |
| Implicit | N/A | N/A | Comprehensive scan state | N/A | RW | Opaque | N/A | Exclusive | N/A | N/A |
| Implicit | N/A | N/A | All comprehensive scan contexts | N/A | RW | Opaque | N/A | Exclusive(i) | N/A | N/A |

10 5.4.34.5. Completion Status Codes

Table 5.138: TDH.MEM.SCAN.CONFIG Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|--|--|
| <code>TDX_MEM_SCAN_CONFIG_ALREADY_DONE</code> | |
| <code>TDX_MEM_RANGE_SCAN_SUCCESS</code> | |
| <code>TDX_MISSING_PAMT_PAGE_PAIR</code> | |
| <code>TDX_OP_STATE_INCORRECT</code> | |
| <code>TDX_OPERAND_ADDR_RANGE_ERROR</code> | |
| <code>TDX_OPERAND_BUSY</code> | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| <code>TDX_OPERAND_INVALID</code> | |
| <code>TDX_OPERAND_PAGE_METADATA_INCORRECT</code> | |

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

5.4.35. TDH.MEM.SCAN.RANGE Leaf

Unreleased Feature: TDH.MEM.SCAN.* is a set of interface functions that will be provided with Non-Blocking Export, a feature which has not been released yet at the time of writing of this document. Details related to that feature serve as a preview and are subject to change.

- 5 Scan a range of the TD's private GPA space and perform the requested operation.

5.4.35.1. Input Operands

Table 5.139: TDH.MEM.SCAN.RANGE Input Operands Definition

| Operand | Name | Description | | |
|---------|--------------------|---|----------------|--|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 92 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | | 63:24 | Reserved | Must be 0 |
| RCX | LIST_OF_LISTS_INFO | Information for GPA list-of-lists in shared memory. For details, see the description below, in 5.4.32.1 and in 3.13.2. <ul style="list-style-type: none"> FIRST_ENTRY must be 0. LAST_ENTRY contains the index of the last valid entry in the GPA List Info page. FORMAT must be LIST_OF_LISTS (2). | | |
| RDX | TDR | HPA of the source TD's TDR page (HKID bits must be 0) | | |
| R8 | CONTROLS | Controls fields: | | |
| | | Bits | Field | Description |
| | | 7:0 | OPERATION | Identifies the requested operation – see the table below Enumeration: See below for details. |
| | | 15:8 | QUALIFIER | Additional qualification of the requested operation |
| | | 62:16 | RESERVED | Reserved, must be 0 |
| 63 | RESUME | 0: This is a new invocation of TDH.MEM.SCAN.RANGE 1: This is resumption of a previously interrupted TDH.MEM.SCAN.RANGE operation | | |
| R9 | RANGE_START | The start address of the private GPA range to scan Must be a valid private GPA, aligned on 4KB. | | |
| R10 | RANGE_SIZE | The size of the GPA range to scan Must be a multiple of 4KB. Bits 63:52 must be 0. | | |
| R11 | OVERALL_NEXT_ENTRY | The next overall index of the GPA List entry to be written by TDH.MEM.SCAN.COMP. For details, see the description below and in 5.4.32.1. Must be between 0 and $512 * (\text{LIST_OF_LISTS_INFO.LAST_ENTRY} + 1) - 1$. | | |

5.4.35.2. Output Operands

Table 5.140: TDH.MEM.SCAN.RANGE Output Operands Definition

| Operand | Name | Description |
|---------|--------------------|---|
| RAX | STATUS | SEAMCALL instruction return code, see 5.4.1 |
| R9 | NEXT_START | The first GPA for next scan, aligned on 4KB On any error or interruption, NEXT_START is updated to the address of the memory last successfully scanned so far + 1; see the description below for details. On successful end of the range scan, NEXT_START is set to RANGE_START + RANGE_SIZE. |
| R10 | REMAINING_SIZE | The remaining size of the GPA range to scan, in multiples of 4KB On any error or interruption, RAMAINING_SIZE is updated to RANGE_SIZE minus the size of the memory successfully scanned so far; see the description below for details. On successful end of the range scan, set to 0. |
| R11 | OVERALL_NEXT_ENTRY | The overall index of the next available GPA List entry. For details, see the description below and in 5.4.32.1. |
| Other | | Unmodified |

5.4.35.3. Leaf Function Description

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

5.4.35.3.1. Overview

TDH.MEM.SCAN.RANGE scans the requested TD's GPA range and performs the requested operation. Most operations fill page information in the returned GPA lists.

5.4.35.3.2. Enumeration

TDH.MEM.SCAN.RANGE is supported if any of its operations are enumerated as available. See the operation modes table below for details. If not supported, calling TDH.MEM.SCAN returns a TDX_OPERAND_INVALID(RAX) status.

5.4.35.3.3. Operation Modes and Qualifiers

TDH.MEM.SCAN.RANGE supports multiple operation modes as detailed below. Some TDH.MEM.SCAN.RANGE operation modes are available only if the TDX module is configured for non-blocking export. Operation modes may have qualifiers.

Table 5.141: TDH.MEM.SCAN.RANGE Operation Modes

| OPERATION Value | Name | Description |
|-----------------|-------|--|
| 0 | DSCAN | Scan for export candidates. DSCAN is only available if the TDX module is configured for non-blocking export. Enumeration: DSCAN support is enumerated by TDX_FEATURES0.NON_BLOCKING_EXPORT (bit 41), readable by TDH.SYS.RD* (see 3.3.3.1). |

| OPERATION Value | Name | Description |
|-----------------|----------------|--|
| | | DSCAN has two sub modes, selected by the QUALIFIER input: 0: EXPORT: Scan and return a GPA list of memory pages that either have not been exported or need to be re-exported. 1: REEXPORT: Scan and return a GPA list of memory pages that need to be re-exported. Other: Reserved |
| 2 | EXPORT_RESTORE | Restore SEPT entries state after an export session has been aborted by TDH.EXPORT.ABORT. EXPORT_RESTORE does not return a page list; LIST_OF_LISTS_INFO is ignored. Enumeration: EXPORT_RESTORE support is enumerated by TDX_FEATURES0.SCAN_EXPORT_RESTORE (bit 43), readable by TDH.SYS.RD* (see 3.3.3.1). |
| Other | RESERVED | Reserved |

5.4.35.3.3.1. DSCAN Details

DSCAN skips SEPT sub-tree that are in a BLOCKED state, i.e.:

- If DSCAN encounters a blocked non-leaf SEPT entry, it skips it (and the whole GPA range mapped by it).
- If DSCAN encounters a blocked leaf SEPT entry, it skips that entry.

5.4.35.3.4. Concurrency

TDH.MEM.SCAN.RANGE may be called concurrently on multiple LPs and may run concurrently with other functions (e.g., TDH.EXPORT.MEM).

5.4.35.3.4.1. Memory Page Concurrency

TDH.MEM.SCAN.RANGE may not detect pages that are concurrently added to the TD, or whose state or attributes (including the Dirty bit) are concurrently modified, while scan is in progress.

5.4.35.3.4.2. SEPT Tree Concurrency

TDH.MEM.SCAN.RANGE acquires a shared lock on the SEPT trees, to prevent blocking or changes to the tree structure (e.g., by TDH.MEM.RANGE.BLOCK or TDH.MEM.SEPT.REMOVE) while TDH.MEM.SCAN.RANGE is in progress. Note that this lock does not prevent SEPT tree structure modification when TDH.MEM.SCAN.RANGE is not executing (e.g., it was interrupted).

TDH.MEM.SCAN.RANGE may prevent starvation of concurrent memory management functions by detecting that they failed to acquire an exclusive lock on the SEPT trees. In this case, TDH.MEM.SCAN.RANGE yields and returns with a TDX_INTERRUPTED_BUSY Status in RAX. The host VMM is expected to resume TDH.MEM.SCAN.RANGE. See the discussion on interruptibility below.

5.4.35.3.4.3. SEPT Entry Concurrency

If TDH.MEM.SCAN.RANGE fails to acquire a lock on an SEPT entry, behavior depends on the requested operation:

- For a DSCAN operation, TDH.MEM.SCAN.RANGE skips the busy SEPT entry.
Note: Non-blocking export is designed to ensure that such entries will be scanned at least once by DCHECK. For details, see the [TD Migration Spec].
- For an EXPORT_RESTORE operation, TDH.MEM.SCAN.RANGE returns with a TDX_INTERRUPTED_BUSY Status in RAX. The host VMM is expected to resume TDH.MEM.SCAN.RANGE. See the discussion on interruptibility below.

5.4.35.3.5. GPA List-of-Lists Processing

Processing of the GPA list-of-list by TDH.MEM.SCAN.COMP and by TDH.MEM.SCAN.RANGE is similar; thus, it is described in 5.4.32.1.

5.4.35.3.6. TLB Tracking

For the DSCAN operation, whenever TDH.MEM.SCAN.RANGE clears the Dirty bit of an SEPT entry, it records the current TD epoch (in the PAMT entry for that page). This allows TDH.EXPORT.MEM to check TLB tracking to help ensure that EPT translation caches have been flushed before the page is exported. Unlike write-blocking based export, TLB tracking is page-specific, allowing TDH.MEM.SCAN.RANGE(DSCAN) of a certain GPA range to run concurrently with TDH.EXPORT.MEM of another range.

TD epoch is not recorded for the EXPORT_RESTORE operation.

5.4.35.3.7. Interruption and Resumption

TDH.MEM.SCAN.RANGE is interruptible and resumable. An interruption occurs in the following cases:

- If a pending interrupt is detected during operation, TDH.MEM.SCAN.RANGE returns with a TDX_INTERRUPTED_RESUMABLE status in RAX.
- If TDH.MEM.SCAN.RANGE fails to acquire a lock on an SEPT entry during an EXPORT_RESTORE operation, it returns with a TDX_INTERRUPTED_BUSY Status in RAX.
- If TDH.MEM.SCAN.RANGE detects that a concurrent function (such as TDH.MEM.PAGE.PROMOTE) has failed to acquire an exclusive lock on the SEPT trees, it may yield and return with a TDX_INTERRUPTED_BUSY Status in RAX.
- A GPA list-of-lists full condition has been detected. In this case, TDH.MEM.SCAN.COMP returns with a TDX_INTERRUPTED_LIST_FULL status.

In all cases, the following are updated with the range information and the next list entry index to process:

- NEXT_START (in R9)
- REMAINING_SIZE (in R10)
- OVERALL_NEXT_ENTRY (in R11)

Except for the GPA list-of-lists full case, the host VMM may re-invoke TDH.MEM.SCAN.RANGE soon after handling the interrupt. To resume after a GPA list-of-lists full case, the host VMM must provide a list-of-lists with available entries. See 5.4.32.1 for details.

5.4.35.4. Operands Information

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

Table 5.142: TDH.MEM.SCAN.RANGE Memory Operands Information

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|----------|---|---------------|--------|------------------|-------------|--------------------------|--------------|--------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | GPA list info page | HPA | RW | Shared | 4KB | None | None | None |
| Explicit | N/A | N/A | GPA list pages (via GPA list info page) | HPA | RW | Shared | 4KB | None | None | None |
| Explicit | RDX | HPA | TDR page | TDR | R | Opaque | 4KB | Shared | Shared | Shared |
| Explicit | R9 | GPA | GPA range | GPA | R | 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 | GPA | L1 Secure EPT tree | N/A | RW | Private | N/A | Shared | N/A | N/A |
| Implicit | N/A | GPA | L1 Secure EPT entry | SEPT Entry | RW | Private | N/A | Exclusive(h) | N/A | N/A |

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

5.4.35.5. Completion Status Codes

Table 5.143: TDH.MEM.SCAN.RANGE Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|--------------------------------------|--|
| TDX_MEM_SCAN_FAILED_BLOCKED_RANGE | |
| TDX_MEM_SCAN_FAILED_OTHER_THREAD | |
| TDX_MEM_SCAN_FAILED_PREPARE_REQUIRED | |
| TDX_MEM_SCAN_SUCCESS | |
| TDX_INTERRUPTED_LIST_FULL | The output GPA lists are full. |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | Operation is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |
| TDX_TDCS_PAGES_REQUIRED | |

5.4.36. TDH.MEM.SCAN.RESET Leaf

Unreleased Feature: TDH.MEM.SCAN.* is a set of interface functions that will be provided with Non-Blocking Export, a feature which has not been released yet at the time of writing of this document. Details related to that feature serve as a preview and are subject to change.

5 Reset the TDX module’s comprehensive memory scan internal state for the specified TD.

5.4.36.1. Input Operands

Table 5.144: TDH.MEM.SCAN.RESET Input Operands Definition

| Operand | Name | Description | | |
|---------|------------------|---|----------------|--|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 95 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | | 63:24 | Reserved | Must be 0 |
| RDX | TDR | HPA of the source TD’s TDR page (HKID bits must be 0) | | |

5.4.36.2. Output Operands

Table 5.145: TDH.MEM.SCAN.RESET Output Operands Definition

| Operand | Name | Description |
|---------|--------|---|
| RAX | STATUS | SEAMCALL instruction return code, see 5.4.1 |
| Other | | Unmodified |

5.4.36.3. Leaf Function Description

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

5.4.36.3.1. Overview

TDH.MEM.SCAN.RESET resets the internal state held by the TDX module for comprehensive memory scans of the specified TD’s GPA address space. It is typically used if a scan by TDH.MEM.SCAN.COMP failed, before attempting a new scan.

5.4.36.3.2. Enumeration

TDH.MEM.SCAN.RESET is supported if TDH.MEM.SCAN.COMP is supported. If not supported, calling TDH.MEM.SCAN.RESET returns a TDX_OPERAND_INVALID(RAX) status.

5.4.36.4. Operands Information

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

Table 5.146: TDH.MEM.SCAN.RANGE Memory Operands Information

| Explicit/ Implicit | Reg. | Ref 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 | Shared | Shared | Shared |
| Explicit | R9 | GPA | GPA range | GPA | R | 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 | Comprehensive scan state | N/A | RW | Opaque | N/A | Exclusive | N/A | N/A |
| Implicit | N/A | N/A | All comprehensive scan contexts | N/A | RW | Opaque | N/A | Exclusive(i) | N/A | N/A |

5.4.36.5. Completion Status Codes**Table 5.147: TDH.MEM.SCAN.RESET Completion Status Codes (Returned in RAX) Definition**

| Completion Status Code | Description |
|--|--|
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | Operation is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |
| Unreleased feature: Status list is not complete | |

5

5.4.37. TDH.MEM.SEPT.ADD Leaf

Add and map 4KB L1 and L2 Secure EPT pages to a TD.

5.4.37.1. Input Operands**Table 5.148: TDH.MEM.SEPT.ADD Input Operands Definition**

| Operand | Description | | |
|---------|--|----------------|---|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 3 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Versions 0 and 1 are supported. See the enumeration details below. |
| | 63:24 | Reserved | Must be 0 |
| RCX | EPT mapping information: | | |
| | Bits | Name | Description |
| | 2:0 | Level | Level of the non-leaf Secure EPT entry that will map the new Secure EPT page – see 3.6.1 Level must be between 1 and 3 for a 4-level EPT or between 1 and 4 for a 5-level EPT. |
| | 11:3 | Reserved | Reserved: must be 0 |
| | 51:12 | GPA | Bits 51:12 of the guest physical address 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 | TD handle and flags: | | |
| | Bits | Name | Description |
| | 0 | ALLOW_EXISTING | Flags that TDH.MEM.SEPT.ADD should not fail if an SEPT page to be added already exists in the L1 or L2 SEPT tree. Instead, it should just return an indication in the output operand, as described below. |
| | 11:1 | Reserved | Reserved: must be 0 |
| | 51:12 | TDR_HPA | Bits 51:12 of the host physical address of the parent TDR page (HKID bits must be 0) |
| | 63:52 | Reserved | Reserved: must be 0 |
| R8 | Host physical address of the new L1 Secure EPT page to be added to the TD (HKID bits must be 0) For TDH.MEM.SEPT.ADD version 1 or higher: <ul style="list-style-type: none"> If the value of R8 is NULL_PA (-1), no L1 SEPT page is added. | | |

| Operand | Description |
|---------|---|
| | <ul style="list-style-type: none"> Else, bit 63 of R8 is ignored. |
| R9 | <p>For TDH.MEM.SEPT.ADD version 1 or higher, R9 specifies the HPA of a new L2 VM #1 Secure EPT page to be added to the TD (HKID bits must be 0).</p> <ul style="list-style-type: none"> If the value of R9 is NULL_PA (-1), no L2 VM #1 SEPT page is added. Else, bit 63 of R9 is ignored. |
| R10 | <p>For TDH.MEM.SEPT.ADD version 1 or higher, R10 specifies the HPA of a new L2 VM #2 Secure EPT page to be added to the TD (HKID bits must be 0).</p> <ul style="list-style-type: none"> If the value of R10 is NULL_PA (-1), no L2 VM #2 SEPT page is added. Else, bit 63 of R10 is ignored. |
| R11 | <p>For TDH.MEM.SEPT.ADD version 1 or higher, R11 specifies the HPA of a new L2 VM #3 Secure EPT page to be added to the TD (HKID bits must be 0).</p> <ul style="list-style-type: none"> If the value of R11 is NULL_PA (-1), no L2 VM #3 SEPT page is added. Else, bit 63 of R11 is ignored. |

5.4.37.2. Output Operands

Table 5.149: TDH.MEM.SEPT.ADD Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| RCX | <p>In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESUMABLE, RCX is unmodified.</p> <p>Else, RCX returns extended error information part 1.</p> <p>In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2</p> <p>The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page; it may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX.</p> <p>In other cases, RCX returns 0.</p> |
| RDX | <p>In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESUMABLE, RDX is unmodified.</p> <p>Else, RDX returns extended error information part 2.</p> <p>In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2</p> <p>In other cases, RDX returns 0.</p> |
| R8 | <p>For TDH.MEM.SEPT.ADD version 1 or higher:</p> <ul style="list-style-type: none"> If a provided L1 SEPT page has been added, R8 returns NULL_PA (-1). Else, if an L1 SEPT page already exists, bit 63 of R8 is set to 1, other bits are unmodified. Else, bit 63 of R8 is cleared to 0, other bits are unmodified. <p>For TDH.MEM.SEPT.ADD version 0, R8 is unmodified.</p> |

| Operand | Description |
|---------|--|
| R9 | <p>For TDH.MEM.SEPT.ADD version 1 or higher:</p> <ul style="list-style-type: none"> • If a provided L2 VM #1 SEPT page has been added, R9 returns NULL_PA (-1). • Else, if an L2 VM #1 SEPT page already exists, bit 63 of R9 is set to 1, other bits are unmodified. • Else, bit 63 of R9 is cleared to 0, other bits are unmodified. <p>For TDH.MEM.SEPT.ADD version 0, R9 is unmodified.</p> |
| R10 | <p>For TDH.MEM.SEPT.ADD version 1 or higher:</p> <ul style="list-style-type: none"> • If a provided L2 VM #2 SEPT page has been added, R10 returns NULL_PA (-1). • Else, if an L2 VM #2 SEPT page already exists, bit 63 of R10 is set to 1, other bits are unmodified. • Else, bit 63 of R10 is cleared to 0, other bits are unmodified. <p>For TDH.MEM.SEPT.ADD version 0, R10 is unmodified.</p> |
| R11 | <p>For TDH.MEM.SEPT.ADD version 1 or higher:</p> <ul style="list-style-type: none"> • If a provided L2 VM #3 SEPT page has been added, R11 returns NULL_PA (-1). • Else, if an L2 VM #3 SEPT page already exists, bit 63 of R11 is set to 1, other bits are unmodified. • Else, bit 63 of R11 is cleared to 0, other bits are unmodified. <p>For TDH.MEM.SEPT.ADD version 0, R11 is unmodified.</p> |
| Other | Unmodified |

5.4.37.3. Leaf Function Description

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

5 5.4.37.3.1. Overview

TDH.MEM.SEPT.ADD adds a set of 4KB Secure EPT pages to a TD and maps them to the provided GPA and level. SEPT pages can be added to the main (L1) SEPT tree and/or to one or more of the L2 VMs' SEPT trees. TDH.MEM.SEPT.ADD initializes the SEPT pages to hold 512 free entries using the TD's ephemeral private key.

10 L2 SEPT trees may not be deeper than the L1 SEPT tree. To add an L2 SEPT page at some level, there must either already be an L1 SEPT page at that level, or an L1 SEPT page at that level is being added by the current TDH.MEM.SEPT.ADD invocation.

5.4.37.3.2. Enumeration

15 Availability of TDH.MEM.SEPT.ADD version 1 is enumerated by TDX_FEATURES0.TD_PARTITIONING (bit 7), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.MEM.SEPT.ADD with a version number higher than 0 returns a TDX_OPERAND_INVALID(RAX) status.

5.4.37.3.3. Dynamic PAMT

If the TDX Module is configured for dynamic PAMT, the PAMT hierarchy can be built on demand. A TDX_MISSING_PAMT_PAGE_PAIR status indicates that a PAMT page pair is missing for one of the new SEPT pages. The host VMM may add it using TDH.PHYMEM.PAMT.ADD and retry the operation.

20 5.4.37.3.4. Interruptibility

If a version number higher than 0 is specified on input, TDH.MEM.SEPT.ADD is interruptible. If a pending interrupt is detected during operation, TDH.MEM.SEPT.ADD returns with a TDX_INTERRUPTED_RESUMABLE status in RAX. The SEPT page HPA values in R8, R9, R10 and R11 are updated.

TDH.MEM.SEPT.ADD is designed to be invoked in a loop until all required SEPT pages have been added:

3. Call TDH.MEM.SEPT.ADD.
4. While RAX indicates TDX_INTERRUPTED_RESUMABLE:
 - 4.1. Call TDH.MEM.SEPT.ADD with the GPR values as returned by the previous call.
 - 4.2. If an error indication other than TDX_INTERRUPTED_RESUMABLE is returned, abort.

5.4.37.3.5. SEPT Page HPA Input and Output Operands

The table below shows the values of the SEPT HPA arguments in R8 – R11 when version 1 or higher is selected and no error occurs (RAX returns TDX_SUCCESS or TDX_INTERRUPTED_RESUMABLE).

Table 5.150: Meaning of TDH.MEM.SEPT.ADD’s SEPT HPA Arguments on Input and Output (Version > 0, No Error)

| Value | R8 – R11 on Input | R8 – R11 on Output |
|--|---|---|
| NULL_PA (-1) | No new SEPT page to add | New SEPT page has been added |
| Bits 62:0: Valid HPA, HKID bits are 0 Bit 63: 0 | Bits 62:0: HPA of new SEPT page to add Bit 63: Ignored | N/A |
| Bits 62:0: Valid HPA, HKID bits are 0 Bit 63: 1 | | SEPT page already exists, new SEPT page has not been used |

5.4.37.3.6. Atomicity

Unless terminated with a TDX_INTERRUPTED_RESUMABLE indication, TDH.MEM.SEPT.ADD either fully succeeds in adding the requested SEPT pages or doesn’t add any page.

In case of an interrupt (TDX_INTERRUPTED_RESUMABLE), if the host VMM invokes TDH.MEM.SEPT.ADD in a loop as described above and doesn’t initiate other operations that impact TDH.MEM.SEPT.ADD (e.g., TDH.MEM.RANGE.BLOCK), then this atomicity still holds at the end of the loop.

5.4.37.4. Operands Information

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

Table 5.151: TDH.MEM.SEPT.ADD Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|---------------------|--|------------------|--------|---------------------|----------------------------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | GPA and Level | Secure EPT page (GPA) ²⁷ | 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) ²⁷ | SEPT_PAGE | RW | Private | 4KB | Exclusive | Shared | Shared |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared | N/A | N/A |
| Implicit | N/A | GPA | L1 Secure EPT tree | N/A | RW | Private | N/A | Shared | N/A | N/A |

²⁷ RCX and R8 denote the same Secure EPT page operand, using HPA and GPA respectively

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

5.4.37.5. Completion Status Codes

Table 5.152: TDH.MEM.SEPT.ADD Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_EPT_ENTRY_STATE_INCORRECT | |
| TDX_EPT_WALK_FAILED | |
| TDX_INTERRUPTED_RESUMABLE | TDH.MEM.SEPT.ADD's operation has been interrupted by an external event; it may be resumed from the point where it was interrupted by calling it again. |
| TDX_L2_SEPT_ENTRY_NOT_FREE | |
| TDX_L2_SEPT_WALK_FAILED | |
| TDX_MISSING_PAMT_PAGE_PAIR | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.MEM.SEPT.ADD is successful |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |

5.4.38. TDH.MEM.SEPT.RD Leaf

Read a Secure EPT entry.

Table 5.153: TDH.MEM.SEPT.RD Input Operands Definition

| Operand | | Description | | |
|---------|---------------------|---|----------------|--|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 25 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version |
| | | 63:24 | Reserved | Must be 0 |
| RCX | GPA Mapping | EPT mapping information: | | |
| | | Bits | Name | Description |
| | | 2:0 | Level | Level of the Secure EPT entry to read – see 3.6.1 Level must be between 0 and 3 for a 4-level EPT or between 0 and 4 for a 5-level EPT. |
| | | 11:3 | Reserved | Reserved: must be 0 |
| | | 51:12 | GPA | Bits 51:12 of the guest physical address for the Secure EPT entry to read Depending on the level, the following least significant bits must be 0: Level 0 (EPTE): None Level 1 (EPDE): Bits 20:12 Level 2 (EPDPTE): Bits 29:12 Level 3 (EPML4E): Bits 38:12 Level 4 (EPML5E): Bits 47:12 |
| | | 63:52 | Reserved | Reserved: must be 0 |
| RDX | TD Handle and Flags | TD handle and flags: | | |
| | | Bits | Name | Description |
| | | 0 | READ_L2_ATTR | Flags that L2 attributes should be returned in R8 |
| | | 11:1 | Reserved | Reserved: must be 0 |
| | | 51:12 | HPA | Bits 51:12 of the host physical address of the parent TDR page (HKID bits must be 0) |
| | | 63:52 | Reserved | Reserved: must be 0 |

5

Table 5.154: TDH.MEM.SEPT.RD Output Operands Definition

| Operand | | Description |
|---------|--------|--|
| RAX | Status | SEAMCALL instruction return code – see 5.4.1 |

| Operand | | Description |
|---------|----------------------|---|
| RCX | SEPT Entry | Secure EPT entry architectural content – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page; it may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. <ul style="list-style-type: none"> In case of successful operation, the requested entry's architectural content is returned. In case of EPT walk error, the architectural content of the Secure EPT entry where the error was detected is returned. In other cases, RCX returns 0. |
| RDX | SEPT Level and State | Secure EPT entry level and state – see 3.6.2 <ul style="list-style-type: none"> In case of successful operation, the requested entry's information is returned. In case of EPT walk error, the information of the Secure EPT entry where the error was detected is returned. In other cases, RDX returns 0. |
| R8 | L2 Attributes | If the TD's ATTRIBUTES.DEBUG is 1 and READ_L2_ATTR is 1, R8 returns the L2 attributes of the applicable L2 SEPT entries, in the format defined in 3.6.3. Else, R8 is unmodified. |
| Other | N/A | Unmodified |

5.4.38.1. Leaf Function Description

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

- 5 TDH.MEM.SEPT.RD reads a Secure EPT entry. If the TD's ATTRIBUTES.DEBUG is 1, then TDH.MEM.SEPT.RD can return the page's L2 attributes.

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

Table 5.155: TDH.MEM.SEPT.RD Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|---------------------|------------------------|------------------|--------|---------------------|----------------------------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | GPA and Level | Secure EPT entry | SEPT_ENTRY | R | Private | 2 ^{12+9*Level} Bytes | None | None | None |
| Explicit | RDX | HPA | TDR page | TDR | R | Opaque | 4KB | Shared | Shared | Shared |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared | N/A | N/A |
| Implicit | N/A | GPA | L1 Secure EPT Tree | N/A | R | Private | N/A | Shared | N/A | N/A |
| Implicit | N/A | GPA | L1 Secure EPT entry | SEPT Entry | RW | Private | N/A | Exclusive | N/A | N/A |

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|--------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Implicit | N/A | GPA | L2 Secure EPT Tree | N/A | R | Private | N/A | Shared(i) | N/A | N/A |
| Implicit | N/A | GPA | L2 Secure EPT entries | SEPT Entry | RW | Private | N/A | Exclusive(i) | N/A | N/A |

5.4.38.2. Completion Status Codes

Table 5.156: TDH.MEM.SEPT.RD Completion Status Codes (Returned in RAX) Definition

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

5.4.39. TDH.MEM.SEPT.REMOVE Leaf

Remove an empty L1 Secure EPT page and any associated L2 SEPT pages from a TD.

5.4.39.1. Input Operands**Table 5.157: TDH.MEM.SEPT.REMOVE Input Operands Definition**

| Operand | Description | | |
|---------|--|----------------|---|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 30 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Versions 0 and 1 are supported. See the enumeration details below. |
| | 63:24 | Reserved | Must be 0 |
| RCX | EPT mapping information: | | |
| | Bits | Name | Description |
| | 2:0 | Level | Level of the non-leaf Secure EPT entry that maps the Secure EPT page to be removed – see 3.6.1 Level must be between 1 and 3 for a 4-level EPT or between 1 and 4 for a 5-level EPT. |
| | 11:3 | Reserved | Reserved: must be 0 |
| | 51:12 | GPA | Bits 51:12 of the guest physical address for the Secure EPT page to be removed Depending on the level, the following least significant bits must be 0: Level 1 (EPT): Bits 20:12 Level 2 (EPD): Bits 29:12 Level 3 (EPDPT): Bits 38:12 Level 4 (EPML4): Bits 47:12 |
| | 63:52 | Reserved | Reserved: must be 0 |
| RDX | Host physical address of the parent TDR page (HKID bits must be 0) | | |

5

5.4.39.2. Output Operands**Table 5.158: TDH.MEM.SEPT.REMOVE Output Operands Definition**

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |

| Operand | Description |
|---------|---|
| RCX | <p>If TDH.MEM.SEPT.REMOVE succeeded:</p> <ul style="list-style-type: none"> RCX returns the HPA of the removed L1 SEPT page (HKID bits are set to 0). If the TDX Module has been configured for Dynamic PAMT, then bit 62 serves as a dynamic PAMT removal hint; it indicates that all entries in the PAMT page pair mapping the removed L1 SEPT page are free. <p>In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2</p> <p>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.</p> <p>In other cases, RCX returns 0.</p> |
| RDX | <p>Extended error information part 2</p> <p>In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2</p> <p>In other cases, RDX returns 0.</p> |
| R9 | <p>If TDH.MEM.SEPT.REMOVE version is 1 or higher:</p> <ul style="list-style-type: none"> If L2 VM #1's L2 SEPT page has been removed, R9 returns the HPA of that SEPT page (HKID bits are set to 0). If the TDX Module has been configured for Dynamic PAMT, then bit 62 serves as a dynamic PAMT removal hint; it indicates that all entries in the PAMT page pair mapping the removed L2 #1 SEPT page are free. Else, R9 returns NULL_PA (-1). <p>Else, R9 is unmodified.</p> |
| R10 | <p>If TDH.MEM.SEPT.REMOVE version is 1 or higher:</p> <ul style="list-style-type: none"> If L2 VM #2's L2 SEPT page has been removed, R10 returns the HPA of that SEPT page (HKID bits are set to 0). If the TDX Module has been configured for Dynamic PAMT, then bit 62 serves as a dynamic PAMT removal hint; it indicates that all entries in the PAMT page pair mapping the removed L2 #2 SEPT page are free. Else, R10 returns NULL_PA (-1). <p>Else, R10 is unmodified.</p> |
| R11 | <p>If TDH.MEM.SEPT.REMOVE version is 1 or higher:</p> <ul style="list-style-type: none"> If L2 VM #3's L2 SEPT page has been removed, R11 returns the HPA of that SEPT page (HKID bits are set to 0). If the TDX Module has been configured for Dynamic PAMT, then bit 62 serves as a dynamic PAMT removal hint; it indicates that all entries in the PAMT page pair mapping the removed L2 #3 SEPT page are free. Else, R11 returns NULL_PA (-1). <p>Else, R11 is unmodified.</p> |
| Other | Unmodified |

5.4.39.3. Leaf Function Description

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

5.4.39.3.1. Overview

TDH.MEM.SEPT.REMOVE removes an empty Secure EPT page or pages, with all 512 entries marked as FREE, from the TD's Secure EPT trees.

The L1 SEPT page and all existing L2 SEPT pages at the requested GPA and level are removed.

- 5 On successful operation, it TDH.MEM.SEPT.REMOVE marks the removed 4KB physical pages as free in PAMT.

5.4.39.3.2. Enumeration

Availability of TDH.MEM.SEPT.REMOVE version 1 is enumerated by TDX_FEATURES0.TD_PARTITIONING (bit 7), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.MEM.SEPT.REMOVE with a version number higher than 0 returns a TDX_OPERAND_INVALID(RAX) status.

- 10 Support of dynamic PAMT is enumerated by TDX_FEATURES0.DYNAMIC_PAMT (bit 36).

5.4.39.3.3. Blocking and TLB Tracking

If the TD may be running, the removed GPA range must be blocked and TLB tracked. Else (e.g., TDH.MR.FINALIZE has not yet been executed, or the TD has been paused for export), no blocking and tracking is required.

5.4.39.3.4. Dynamic PAMT

- 15 If the TDX Module is configured for dynamic PAMT, bit 62 of RCX, R9, R10 and R11 are returned as hints indicating that all entries in the PAMT pages mapping the respective removed SEPT pages are free, and they can be removed using TDH.PHYMEM.PAMT.REMOVE.

Note: If the same PAMT pages map more than one of the removed SEPT pages, bit 62 may indicate so in only one of RCX, R9, R10 and R11.

- 20 **5.4.39.3.5. Atomicity**

TDH.MEM.SEPT.REMOVE either fully succeeds in removing the requested SEPT pages or doesn't remove any page.

5.4.39.3.6. Removed Page Initialization

On platforms which do not use ACT, after the SEPT pages have been removed, the host VMM should initialize their content before they are reused as non-private pages, as described in the [Base Spec].

- 25 **5.4.39.4. Operands Information**

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

Table 5.159: TDH.MEM.SEPT.REMOVE Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|---------------------|------------------------|------------------|--------|---------------------|----------------------------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | GPA and Level | Secure EPT page | SEPT_PAGE | R | Private | $2^{12+9*\text{Level}}$ Bytes | Exclusive | None | None |
| Explicit | RDX | HPA | TDR page | TDR | RW | Opaque | 4KB | Shared | Shared | Shared |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared | N/A | N/A |
| Implicit | N/A | GPA | L1 Secure EPT Tree | N/A | RW | Private | N/A | Exclusive | N/A | N/A |
| Implicit | N/A | GPA | L1 Secure EPT entry | SEPT Entry | RW | Private | N/A | Exclusive(h) | N/A | N/A |

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|--------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Implicit | N/A | GPA | L2 Secure EPT Trees | N/A | RW | Private | N/A | Exclusive(i) | N/A | N/A |
| Implicit | N/A | GPA | L2 Secure EPT entries | SEPT Entry | RW | Private | N/A | Exclusive(i) | N/A | N/A |

5.4.39.5. Completion Status Codes

Table 5.160: TDH.MEM.SEPT.REMOVE Completion Status Codes (Returned in RAX) Definition

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

5.4.40. TDH.MEM.SHARED.SEPT.WR Leaf

Add mapping of a Shared GPA range from Secure EPT into Shared EPT pages.

Table 5.161: TDH.MEM.SHARED.SEPT.WR Input Operands Definition

| Operand | Description | | |
|---------|---|---------------------|---|
| RAX | SEAMCALL instruction leaf number and version | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 163 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Versions 0 and 1 are supported. See the enumeration details below. |
| | 63:24 | Reserved | Must be 0 |
| RCX | EPT mapping information: | | |
| | Bits | Name | Description |
| | 2:0 | Level | Level of the Secure EPT entry to write Level must be the level of the Secure EPT page which is indexed by the GPA Shared bit (i.e., GPAW + 3). |
| | 11:3 | Reserved | Reserved: must be 0 |
| | 51:12 | GPA | Bits 51:12 of the shared guest physical address for the Secure EPT entry to write. Depending on the level, the following least significant bits must be 0: Level 3 (EPML4E): Bits 38:12 Level 4 (EPML5E): Bits 47:12 |
| 63:52 | Reserved | Reserved: must be 0 | |
| RDX | Host physical address of the parent TDR page (HKID bits must be 0) | | |
| R8 | If the requested version is 1 or higher and the value of R8 is NULL_PA (-1), R8 is ignored. Else, R8 specifies a value to be written to the applicable L1 EPT entry. | | |
| R9 | If the requested version is 0, R9 is ignored. Else (the requested version is 1 or higher), the number of L2 VMs is ≥ 1 and the value of R9 is not NULL_PA (-1), R9 specifies a value to be written to the applicable L2 VM #1 Secure EPT entry. Else, R9 is ignored. | | |
| R10 | If the requested version is 0, R10 is ignored. Else (the requested version is 1 or higher), the number of L2 VMs is ≥ 2 and the value of R10 is not NULL_PA (-1), R10 specifies a value to be written to the applicable L2 VM #2 Secure EPT entry. Else, R10 is ignored. | | |
| R11 | If the requested version is 0, R11 is ignored. Else (the requested version is 1 or higher), the number of L2 VMs is ≥ 3 and the value of R11 is not NULL_PA (-1), R11 specifies a value to be written to the applicable L2 VM #3 Secure EPT entry. Else, R11 is ignored. | | |

Table 5.162: TDH.MEM.SHARED.SEPT.WR Output Operands Definition

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

5.4.40.1. Leaf Function Description

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

5.4.40.1.1. Overview

TDH.MEM.SHARED.SEPT.WR enables the host VMM to write Secure EPT entries associated with Shared EPT, to map Shared EPT pages. This is required to allow TDX Connect devices DMA translations of shared GPAs using the Secure EPT root page. Only SEPT entries at the level corresponding to the SHARED bit, and associated with a SHARED bit value of 1, are allowed to be written.

TDH.MEM.SHARED.SEPT.WR updates the L1 SEPT tree and all L2 SEPT trees (per the number of L2 VMs configured for the TD).

5.4.40.1.2. Enumeration

Availability of TDH.MEM.SHARED.SEPT.WR is enumerated by TDX_FEATURES0.TDX_CONNECT (bit 6), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.MEM.SHARED.SEPT.WR returns a TDX_OPERAND_INVALID(RAX) status.

Availability of TDH.MEM.SHARED.SEPT.WR version 1 or higher is enumerated by TDX_FEATURES0.TDX_CONNECT_PARTITIONING (bit 32).

5.4.40.2. Operands Information

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

Table 5.163: TDH.MEM.SHARED.SEPT.WR Operands Information Definition

| Explicit/ Implicit | Reg. | Ref. Type | Resource | Resource Type | Access | Access Semantics | Align. Check | Concurrency Restrictions | | |
|-----------------------|------|---------------------|------------------------|------------------|--------|---------------------|----------------------------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | GPA and Level | L1 Secure EPT entry | SEPT_ENTRY | R | Private | $2^{12+9*\text{Level}}$ Bytes | None | None | None |

| Explicit/ Implicit | Reg. | Ref. 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 | Shared | Shared | Shared |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared | N/A | N/A |
| Implicit | N/A | GPA | L1 Secure EPT Tree | N/A | R | Private | N/A | Shared | N/A | N/A |
| Implicit | N/A | GPA | L2 Secure EPT Trees | N/A | R | Private | N/A | Shared(i) | N/A | N/A |
| Implicit | N/A | GPA | L2 Secure EPT entries | SEPT Entry | RW | Private | N/A | None | N/A | N/A |

5.4.40.3. Completion Status Codes

Table 5.164: TDH.MEM.SHARED.SEPT.WR Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------------|-------------|
| TDX_EPT_WALK_FAILED | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | |
| TDX_OPERAND_INVALID | |
| TDX_TDCS_NOT_ALLOCATED | |
| TDX_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TD_NOT_INITIALIZED | |

5.4.41. TDH.MEM.TRACK Leaf

Increment the TD's TLB epoch counter.

5.4.41.1. Input Operands

Table 5.165: TDH.MEM.TRACK Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 38 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| 63:24 | Reserved | Must be 0 | |
| RCX | The physical address of the parent TDR page (HKID bits must be 0) | | |

5

5.4.41.2. Output Operands

Table 5.166: TDH.MEM.TRACK Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.41.3. Leaf Function Description

- 10 **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. It is used as part of the TLB invalidation tracking sequence, e.g., after blocking a memory page using TDH.MEM.RANGE.BLOCK and before removing it using TDH.MEM.PAGE.REMOVE. For details, see the [TDX Module Base Spec] discussion of TLB tracking.

15 5.4.41.4. Operands Information

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

Table 5.167: TDH.MEM.TRACK Operands Information Definition

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

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|-------------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Implicit | N/A | N/A | TDCS Epoch Tracking Fields | N/A | RW | Opaque | N/A | Exclusive | N/A | N/A |

5.4.41.5. Completion Status Codes

Table 5.168: TDH.MEM.TRACK Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|---|
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. Note the special case where the indicated operand is TLB_EPOCH. This may happen due to a conflict with TDH.VP.ENTER. The host VMM should retry TDH.MEM.TRACK. |
| TDX_OPERAND_BUSY | |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_PREVIOUS_TLB_EPOCH_BUSY | |
| TDX_SUCCESS | TDH.MEM.TRACK is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |

5.4.42. TDH.MEM.WR Leaf

5.4.42.1. Input Operands

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

Table 5.169: TDH.MEM.WR Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 14 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | The guest physical address of a naturally aligned 8-byte chunk of a guest TD private page | | |
| RDX | Host physical address of the parent TDR page (HKID bits must be 0) | | |
| R8 | Data to be written to memory | | |

5

5.4.42.2. Output Operands

Table 5.170: TDH.MEM.WR Output Operands Definition

| Operand | Description |
|---------|---|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| RCX | Secure EPT entry architectural content – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page and may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. <ul style="list-style-type: none"> In case of successful operation, the requested entry's architectural content is returned. In case of EPT walk error, the architectural content of the Secure EPT entry where the error was detected is returned. In other cases, RCX returns 0. |
| RDX | Secure EPT entry level and state – see 3.6.2 <ul style="list-style-type: none"> 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 |

5.4.42.3. 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.

5 5.4.42.4. Operands Information

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

Table 5.171: TDH.MEM.WR Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|----------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | GPA | TD private memory | Blob | RW | Private | 8B | None | None | None |
| Explicit | RDX | HPA | TDR page | TDR | R | Opaque | 4KB | Shared | Shared | Shared |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | R | Opaque | N/A | Shared | N/A | N/A |
| Implicit | N/A | GPA | Secure EPT tree | N/A | R | Private | N/A | Shared | N/A | N/A |
| Implicit | N/A | GPA | Secure EPT entry | SEPT Entry | R | Private | N/A | Exclusive | N/A | N/A |

10 5.4.42.5. Completion Status Codes

Table 5.172: TDH.MEM.WR Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|-------------|
| TDX_EPT_ENTRY_NOT_PRESENT | |
| TDX_EPT_ENTRY_STATE_INCORRECT | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |

5.4.43. TDH.MIG.STREAM.CREATE Leaf

Create a Migration Stream and its MIGSC control structure.

5.4.43.1. Input Operands

Table 5.173: TDH.MIG.STREAM.CREATE Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 96 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | The physical address of a page where MIGSC will be created | | |
| RDX | The physical address of the owner TDR page (HKID bits must be 0) | | |

5

5.4.43.2. Output Operands

Table 5.174: TDH.MIG.STREAM.CREATE Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.43.3. Leaf Function Description

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

5.4.43.3.1. Overview

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.

15 TDH.MIG.STREAM.CREATE can only be successfully invoked if no migration session is in progress.

5.4.43.3.2. Enumeration

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

The maximum number of migration streams per TD is enumerated by MAX_MIGS.

20 5.4.43.3.3. Dynamic PAMT

If the TDX Module is configured for dynamic PAMT, the PAMT hierarchy can be built on demand. A TDX_MISSING_PAMT_PAGE_PAIR status indicates that a PAMT page pair is missing for the new MIGSC page. The host VMM may add it using TDH.PHYMEM.PAMT.ADD and retry the operation.

5.4.43.4. Operands Information

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

Table 5.175: TDH.MIG.STREAM.CREATE Operands Information Definition

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

5

5.4.43.5. Completion Status Codes

Table 5.176: TDH.MIG.STREAM.CREATE Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_MISSING_PAMT_PAGE_PAIR | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.MIG.STREAM.CREATE is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |

5.4.44. TDH.MNG.ADDCX Leaf

Add a TDCS physical page to a guest TD.

5.4.44.1. Input Operands

Table 5.177: TDH.MNG.ADDCX Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 1 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | The physical address of a page where TDCX will be added (HKID bits must be 0) | | |
| RDX | The physical address of the owner TDR page (HKID bits must be 0) | | |

5

5.4.44.2. Output Operands

Table 5.178: TDH.MNG.ADDCX Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.44.3. Leaf Function Description

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

5.4.44.3.1. Overview

TDH.MNG.ADDCX adds a TDCS physical page, which is a child of the specified TDR.

5.4.44.3.2. Dynamic PAMT

If the TDX Module is configured for dynamic PAMT, the PAMT hierarchy can be built on demand. A TDX_MISSING_PAMT_PAGE_PAIR status indicates that a PAMT page pair is missing for the new TDCX page. The host VMM may add it using TDH.PHYMEM.PAMT.ADD and retry the operation.

5.4.44.4. Operands Information

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

20

Table 5.179: TDH.MNG.ADDCX Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------|-------------|--------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ²⁸ |
| Explicit | RCX | HPA | TDCX page | Blob | RW | Opaque | 4KB | Exclusive | Shared | Shared | Exclusive |
| Explicit | RDX | HPA | TDR page | TDR | RW | Opaque | 4KB | Exclusive | Shared | Shared | None |

5.4.44.5. Completion Status Codes

Table 5.180: TDH.MNG.ADDCX Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_MISSING_PAMT_PAGE_PAIR | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_RND_NO_ENTROPY | Failed to generate a random migration encryption key. This is typically caused by an entropy error of the CPU's random number generator, and may be impacted by RDSEED, RDRAND or PCONFIG executing on other LPs. The operation should be retried. |
| TDX_SUCCESS | TDH.MNG.ADDCX is successful |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCX_NUM_INCORRECT | |

5

²⁸ ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

5.4.45. TDH.MNG.CREATE Leaf

Create a new guest TD and its TDR root page.

5.4.45.1. Input Operands

Table 5.181: TDH.MNG.CREATE Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 9 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | The physical address of a page where TDR will be created (HKID bits must be 0) | | |
| RDX | Bits | Name | Description |
| | 15:0 | HKID | The TD’s ephemeral private HKID |
| | 63:16 | Reserved | Reserved: must be 0 |

5

5.4.45.2. Output Operands

Table 5.182: TDH.MNG.CREATE Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.45.3. Leaf Function Description

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

5.4.45.3.1. Overview

TDH.MNG.CREATE starts the TD creation sequence. It creates a TDR page which is the root page of a new guest TD.

5.4.45.3.2. Dynamic PAMT

15 If the TDX Module is configured for dynamic PAMT, the PAMT hierarchy can be built on demand. A TDX_MISSING_PAMT_PAGE_PAIR status indicates that a PAMT page pair is missing for the new TDR page. The host VMM may add it using TDH.PHYMEM.PAMT.ADD and retry the operation.

5.4.45.4. Operands Information

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

Table 5.183: TDH.MNG.CREATE Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------|-------------|----------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ²⁹ |
| Explicit | RCX | HPA | TDR page | TDR | RW | Opaque | 4KB | Exclusive | Shared | Shared | Exclusive |
| Implicit | N/A | N/A | KOT | KOT | N/A | Hidden | N/A | Exclusive | N/A | N/A | None |

5.4.45.5. Completion Status Codes

Table 5.184: TDH.MNG.CREATE Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_HKID_NOT_FREE | |
| TDX_MISSING_PAMT_PAGE_PAIR | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_RND_NO_ENTROPY | Random TD_UUID generation (e.g., RDRAND or RDSEED) failed because the hardware random number generator did not have enough entropy. The host VMM should retry the operation. |
| TDX_SUCCESS | TDH.MNG.CREATE is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |

5

²⁹ ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

5.4.46. TDH.MNG.INIT Leaf

Initialize TD-scope control structures TDR and TDCS.

5.4.46.1. Input Operands

Table 5.185: TDH.MNG.INIT Input Operands Definition

| Operand | Description | | |
|---------|---|-------------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 21 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | TD handle and flags: | | |
| | Bits | Name | Description |
| | 0 | EVENT_FILTERING | Flags that performance monitoring events are filtered based on the EVENT_FILTER array specified by R8. Enumeration: Support of this flag is enumerated by TDX_FEATURES0.EVENT_FILTERING (bit 24). If not supported, its value must be 0. |
| | 11:1 | Reserved | Reserved: must be 0 |
| | 51:12 | TDR_HPA | Bits 51:12 of the host physical address of the parent TDR page (HKID bits must be 0) |
| | 63:52 | Reserved | Reserved: must be 0 |
| RDX | The physical address (including HKID bits) of an input TD_PARAMS_STRUCT | | |
| R8 | If RCX.EVENT_FILTERING is 0 or the configured ATTRIBUTES.PERFMON is 0, then R8 is ignored. Else, R8 provides the following information: | | |
| | Bits | Name | Description |
| | 11:0 | EVENT_FILTERS_NUM | The number of valid entries in the EVENT_FILTERS_ARRAY. Must be higher than 0 and lower or equal to MAX_EVENT_FILTERS, readable by TDH.SYS.RD*. |
| | 51:12 | EVENT_FILTERS_HPA | Bits 51:12 of the shared HPA (including HKID) of an array of EVENT_FILTER entries. The entry format and array restrictions are defined in 3.4.6. |
| | 63:52 | Reserved | Reserved: must be 0 |

5

5.4.46.2. Output Operands

Table 5.186: TDH.MNG.INIT Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |

| Operand | Description | | |
|---------|--|---------|---|
| RCX | Extended error information In case of a TD_PARAMS_STRUCT.CPUID_CONFIG error, RCX returns the applicable CPUID information as shown below. In all other cases, RCX returns 0. | | |
| | Bits | Name | Description |
| | 31:0 | LEAF | CPUID leaf number |
| | 63:32 | SUBLEAF | CPUID sub-leaf number: if sub-leaf is not applicable, value is -1 (0xFFFFFFFF). |
| Other | Unmodified | | |

5.4.46.3. 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.

5 **5.4.46.4. Leaf Function Description**

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

5.4.46.4.1. Overview

10 TDH.MNG.INIT initializes the TD-scope control structures TDR and TDCS based on a set of TD parameters provided as input.

5.4.46.4.2. Enumeration

Support of RCX.EVENT_FILTERING and the EVENT_FILTERS array is enumerated by TDX_FEATURES0.PERFMON_EVENT_FILTERING (bit 24), readable by TDH.SYS.RD*.

5.4.46.5. Operands Information

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

Table 5.187: TDH.MNG.INIT Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------------------|-------------|----------------|-----------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ³⁰ |
| Explicit | RCX | HPA | TDR page | TDR | RW | Opaque | 4KB | Exclusive | Shared | Shared | None |
| Explicit | RDX | HPA | TD Parameters | TD_PARAMS | R | Shared | 1024B | None | N/A | N/A | None |
| Explicit | R8 ³¹ | HPA | EVENT_FILTERS | EVENT_FILTER array | RW | Shared | 4KB | None | N/A | N/A | None |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Exclusive(i) | N/A | N/A | None |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Exclusive(i) | N/A | N/A | None |

³⁰ ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

³¹ Only if RCX.EVENT_FILTERING is set to 1

5.4.46.6. Completion Status Codes

Table 5.188: TDH.MNG.INIT Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_EVENT_FILTER_INVALID | |
| TDX_EVENT_FILTER_ORDER_INVALID | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.MNG.INIT is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |
| TDX_TDCX_NUM_INCORRECT | |

5.4.47. TDH.MNG.KEY.CONFIG Leaf

Configure the TD ephemeral private key on a single package.

5.4.47.1. Input Operands

Table 5.189: TDH.MNG.KEY.CONFIG Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 8 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | The physical address of a TDR page (HKID bits must be 0) | | |

5

5.4.47.2. Output Operands

Table 5.190: TDH.MNG.KEY.CONFIG Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.47.3. Leaf Function Description

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

5.4.47.3.1. Overview

TDH.MNG.KEY.CONFIG configures the TD's ephemeral private key on a single package.

5.4.47.3.2. Failure Conditions

15 TDH.MNG.KEY.CONFIG may fail due to the following conditions:

- A conflict with a concurrent TDH.MNG.KEY.CONFIG or a PCONFIG instruction running on the same package. In this case, a TDX_OPERAND_BUSY status is returned.
- Failure to generate a random key, typically caused by an entropy error of the CPU's random number generator, and may be impacted by RDSEED, RDRAND or PCONFIG executing on other LPs. In this case, a TDX_KEY_GENERATION_FAILED is returned.

20

5.4.47.3.3. 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.

5.4.47.4. Operands Information

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

Table 5.191: TDH.MNG.KEY.CONFIG Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|----------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | 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 |

5.4.47.5. Completion Status Codes

Table 5.192: TDH.MNG.KEY.CONFIG Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|---|
| TDX_KEY_CONFIGURED | |
| TDX_KEY_GENERATION_FAILED | Failed to generate a random key. This is typically caused by an entropy error of the CPU's random number generator, and may be impacted by RDSEED, RDRAND or PCONFIG executing on other LPs. The operation should be retried. |
| TDX_LIFECYCLE_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. Specifically, key configuration may fail due to a concurrently running PCONFIG instruction. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.MNG.KEY.CONFIG is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |

5

5.4.48. TDH.MNG.KEY.FREEID Leaf

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

5.4.48.1. Input Operands

Table 5.193: TDH.MNG.KEY.FREEID Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 20 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| 63:24 | Reserved | Must be 0 | |
| RCX | The physical address of a TDR page (HKID bits must be 0) | | |

5

5.4.48.2. Output Operands

Table 5.194: TDH.MNG.KEY.FREEID Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.48.3. Leaf Function Description

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

TDH.MNG.KEY.FREEID ends the platform cache flush sequence for the HKIDs associated with the specified TD after TDH.PHYMEM.CACHE.WB has been executed (unless that function is not required, as enumerated by TDX_FEATURES0.SKIP_PHYMEM_CACHE_WB (bit 34)) on all the required WBINVD domains. It marks the TD's HKIDs in KOT as free, and the TD itself as being torn down.

15

5.4.48.4. Operands Information

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

Table 5.195: TDH.MNG.KEY.FREEID Operands Information Definition

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

20

5.4.48.5. *Completion Status Codes*

Table 5.196: TDH.MNG.KEY.FREEID Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_LIFECYCLE_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.MNG.KEY.FREEID is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_WBCACHE_NOT_COMPLETE | |

5.4.49. TDH.MNG.KEY.RECLAIMID Leaf (Deprecated)

This function is deprecated; it is provided for backward compatibility.

5.4.49.1. Input Operands**Table 5.197: TDH.MNG.KEY.RECLAIMID Input Operands Definition**

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 27 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |

5

5.4.49.2. Output Operands**Table 5.198: TDH.MNG.KEY.RECLAIMID Output Operands Definition**

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.49.3. Leaf Function Description

10 **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.

5.4.49.4. Completion Status Codes

15

Table 5.199: TDH.MNG.KEY.RECLAIMID Completion Status Codes (Returned in RAX) Definition

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

5.4.50. TDH.MNG.RD Leaf

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

5.4.50.1. Input Operands

Table 5.200: TDH.MNG.RD Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|---|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 11 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Version number may be 0 or 1. See the enumeration details below. |
| | 63:24 | Reserved | Must be 0 |
| RCX | The physical address of a TDR page (HKID bits must be 0) | | |
| RDX | <p>Field identifier – see 3.10</p> <p>The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0.</p> <p>WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored.</p> <p>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.</p> | | |

5

5.4.50.2. Output Operands

Table 5.201: TDH.MNG.RD Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| RDX | <p>For TDH.MNG.RD version 0, RDX is unmodified.</p> <p>For TDH.MNG.RD version 1 or higher:</p> <ul style="list-style-type: none"> If the input field identifier was -1, RDX returns the first readable field identifier. <p>Else, in case of an error, RDX returns -1. On success, RDX returns the next readable field identifier. A value of -1 indicates no next field identifier is available.</p> |
| R8 | <p>Contents of the field</p> <p>In case of no success, as indicated by RAX, R8 returns 0.</p> |
| Other | Unmodified |

5.4.50.3. Leaf Function Description

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

5.4.50.3.1. Overview

TDH.MNG.RD reads a TD-scope metadata field (control structure field) of a TD.

If version 1 or higher is specified in RAX, RDX returns the next host-side readable field identifier. This may be used by the host VMM to dump the host readable TD metadata. To read all the available fields, the host VMM can invoke TDH.MNG.RD in a loop, starting with field identifier -1 as an input, until RDX returns -1. A status code of TDX_METADATA_FIELD_SKIP indicates that the returned value is not applicable.

5 5.4.50.3.2. Enumeration

Availability of TDH.MNG.RD version 1 is enumerated by TDX_FEATURES0.ENHANCED_METADATA (bit 3), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.MNG.RD with a version number higher than 0 returns a TDX_OPERAND_INVALID(RAX) status.

5.4.50.4. Operands Information

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

Table 5.202: TDH.MNG.RD Operands Information Definition

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

5.4.50.5. Completion Status Codes

15 **Table 5.203: TDH.MNG.RD Completion Status Codes (Returned in RAX) Definition**

| Completion Status Code | Description |
|--|---|
| TDX_METADATA_FIELD_ID_INCORRECT | |
| TDX_METADATA_FIELD_NOT_READABLE | |
| TDX_METADATA_FIELD_SKIP | Indicates that the field value being read is not applicable and needs to be skipped. If called in a loop, use RDX as the identifier of the next field to be read, if any. |
| TDX_METADATA_FIRST_FIELD_ID_IN_CONTEXT | Indicates that the first field ID in context is returned |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.MNG.RD is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |

| Completion Status Code | Description |
|----------------------------|-------------|
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |

5.4.51. TDH.MNG.VPFLUSHDONE Leaf

Start the TD teardown sequence by checking that none of the TD's VCPUs are associated with an LP.

5.4.51.1. Input Operands

Table 5.204: TDH.MNG.VPFLUSHDONE Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 19 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| 63:24 | Reserved | Must be 0 | |
| RCX | The physical address of a TDR page (HKID bits must be 0) | | |

5

5.4.51.2. Output Operands

Table 5.205: TDH.MNG.VPFLUSHDONE Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.51.3. 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 starts that TD teardown sequence. It 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.

For details, see the [Base Spec] chapter on TD Lifecycle Management.

5.4.51.4. Operands Information

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

Table 5.206: TDH.MNG.VPFLUSHDONE Operands Information

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|----------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | TDR page | TDR | RW | Opaque | 4KB | Exclusive | Shared | Shared |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Exclusive(i) | N/A | N/A |
| Implicit | N/A | N/A | KOT | KOT | N/A | Hidden | N/A | Exclusive | N/A | N/A |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | None | Opaque | N/A | Exclusive(i) | N/A | N/A |

5.4.51.5. Completion Status Codes

Table 5.207: TDH.MNG.VPFLUSHDONE Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_FLUSHVP_NOT_DONE | |
| TDX_IOMMU_IOTLB_TRACKING_NOT_DONE | Applicable only if the TDX Module supports TDX Connect. |
| TDX_LIFECYCLE_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.MNG.VPFLUSHDONE is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_HAS_ATTACHED_DEVICES | Applicable only if the TDX Module supports TDX Connect. |

5.4.52. TDH.MNG.WR Leaf

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

5.4.52.1. Input Operands

Table 5.208: TDH.MNG.WR Input Operands Definitions

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 13 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | The physical address of a TDR page (HKID bits must be 0) | | |
| RDX | Field identifier – see 3.10 The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0. WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored. | | |
| R8 | Data to write to the field | | |
| R9 | A 64b write mask to indicate which bits of the value in R8 are to be written to the field | | |

5

5.4.52.2. Output Operands

Table 5.209: TDH.MNG.WR Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| R8 | Previous content of the field In case of an error, as indicated by RAX, R8 returns 0. |
| Other | Unmodified |

5.4.52.3. Leaf Function Description

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

TDH.MNG.WR writes a TD-scope metadata field (control structure field) of a TD. The value (R8) is written as specified by the write mask (R9). Writing is subject to the field's internal write mask (per the TD's ATTRIBUTES.DEBUG bit). Writing of specific fields is also subject to additional rules.

15

Table 5.210: Metadata Field Write Rules

| Write Mask Bit in R9 | Internal Write Mask Bit | Value Bit in R8 |
|----------------------|-------------------------|------------------|
| 0 | N/A | Silently ignored |

| Write Mask Bit in R9 | Internal Write Mask Bit | Value Bit in R8 |
|----------------------|-------------------------|---|
| 1 | 0 | Must be the same as the current field's bit |
| 1 | 1 | Written to the current field's bit |

5.4.52.4. Operands Information

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

5

Table 5.211: TDH.MNG.WR Operands Information Definition

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

5.4.52.5. Completion Status Codes

Table 5.212: TDH.MNG.WR Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_METADATA_FIELD_ID_INCORRECT | |
| TDX_METADATA_FIELD_NOT_READABLE | |
| TDX_METADATA_FIELD_NOT_WRITABLE | |
| TDX_METADATA_FIELD_VALUE_NOT_VALID | |
| TDX_METADATA_WR_MASK_NOT_VALID | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.MNG.WR is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TD_NON_DEBUG | |
| TDX_TDCS_NOT_ALLOCATED | |

5.4.53. TDH.MR.EXTEND Leaf

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

5.4.53.1. Input Operands

Table 5.213: TDH.MR.EXTEND Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 16 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | The GPA of the TD page chunk to be measured | | |
| RDX | The physical address of the TDR page of the target TD (HKID bits must be 0) | | |

5

5.4.53.2. Output Operands

Table 5.214: TDH.MR.EXTEND Output Operands Definition

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

5.4.53.3. Leaf Function Description

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

5.4.53.3.1. Overview

TDH.MR.EXTEND updates the MRTD measurement register in TDCS with the measurement of the indicated 256 bytes chunk of a TD private page. For pages whose contents need to be measured, once the page is copied into the TD memory area by TDH.MEM.PAGE.ADD, the host VMM needs to call TDH.MR.EXTEND multiple times to measure the pages contents into MRTD. TDEXEND can be executed only before TDH.MR.FINALIZE.

15

Note: TDH.MR.EXTEND works on a 256B chunk of a page, not on a full page, due to instruction latency considerations.

5.4.53.3.2. TD Measurement Register (MRTD) Extension

TDH.MR.EXTEND extends the TD’s 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:

- 5 • 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.

5.4.53.3.3. MRTD Extension Compatibility Aspects

- 10 If MRTD was initialized by TDH.MNG.INIT, and later the TDX Module was updated before the current TDH.MR.EXTEND was called, there could be incompatibility issues with the intermediate MRTD context format between the original and the new TD module. On TDH.SYS.UPDATE, the host VMM can configure the TDX Module to detect such incompatibility. If detected, TDH.MR.EXTEND returns a TDX_INCOMPATIBLE_MRTD_CONTEXT status. In this case, the host VMM is expected to tear down the TD and rebuild it. For details, see the [Base Spec] section titled “Compatibility Aspects of TD-Preserving Update”.
- 15

5.4.53.4. Operands Information

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

Table 5.215: TDH.MR.EXTEND Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|--------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | GPA | TD private page chunk | Blob | R | Private | 256B | None | None | None |
| Explicit | RDX | HPA | TDR page | TDR | R | Opaque | 4KB | Exclusive | Shared | Shared |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | 4KB | Exclusive(i) | N/A | N/A |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Exclusive(i) | N/A | N/A |
| Implicit | N/A | GPA | Secure EPT tree | N/A | R | Private | N/A | Exclusive(i) | N/A | N/A |
| Implicit | N/A | GPA | Secure EPT entry | SEPT Entry | RW | Private | N/A | Exclusive(i) | N/A | N/A |

20

5.4.53.5. Completion Status Codes

Table 5.216: TDH.MR.EXTEND Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------|-------------|
| TDX_EPT_ENTRY_NOT_PRESENT | |
| TDX_EPT_WALK_FAILED | |
| TDX_INCOMPATIBLE_MRTD_CONTEXT | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.MR.EXTEND is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |

5.4.54. TDH.MR.FINALIZE Leaf

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

5.4.54.1. Input Operands

Table 5.217: TDH.MR.FINALIZE Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 17 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| 63:24 | Reserved | Must be 0 | |
| RCX | The physical address of the parent TDR page (HKID bits must be 0) | | |

5

5.4.54.2. Output Operands

Table 5.218: TDH.MR.FINALIZE Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.54.3. Leaf Function Description

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

5.4.54.3.1. Overview

TDH.MR.FINALIZE does the following:

- 15 • Completes the measurement of the initial TD contents (MRTD).
- Calculates SERVTD_HASH for bound or pre-bound Service TDs.
- Marks the TD as finalized so it can be executed.

5.4.54.3.2. MRTD Finalization Compatibility Aspects

20 If MRTD was initialized by TDH.MNG.INIT, and later the TDX Module was updated before the current TDH.MR.FINALIZE was called, there could be incompatibility issues with the intermediate MRTD context format between the original and the new TD module. On TDH.SYS.UPDATE, the host VMM can configure the TDX Module to detect such incompatibility. If detected, TDH.MR.FINALIZE returns a TDX_INCOMPATIBLE_MRTD_CONTEXT status. In this case, the host VMM is expected to tear down the TD and rebuild it. For details, see the [Base Spec] section titled “Compatibility Aspects of TD-Preserving Update”.

5.4.54.3.3. SERVTD_HASH Calculation

25 TDH.MR.FINALIZE calculates the TD’s SERVTD_HASH as follows:

1. Get all service TD binding slots whose SERVTD_BINDING_STATE is not NOT_BOUND.
 - 1.1. If no service TD binding slots apply, SERVTD_HASH value is 0.

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).
3. Concatenate SERVTD_INFO_HASH, SERVTD_TYPE and SERVTD_ATTR of each slot in a temporary buffer:
 - 3.1. SERVTD_INFO_HASH in bytes 47:0
 - 3.2. SERVTD_TYPE in bytes 49:48
 - 3.3. SERVTD_ATTR in bytes 57:50
4. Concatenate all temporary buffers.
5. Calculate SHA384 over the concatenated buffers.

5.4.54.4. Operands Information

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

Table 5.219: TDH.MR.FINALIZE Operands Information Definition

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

5.4.54.5. Completion Status Codes

Table 5.220: TDH.MR.FINALIZE Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_INCOMPATIBLE_MRTD_CONTEXT | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.MR.FINALIZE is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |

5.4.55. TDH.PHYMEM.CACHE.WB Leaf

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

If the value of TDX_FEATURES0.SKIP_PHYMEM_CACHE_WB (bit 34), readable by TDH.SYS.RD*, is 1, TDH.PHYMEM.CACHE.WB is provided for backward compatibility. It returns immediately, indicating success.

5.4.55.1. Input Operands

Table 5.221: TDH.PHYMEM.CACHE.WB Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|---|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 40 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | Command, as described below: | | |
| | Value | Name | Description |
| | 0 | WB_START_CMD | Start a new TDH.PHYMEM.CACHE.WB cycle with no cache invalidation. |
| | 1 | WB_RESUME_CMD | Resume a previously interrupted TDH.PHYMEM.CACHE.WB cycle with no cache invalidation. |
| | Other | | Reserved |

5.4.55.2. Output Operands

Table 5.222: TDH.PHYMEM.CACHE.WB Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.55.3. Leaf Function Description

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

5.4.55.3.1. Overview

TDH.PHYMEM.CACHE.WB writes back the cache hierarchy to memory and updates the KOT state to allow reuse of HKIDs.

5.4.55.3.2. Enumeration

If the value of TDX_FEATURES0.SKIP_PHYMEM_CACHE_WB (bit 34), readable by TDH.SYS.RD*, is 1, TDH.PHYMEM.CACHE.WB is provided for backward compatibility. It does nothing and returns immediately with TDX_SUCCESS.

5.4.55.3.3. Interruptibility

TDH.PHYMEM.CACHE.WB is interruptible. If a pending interrupt is detected during operation, TDH.PHYMEM.CACHE.WB returns with a TDX_INTERRUPTED_RESUMABLE status in RAX.

The host VMM initially calls TDH.PHYMEM.CACHE.WB with RCX indicating WB_START_CMD. If TDH.PHYMEM.CACHE.WB returns TDX_INTERRUPTED_RESUMABLE (or any other recoverable error) status in RAX, the host VMM should call TDH.PHYMEM.CACHE.WB with RCX indicating WB_RESUME_CMD in a loop until it completes its operation, as indicated by TDX_SUCCESS status.

5.4.55.3.4. Host VMM Should Always Resume an Interrupted TDH.PHYMEM.CACHE.WB

When TDH.PHYMEM.CACHE.WB is interrupted, the CPU still considers this as a cache write-back operation in progress. The host VMM should complete the cache write-back operation by resuming TDH.PHYMEM.CACHE.WB (i.e., with RCX indicating WB_RESUME_CMD) until completed successfully. **Failure to do so will result in memory performance impact that would only be resolved by a restart.** The host VMM should also refrain from executing WBINVD or WBNOINVD while the TDH.PHYMEM.CACHE.WB cycle is in progress.

5.4.55.3.5. Other TDH.PHYMEM.CACHE.WB Characteristics

- TDH.PHYMEM.CACHE.WB does not invalidate cache lines.
- The function operates on cache lines associated with any HKID.
- The function is designed to ensure write back of at least those cache lines where the state of that HKID (in the KOT) was HKID_FLUSHED at the time of the first invocation (RCX == WB_START_CMD).
- Depending on the implementation, the instruction may write back additional cache lines.
- The scope at which TDH.PHYMEM.CACHE.WB operates (e.g., package or core) is determined at Intel TDX Module initialization time.

5.4.55.4. Operands Information

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

Table 5.223: TDH.PHYMEM.CACHE.WB (Implicit) Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|--------------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Implicit | N/A | N/A | 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 |

5.4.55.5. Completion Status Codes

Table 5.224: TDH.PHYMEM.CACHE.WB Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------------|--|
| TDX_INTERRUPTED_RESUMABLE | TDH.PHYMEM.CACHE.WB was interrupted; it is recommended to resume it with RCX indicating WB_RESUME_CMD |
| TDX_NO_HKID_READY_TO_WBCACHE | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |

| Completion Status Code | Description |
|------------------------|------------------------------------|
| TDX_SUCCESS | TDH.PHYMEM.CACHE.WB is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |

5.4.56. TDH.PHYMEM.PAGE.RDMD Leaf

Read the metadata of a page, of containing large page or of an HPA range in TDMR.

5.4.56.1. Input Operands**Table 5.225: TDH.PHYMEM.PAGE.RDMD Operands**

| Operand | Description | | |
|---------|--|---------------------|---|
| RAX | SEAMCALL instruction leaf number and version | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 24 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0. |
| | 63:24 | Reserved | Must be 0 |
| RCX | HPA range mapping information: | | |
| | Bits | Name | Description |
| | 2:0 | MIN_SIZE | Minimum size of the page 0: 4KB 1: 2MB 2: 1GB MIN_SIZE values other than 4KB are only allowed if the TDX Module is configured for dynamic PAMT. They allow the caller to read the metadata stored by non-leaf PAMT entries. |
| | 11:3 | Reserved | Reserved: must be 0 |
| | 51:12 | HPA | Bits 51:12 of a physical address of a range in TDMR (HKID bits must be 0). Depending on the requested MIN_SIZE, the following bits must be 0: 0 (4KB): None 1 (2MB): Bits 20:12 2 (1GB): Bits 29:12 |
| 63:52 | Reserved | Reserved: must be 0 | |

5

5.4.56.2. Output Operands**Table 5.226: TDH.PHYMEM.PAGE.RDMD Output Operands Definition**

| Operand | Description | | |
|---------|--|------|-------------|
| RAX | SEAMCALL instruction return code – see 5.4.1 | | |
| RCX | Bits | Name | Description |
| | | | |

| Operand | Description | | |
|---------|---|-------------|---|
| | 3:0 | PT | <p>Page Type – see 3.5.1</p> <p>If the TDX Module supports Dynamic PAMT, PT_PAMT is returned in two cases:</p> <ul style="list-style-type: none"> If NON_LEAF (see below) is 0, it means that the specified HPA is of a PAMT page. In this case, then the ACTUAL_SIZE returned in R8 is 0 (4KB). If NON_LEAF (see below) is 1, it means that the specified HPA is of a non-leaf PAMT entry. In this case, then the ACTUAL_SIZE returned in R8 is 1 (2MB). Note that this only happens if the specified MIN_SIZE was also 1 (2MB). <p>In case of an error, PT returns 0.</p> |
| | 62:4 | Reserved | Set to 0 |
| | 63 | NON_LEAF | <p>If the TDX Module is configured for dynamic PAMT, NON_LEAF Indicates that the applicable PAMT entry is a non-leaf entry.</p> <p>Else, NON_LEAF returns 0.</p> <p>In case of an error, NON_LEAF returns 0.</p> |
| RDX | <p>For PT values associated with a specific TD, RDX returns the HPA of the TD's TDR control structure page, if applicable (HKID bits are set to 0).</p> <p>If the TDX Module supports TDX Connect, then for a PT value of PT_IOMMU_MT, this field returns the IOMMU_ID. See the [TDX Connect ABI].</p> <p>If the TDX Module supports Dynamic PAMT, then if RCX.PT returns PT_PAMT, RDX returns 0.</p> <p>In multiple error cases, as indicated by RAX, RDX returns 0. In other error cases, RDX still returns the OWNER information. See the completion status codes table below for details.</p> | | |
| R8 | Bits | Name | Description |
| | 2:0 | ACTUAL_SIZE | <p>In case of an error, as indicated by RAX, this field returns 0.</p> <p>If the requested MIN_SIZE was 0 (4KB), this field returns the actual size of the containing 4KB, 2MB or 1GB page, depending on the page mapping.</p> <p>If the requested MIN_SIZE was 1 (2MB), this field returns the actual size of the containing 2MB or 1GB page, depending on the page mapping.</p> <p>If the requested MIN_SIZE was 2 (1GB), this field returns 2 (1GB).</p> |
| | 63:3 | Reserved | Set to 0 |
| R9 | <p>For a PT value of PT_REG, PT_EPT or PT_IOMMU_MT, R9 returns BEPOCH.</p> <p>If the TDX Module is configured for dynamic PAMT, RCX.NON_LEAF is 1, indicating a non-leaf PAMT entry, R9 returns the number of entries in the child PAMT pages which are neither free nor reserved.</p> <p>In other cases, R9 returns 0.</p> | | |
| R10 | <p>If the TDX Module is configured for dynamic PAMT, RCX.NON_LEAF is 1, indicating a non-leaf PAMT entry, R10 returns the HPA of the first PAMT page pointed by this entry. HKID bits are set to 0.</p> <p>Else, R10 returns 0.</p> | | |
| R11 | <p>If the TDX Module is configured for dynamic PAMT, RCX.NON_LEAF is 1, indicating a non-leaf PAMT entry, R11 returns the HPA of the second PAMT page pointed by this entry. HKID bits are set to 0.</p> <p>Else, R11 returns 0.</p> | | |

| Operand | Description |
|---------|-------------|
| Other | Unmodified |

5.4.56.3. Leaf Function Description

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

5.4.56.3.1. Overview

TDH.PHYMEM.PAGE.RDMD finds the containing page or HPA range (4KB, 2MB or 1GB) of the given page in TDMR and reads its metadata from its PAMT entry.

5.4.56.3.2. Enumeration

Support of requested HPA range sizes other than 0 (4KB) is enumerated by TDX_FEATURES0.DYNAMIC_PAMT (bit 36), readable by the host VMM using TDH.SYS.RD*.

5.4.56.4. Operands Information

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

Table 5.227: TDH.PHYMEM.PAGE.RDMD Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|----------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | Target page | Blob | None | Opaque/ Private | 4KB | Shared | Shared | Shared |

15

5.4.56.5. Completion Status Codes

Table 5.228: TDH.PHYMEM.PAGE.RDMD Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------------|--|
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDH.PHYMEM.PAGE.RDMD is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |

5.4.57. 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.

5.4.57.1. Input Operands

5

Table 5.229: TDH.PHYMEM.PAGE.RECLAIM Input Operands Definition

| Operand | Name | Description | | |
|---------|------------------|--|----------------|---|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 28 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 or 1, see enumeration details below. |
| | 63:24 | Reserved | Must be 0 | |
| RCX | PAGE | The physical address of a 4KB, 2MB or 1GB page to be reclaimed (HKID bits must be 0) | | |

5.4.57.2. Output Operands**Table 5.230: TDH.PHYMEM.PAGE.RECLAIM Output Operands Definition**

| Operand | Name | Description | | |
|---------|--------|---|----------|---|
| RAX | STATUS | SEAMCALL instruction return code – see 5.4.1 | | |
| RCX | PT | Page Type (PT) – see 3.5.1 In multiple error cases, as indicated by RAX, RCX returns 0. In other error cases, RCX still returns the PT information. See the completion status codes table below for details. | | |
| RDX | OWNER | For most PT values, this field returns the HPA of the TD's TDR control structure page, if applicable (HKID bits are set to 0). For PT value of PT_IOMMU_MT (applicable only if the TDX Module supports TDX Connect), this field returns the IOMMU_ID. See the [TDX Connect ABI]. In multiple error cases, as indicated by RAX, RDX returns 0. In other error cases, RDX still returns the OWNER information. See the completion status codes table below for details. | | |
| R8 | SIZE | Bits | Name | Description |
| | | 2:0 | Size | Size of the containing 4KB, 2MB or 1GB page |
| | | 63:3 | Reserved | Set to 0 |
| | | In multiple error cases, as indicated by RAX, RDX returns 0. In other error cases, RDX still returns the size information. See the completion status codes table below for details. | | |
| R9 | FLAGS | Bits | Name | Description |
| | | 61:0 | Reserved | Set to 0 |

| Operand | Name | Description | | |
|---------|----------|--------------------|-------------------|--|
| | | 62 | PAMT_REMOVAL_HINT | If the TDX Module has been configured for Dynamic PAMT, this bit indicates that all entries in the PAMT page pair mapping the removed page are free. This only applies to 4KB pages. Else, this bit is set to 0 |
| | | 63 | Reserved | Set to 0 |
| R10 | RESERVED | Reserved: set to 0 | | |
| R11 | RESERVED | Reserved: set to 0 | | |
| Other | | Unmodified | | |

5.4.57.3. Leaf Function Description

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

5 5.4.57.3.1. Overview

TDH.PHYMEM.PAGE.RECLAIM reclaims a TD-owned physical page from the TD.

5.4.57.3.2. Enumeration

Support of TDH.PHYMEM.PAGE.RECLAIM version 1 is enumerated by TDX_FEATURES0.ACT (bit 14), readable by the host VMM using TDH.SYS.RD*.

10 Support for reclaiming PT_TR pages is enumerated by TDX_FEATURES0.NON_BLOCKING_RESIZE (bit 35).

5.4.57.3.3. Owner TD Lifecycle State

Normally, TDH.PHYMEM.PAGE.RECLAIM can reclaim pages only if the owner TD is in the TD_TEARDOWN state. The only exception are PT_TR pages (see below).

5.4.57.3.4. Reclaiming PT_TR Pages

15 If supported, PT_TR pages are created by TDH.MEM.PAGE.PROMOTE if the NO_TRACK option is selected (see 5.4.26).

Normally, TDH.PHYMEM.PAGE.RECLAIM can reclaim pages only if the owner TD is in the TD_TEARDOWN state. However, if reclaiming PT_TR pages is supported, a PT_TR page may be reclaimed while the TD is in its normal operating state (TD_KEYS_CONFIGURED). In that case, if the TD may be running, the function checks the TLB tracking of the reclaimed PT_TR page.

20 5.4.57.3.5. Dynamic PAMT

If the TDX Module is configured for dynamic PAMT and the removed page size is 4KB, R9[62] is returned as a hint indicating that all entries in the PAMT page pair mapping the removed physical page are free, and they can be removed using TDH.PHYMEM.PAMT.REMOVE.

5.4.57.3.6. Reclaimed Page Initialization

25 After the page has been reclaimed, the host VMM should initialize its content before it is reused as a non-private page, as described in the [Base Spec].

5.4.57.4. Operands Information

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

Table 5.231: TDH.PHYMEM.PAGE.RECLAIM Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------|-------------|------------------------------|------------------|--------|---------------------|--------------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ³² |
| Explicit | RCX | HPA | Target page | Blob | RW | Opaque/ Private | 4KB, 2MB or 1GB | Exclusive | Shared | Shared | Exclusive |
| Implicit | N/A | N/A | TDR page ³³ | TDR | RW | Opaque | 4KB | Shared | N/A | N/A | None |
| Implicit | N/A | N/A | TDCS structure ³⁴ | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A | None |
| Implicit | N/A | N/A | TDCS.OP_STATE ³⁴ | OP_STATE | RW | Opaque | N/A | Shared | N/A | N/A | None |

5.4.57.5. Completion Status Codes

Table 5.232: TDH.PHYMEM.PAGE.RECLAIM Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|---|
| TDX_IOMMU_MT_PAGE_IN_USE | Applies only if the TDX Module supports TDX Connect |
| TDX_LIFECYCLE_STATE_INCORRECT | RCX, RDX and R8 return the actual PT, OWNER and SIZE information. |
| TDX_OP_STATE_INCORRECT | Attempted reclaim of PT_TR while the TD's OP_STATE doesn't allow it |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. If the page is not a TDR page but the owner TDR is busy, then RCX, RDX and R8 return the actual PT, OWNER and SIZE information. |
| TDX_OPERAND_INVALID | If the page physical address is not aligned on its size, then RCX, RDX and R8 return the actual PT, OWNER and SIZE information. |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.PHYMEM.PAGE.RECLAIM is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_ASSOCIATED_PAGES_EXIST | RCX, RDX and R8 return the actual PT, OWNER and SIZE information. |
| TDX_TD_FATAL | Attempted reclaim of a PT_TR page while the TD is in a FATAL state but not in TD_TEARDOWN state |

³² ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

³³ Except when TDR is the target page

³⁴ Only if the TDX Module supports NON_BLOCKING_RESIZE, and TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED

| Completion Status Code | Description |
|---------------------------|--|
| TDX_TDCS_NOT_ALLOCATED | |
| TDX_TLB_TRACKING_NOT_DONE | Applicable only if reclaiming PT_TR pages, enumerated by TDX_FEATURES0.NON_BLOCKING_RESIZE (bit 35), is supported. |

5.4.58. TDH.PHYMEM.PAGE.WBINVD Leaf

Write back and invalidate all cache lines associated with the specified non-TDX 4KB memory page and HKID.

5.4.58.1. Input Operands**Table 5.233: TDH.PHYMEM.PAGE.WBINVD Input Operands Definition**

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 41 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | Physical address (including HKID bits) of a 4KB page in TDMR which is not allocated to TDX | | |

5

5.4.58.2. Output Operands**Table 5.234: TDH.PHYMEM.PAGE.WBINVD Output Operands Definition**

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.58.3. 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.

5.4.58.4. Operands Information

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

20

Table 5.235: TDH.PHYMEM.PAGE.WBINVD Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|-------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | Target page | Blob | R | Private/ Opaque | 4KB | Shared | Shared | Shared |

5.4.58.5. Completion Status Codes**Table 5.236: TDH.PHYMEM.PAGE.WBINVD Completion Status Codes (Returned in RAX) Definition**

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.PHYMEM.PAGE.WBINVD is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |

5.4.59. TDH.PHYMEM.PAMT.ADD Leaf

Add a pair of physical 4KB PAMT pages which map a given HPA range.

5.4.59.1. Input Operands**Table 5.237: TDH.PHYMEM.PAMT.ADD Input Operands Definition**

| Operand | Description | | |
|---------|--|---------------------|--|
| RAX | SEAMCALL instruction leaf number and version | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 58 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0. |
| | 63:24 | Reserved | Must be 0 |
| RCX | HPA range mapping information: | | |
| | Bits | Name | Description |
| | 2:0 | Level | Level of the non-leaf PAMT entry mapping the PAMT page pair to be added Must be 1 (2MB). |
| | 11:3 | Reserved | Reserved: must be 0 |
| | 51:12 | HPA | Bits 51:12 of the HPA range for which the PAMT page pair is to be added (HKID bits must be 0) Bits 20:12 must be 0. |
| 63:52 | Reserved | Reserved: must be 0 | |
| RDX | The physical address of the first new PAMT page to be added (HKID bits must be 0) | | |
| R8 | The physical address of the second new PAMT page to be added (HKID bits must be 0) | | |

5

5.4.59.2. Output Operands**Table 5.238: TDH.PHYMEM.PAMT.ADD Output Operands Definition**

| Operand | Name | Description |
|---------|--------|----------------------------------|
| RAX | STATUS | SEAMCALL instruction return code |
| Other | | Unmodified |

5.4.59.3. Leaf Function Description

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

5.4.59.3.1. Overview

TDH.PHYMEM.PAMT.ADD adds a pair of PAMT pages that map an HPA range at the specified level.

5.4.59.3.2. Enumeration

Support of TDH.PHYMEM.PAMT.ADD is enumerated by TDX_FEATURES0.DYNAMIC_PAMT (bit 36), readable by the host VMM using TDH.SYS.RD*.

5.4.59.4. Operands Information

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

Table 5.239: TDH.PHYMEM.PAMT.ADD Operands Information Definition

| Explicit/ Implicit | Reg. | Ref. Type | Resource | Resource Type | Access | Access Semantics | Align. Check | Concurrency Restrictions | | | |
|-----------------------|------------|--------------|-------------------------------|------------------|--------|---------------------|-----------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ³⁵ |
| Explicit | RCX | HPA | 2MB HPA range to map | N/A | RW | Opaque | 2MB | Exclusive ³⁶ | N/A | Shared | None |
| Explicit | R8, RDX | HPA | New PAMT pages | PAMT_PAGE | RW | Opaque | 4KB | Shared | Shared | Shared | Exclusive |

5.4.59.5. Completion Status Codes

10

Table 5.240: TDH.PHYMEM.PAMT.ADD Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------------|-------------|
| TDX_HPA_RANGE_NOT_FREE | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | |
| TDX_OPERAND_INVALID | |
| TDX_PAGE_METADATA_INCORRECT | |
| TDX_PAGE_NOT_FREE | |
| TDX_SUCCESS | |

³⁵ ACT is implemented only on client platforms. This is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

³⁶ Concurrency is enforced by locking the parent non-leaf PAMT entry in the PAMT tree, which points to the new PAMT pages.

5.4.60. TDH.PHYMEM.PAMT.REMOVE Leaf

Remove a pair of physical 4KB PAMT pages which map a given HPA range.

5.4.60.1. Input Operands**Table 5.241: TDH.PHYMEM.PAMT.REMOVE Input Operands Definition**

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 59 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0. |
| | 63:24 | Reserved | Must be 0 |
| RCX | HPA range mapping information: | | |
| | Bits | Name | Description |
| | 2:0 | Level | Level of the non-leaf PAMT entry mapping the PAMT page pair to be removed Must be 1 (2MB). |
| | 11:3 | Reserved | Reserved: must be 0 |
| | 51:12 | HPA | Bits 51:12 of the HPA range for which the PAMT page pair is to be removed (HKID bits must be 0) Bits 20:12 must be 0. |
| | 63:52 | Reserved | Reserved: must be 0 |

5

5.4.60.2. Output Operands**Table 5.242: TDH.PHYMEM.PAMT.REMOVE Output Operands Definition**

| Operand | Name | Description |
|---------|--------------|---|
| RAX | STATUS | SEAMCALL instruction return code |
| RDX | REMOVED_HPA0 | If TDH.PHYMEM.PAMT.REMOVE completed successfully, RDX returns the HPA (HKID bits are 0) of the first removed 4KB PAMT page. Else, RDX is unmodified. |
| R8 | REMOVED_HPA1 | If TDH.PHYMEM.PAMT.REMOVE completed successfully, R8 returns the HPA (HKID bits are 0) of the second removed 4KB PAMT page. Else, R8 is unmodified. |
| Other | | Unmodified |

5.4.60.3. Leaf Function Description

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

5.4.60.3.1. Overview

TDH.PHYMEM.PAMT.REMOVE first checks that all 512 entries in the PAMT page pair to be removed are free (PT_NDA). It then removes the PAMT page pair and marks its parent PAMT entry as PAMT_NDA.

5.4.60.3.2. Enumeration

- 5 Support of TDH.PHYMEM.PAMT.REMOVE is enumerated by TDX_FEATURES0.DYNAMIC_PAMT (bit 36), readable by the host VMM using TDH.SYS.RD*.

5.4.60.4. Operands Information

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

10 **Table 5.243: TDH.PHYMEM.PAMT.REMOVE Operands Information Definition**

| Explicit/ Implicit | Reg. | Ref. Type | Resource | Resource Type | Access | Access Semantics | Align. Check | Concurrency Restrictions | | | |
|-----------------------|------|--------------|--------------------------|------------------|--------|---------------------|-----------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ³⁷ |
| Explicit | RCX | HPA | HPA range | N/A | RW | Opaque | 2MB | Exclusive ³⁸ | N/A | Shared | None |
| Implicit | N/A | HPA | Removed PAMT pages | PAMT_PAGE | RW | Opaque | None | Exclusive(i) | None | None | Exclusive |

5.4.60.5. Completion Status Codes

Table 5.244: TDH.PHYMEM.PAMT.REMOVE Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-----------------------------|-------------|
| TDX_NO_PAMT_PAGE_PAIR | |
| TDX_OPERAND_ADD_RANGE_ERROR | |
| TDX_OPERAND_BUSY | |
| TDX_OPERAND_INVALID | |
| TDX_PAMT_PAGE_NOT_EMPTY | |
| TDX_SUCCESS | |

³⁷ ACT is implemented only on client platforms. This is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

³⁸ Concurrency is enforced by locking the parent non-leaf PAMT entry in the PAMT tree, which points to the new PAMT pages.

5.4.61. TDH.SERVTD.BIND Leaf

Bind a service TD to a target TD.

5.4.61.1. Input Operands

Table 5.245: TDH.SERVTD.BIND Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 48 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | The physical address of the target TD's TDR page (HKID bits must be 0) | | |
| RDX | The physical address of the service TD's TDR page (HKID bits must be 0) | | |
| R8 | Index (slot number) in the target TD's service TD binding table | | |
| R9 | SERVTD_TYPE: Service TD type | | |
| R10 | SERVTD_ATTR: Service TD attributes | | |

5

5.4.61.2. Output Operands

Table 5.246: TDH.SERVTD.BIND Output Operands Definition

| Operand | Description |
|-------------------------------------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| RCX | Binding handle In case of an error, as indicated by RAX, RCX returns 0. |
| R10 | TD_UUID bits 63:0 In case of an error, as indicated by RAX, R10 returns 0. |
| R11 | TD_UUID bits 127:64 In case of an error, as indicated by RAX, R11 returns 0. |
| R12 | TD_UUID bits 191:128 In case of an error, as indicated by RAX, R12 returns 0. |
| R13 | TD_UUID bits 255:192 In case of an error, as indicated by RAX, R13 returns 0. |
| AVX, AVX2 and AVX512 state | May be reset to the architectural RESET state |
| Other | Unmodified |

5.4.61.3. Leaf Function Description

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

5 5.4.61.3.1. Overview

TDH.SERVTD.BIND binds a service TD to a target TD.

5.4.61.3.2. Enumeration

Availability of TDH.SERVTD.BIND is enumerated by TDX_FEATURES0.SERVICE_TD (bit 2), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.SERVTD.BIND returns a TDX_OPERAND_INVALID(RAX) status.

10 5.4.61.4. Operands Information

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

Table 5.247: TDH.SERVTD.BIND Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|-----------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | Target TD's TDR page | TDR | R | Opaque | 4KB | Shared | Shared | Shared |
| Explicit | RDX | HPA | Service TD's TDR page | TDR | R | Opaque | 4KB | Shared | Shared | Shared |
| Implicit | N/A | N/A | Target TD's TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A |
| Implicit | N/A | N/A | Target TD's TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared | N/A | N/A |
| Implicit | N/A | N/A | Binding table | | RW | Opaque | N/A | Exclusive | None | None |
| Implicit | N/A | N/A | Service TD's TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A |
| Implicit | N/A | N/A | Service TD's TDCS.OP_STATE | OP_STATE | R | Opaque | N/A | Shared | N/A | N/A |
| Implicit | N/A | N/A | Service TD's TDCS.RTMR | SHA384_HASH | N/A | Opaque | N/A | Shared | N/A | N/A |

15 5.4.61.5. Completion Status Codes

Table 5.248: TDH.SERVTD.BIND Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------------|--|
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |

| Completion Status Code | Description |
|-------------------------------------|--------------------------------|
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SERVTD_ATTR_MISMATCH | |
| TDX_SERVTD_INFO_HASH_MISMATCH | |
| TDX_SERVTD_TYPE_MISMATCH | |
| TDX_SERVTD_UUID_MISMATCH | |
| TDX_SUCCESS | TDH.SERVTD.BIND is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |

5.4.62. TDH.SERVTD.REBIND Leaf

Rebind a new service TD to a target TD.

5.4.62.1. Input Operands

Table 5.249: TDH.SERVTD.REBIND Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 97 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | The physical address of the target TD's TDR page (HKID bits must be 0) | | |
| RDX | The physical address of the service TD's TDR page (HKID bits must be 0) | | |
| R8 | Index (slot number) in the target TD's service TD binding table | | |
| R9 | SERVTD_TYPE: Service TD type | | |
| R10 | SERVTD_ATTR: Service TD attributes | | |

5

5.4.62.2. Output Operands

Table 5.250: TDH.SERVTD.REBIND Output Operands Definition

| Operand | Description |
|----------------------------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| RCX | Binding handle In case of an error, as indicated by RAX, RCX returns 0. |
| R10 | TD_UUID bits 63:0 In case of an error, as indicated by RAX, R10 returns 0. |
| R11 | TD_UUID bits 127:64 In case of an error, as indicated by RAX, R11 returns 0. |
| R12 | TD_UUID bits 191:128 In case of an error, as indicated by RAX, R12 returns 0. |
| R13 | TD_UUID bits 255:192 In case of an error, as indicated by RAX, R13 returns 0. |
| AVX, AVX2 and AVX512 state | May be reset to the architectural RESET state |
| Other | Unmodified |

5.4.62.3. Leaf Function Description

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

5.4.62.3.1. Overview

- 5 TDH.SERVTD.REBIND binds a new service TD to a target TD. Current bound service TD must approve the transition to the new service TD by calling the TDG.SERVTD.REBIND.APPROVE API prior to this call.

5.4.62.3.2. Enumeration

Availability of TDH.SERVTD.REBIND is enumerated by TDX_FEATURES0.SERVTD_REBIND (bit 48), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.SERVTD.REBIND returns a TDX_OPERAND_INVALID(RAX) status.

10 5.4.62.4. Operands Information

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

Table 5.251: TDH.SERVTD.REBIND Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|-----------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | Target TD's TDR page | TDR | R | Opaque | 4KB | Shared | Shared | Shared |
| Explicit | RDX | HPA | Service TD's TDR page | TDR | R | Opaque | 4KB | Shared | Shared | Shared |
| Implicit | N/A | N/A | Target TD's TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A |
| Implicit | N/A | N/A | Target TD's TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared | N/A | N/A |
| Implicit | N/A | N/A | Binding table | | RW | Opaque | N/A | Exclusive | None | None |
| Implicit | N/A | N/A | Service TD's TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A |
| Implicit | N/A | N/A | Service TD's TDCS.OP_STATE | OP_STATE | R | Opaque | N/A | Shared | N/A | N/A |
| Implicit | N/A | N/A | Service TD's TDCS.RTMR | SHA384_HASH | N/A | Opaque | N/A | Shared | N/A | N/A |

15 5.4.62.5. Completion Status Codes

Table 5.252: TDH.SERVTD.REBIND Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------------|--|
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_REBIND_TOKEN_MISMATCH | Rebind session tokens did not match |
| TDX_SERVTD_ATTR_MISMATCH | |
| TDX_SERVTD_EXT_MISMATCH | Rebind SERVTD_EXT hash did not match |
| TDX_SERVTD_INCORRECT_BINDING_STATE | Target TD binding state is not in the expected state |
| TDX_SERVTD_INFO_HASH_MISMATCH | |
| TDX_SERVTD_TYPE_MISMATCH | |
| TDX_SERVTD_UUID_MISMATCH | |
| TDX_SUCCESS | TDH.SERVTD.REBIND is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |

5.4.63. TDH.SERVTD.PREBIND Leaf

Pre-bind a service TD to a target TD.

5.4.63.1. Input Operands**Table 5.253: TDH.SERVTD.PREBIND Input Operands Definition**

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 49 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | The physical address of the target TD's TDR page (HKID bits must be 0) | | |
| RDX | The physical address (including HKID bits) of SERVTD_INFO_HASH, the expected SHA384 hash of the service TD's TDINFO_STRUCT | | |
| R8 | Index (slot number) in the target TD's service TD binding table | | |
| R9 | SERVTD_TYPE: Expected service TD type | | |
| R10 | SERVTD_ATTR: Expected service TD attributes | | |

5

5.4.63.2. Output Operands**Table 5.254: TDH.SERVTD.PREBIND Output Operands Definition**

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.63.3. Leaf Function Description

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

5.4.63.3.1. Overview

TDH.SERVTD.PREBIND pre-binds a service TD to a target TD, by setting its expected binding parameters.

5.4.63.3.2. Enumeration

15 Availability of TDH.SERVTD.PREBIND is enumerated by TDX_FEATURES0.SERVICE_TD (bit 2), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.SERVTD.PREBIND returns a TDX_OPERAND_INVALID(RAX) status.

5.4.63.4. Operands Information

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

Table 5.255: TDH.SERVTD.PREBIND Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|----------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | Target TD's TDR page | TDR | R | Opaque | 4KB | Shared | Shared | Shared |
| Explicit | RDX | HPA | SERVTD_INFO_HASH | SHA384_HASH | R | Shared | 64B | N/A | N/A | N/A |
| Implicit | N/A | N/A | Target TD's TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A |
| Implicit | N/A | N/A | Target TD's TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared | N/A | N/A |
| Implicit | N/A | N/A | Binding table | | RW | Opaque | N/A | Exclusive | None | None |

5.4.63.5. Completion Status Codes

Table 5.256: TDH.SERVTD.PREBIND Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SERVTD_ALREADY_BOUND_FOR_TYPE | |
| TDX_SUCCESS | TDH.SERVTD.PREBIND is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |

5.4.64. TDH.SYS.CONFIG Leaf

Unreleased Feature: Some of the text in this section is related to Non-Blocking Export, a feature which has not been released yet at the time of writing of this document. Details related to that feature serve as a preview and are subject to change.

- 5 Globally configure the Intel TDX Module.

5.4.64.1. Input Operands

Table 5.257: TDH.SYS.CONFIG Input Operands Definition

| Operand | Name | Description | | |
|---------|-------------------|--|----------------|---|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 45 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Versions 0 and 1 are supported. See enumeration details below. |
| | | 63:24 | Reserved | Must be 0 |
| RCX | TDMR_INFO_HPA | The physical address of an array of pointers, each containing the physical address of a single TDMR_INFO entry (see 3.3.3). The pointer array must be sorted such that TDMR base addresses (TDMMR_INFO.TDMR_BASE) are sorted from the lowest to the highest base address, and TDMRs do not overlap with each other. | | |
| RDX | TDMMR_INFO_NUM | The number of pointers in the above buffer, between 1 and 64 | | |
| R8 | INIT_PARAMS | Bits | Name | Description |
| | | 15:0 | HKID | TDX Module's global private HKID value |
| | | 16 | DYNAMIC_PAMT | Selects whether the TDX Module will use a static or dynamic allocation of PAMT 0: Static PAMT 1: Dynamic PAMT Enumeration: Support of Dynamic PAMT is enumerated by TDMMR_FEATURES0.DYNAMIC_PAMT (bit 36), readable by TDH.SYS.RD*. |
| | | 63:17 | Reserved | Reserved: must be 0 |
| R9 | FEATURES_ENABLED0 | If the requested version in RAX is 1 or higher, R9 specifies TDX Module feature enabling flags, formatted similarly to TDMMR_FEATURES0, readable by TDH.SYS.RD*. A bit may be set to 1 if the corresponding TDMMR_FEATURES0 bit is 1. Else, R9 is ignored. | | |

| | | | | |
|-----|------------------|---|---------------------|--|
| | | 6 | TDX_CONNECT | Enables TDX Connect Enumeration: Support of TDX Connect is enumerated by TDX_FEATURES0.TDX_CONNECT (bit 6), readable by TDH.SYS.RD*. |
| | | 41 | NON_BLOCKING_EXPORT | Controls TD export mode: 0: Write-blocking based export 1: Non-blocking export |
| | | Bits whose value in TDX_FEATURES0 is 0 | | Must be 0 |
| | | Other bits | | Ignored |
| R10 | FEATURES_ENABLE1 | If the requested version in RAX is 1 or higher, R10 is reserved for additional TDX Module feature enabling flags. Its value must be 0. Else, R10 is ignored. | | |
| R11 | RTC | If the requested version in RAX is 1 or higher, R11 provides a real-time clock (RTC) value – the number of seconds passes since 1/1/1970. RTC value must be lower than 2 ⁶² . A value of 0 indicates that no RTC is provided. If FEATURE_ENABLES0.TDX_CONNECT (R9 bit 6) is 1, R11 must not be 0. Else, R11 is ignored. | | |

5.4.64.2. Output Operands

Table 5.258: TDH.SYS.CONFIG Output Operands Definition

| Operand | Name | Description |
|---------|--------|--|
| RAX | Status | SEAMCALL instruction return code – see 5.4.1 |
| Other | | Unmodified |

5.4.64.3. Leaf Function Description

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

5.4.64.3.1. Overview

10 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.

5.4.64.3.2. Enumeration

Support of dynamic PAMT is enumerated by TDX_FEATURES0.DYNAMIC_PAMT (bit 36), readable by the host VMM using TDH.SYS.RD*.

Support of TDX Connect is enumerated by TDX_FEATURES0.TDX_CONNECT (bit 6).

15 TDH.SYS.CONFIG version 1 is supported if the value of any of the following TDX_FEATURES0 bits is 1, indicating that their respective features are supported:

- TDX_CONNECT (bit 6)
- NRX (bit 39)
- NON_BLOCKING_EXPORT (bit 41)
- 20 • UPDATE_COMPATIBILITY (bit 47)

5.4.64.3.3. Private HKID Space Impact

To be configured for dynamic PAMT (if supported by the TDX Module), the HKID address space partitioning may be required to be configured such that the number of bits allocated to memory encryption keys is not lower than a certain minimum. This is enumerated by `MIN_DYNAMIC_PAMT_NUM_HKID_BITS`, readable by `TDH.SYS.RD*`.

5 5.4.64.4. Operands Information

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

Table 5.259: TDH.SYS.CONFIG Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|--|---------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | TDMR Info Pointers | Array of HPA | R | Shared | 512B | None | N/A | N/A |
| Explicit | N/A | HPA | TDMR Info | TDMR_INFO | R | Shared | 512B | None | N/A | N/A |
| Implicit | N/A | N/A | All Intel TDX Module internal variables | N/A | RW | Hidden | N/A | Exclusive | N/A | N/A |

10 5.4.64.5. Completion Status Codes

Table 5.260: 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 | |
| TDX_OPERAND_INVALID | |
| TDX_PAMT_OUTSIDE_CMRS | |
| TDX_PAMT_OVERLAP | |
| TDX_SUCCESS | TDH.SYS.CONFIG is successful. |
| TDX_SYS_BUSY | The operation was invoked when another TDX Module operation was in progress. The operation may be retried. |
| TDX_SYS_CONFIG_NOT_PENDING | |
| TDX_SYS_SHUTDOWN | |
| TDX_TDMR_ALREADY_INITIALIZED | |
| TDX_TDMR_OUTSIDE_CMRS | |

5.4.65. 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.

5 5.4.65.1. Input Operands

Table 5.261: TDH.SYS.INFO Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 32 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | The physical address (including HKID bits) of a buffer where the output TDSYSINFO_STRUCT will be written | | |
| RDX | The number of bytes in the above buffer | | |
| R8 | The physical address (including HKID bits) of a buffer where an array of CMR_INFO will be written | | |
| R9 | The number of CMR_INFO entries in the above buffer | | |

5.4.65.2. Output Operands

Table 5.262: TDH.SYS.INFO Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| RDX | The actual number of bytes written to the above buffer In case of an error, as indicated by RAX, RDX returns 0. |
| R9 | The number of CMR_INFO entries actually written to the above buffer In case of an error, as indicated by RAX, R9 returns 0. |
| Other | Unmodified |

10

5.4.65.3. 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.

15 5.4.65.4. Operands Information

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

Table 5.263: TDH.SYS.INFO Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|--|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | TDX system information structure | TDSYSINFO_STRUCT | RW | Shared | 1024B | None | N/A | N/A |
| Explicit | R8 | HPA | CMR table | CMR_INFO_ARRAY | RW | Shared | 512B | None | N/A | N/A |

5.4.65.5. Completion Status Codes**Table 5.264: TDH.SYS.INFO Completion Status Codes (Returned in RAX) Definition**

| Completion Status Code | Description |
|------------------------|-----------------------------|
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDH.SYS.INFO is successful. |
| TDX_SYS_SHUTDOWN | |
| TDX_SYSINITLP_NOT_DONE | |

5

5.4.66. TDH.SYS.INIT Leaf

Globally initialize the Intel TDX Module.

5.4.66.1. Input Operands

Table 5.265: TDH.SYS.INIT Input Operands Definition

| Operand | Description | | |
|---------|---|---------------------|---|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 33 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version May be 0 or 1 (see enumeration details below) |
| | 63:24 | Reserved | Must be 0 |
| RCX | Reserved | | |
| | Bits | Name | Description |
| | 63:0 | RESERVED | Reserved: must be 0 |
| R8 | Fatal error diagnostics configuration If the version number provided in RAX is 0, this field is ignored. | | |
| | Bits | Name | Description |
| | 5:0 | RESERVED | Must be 0 |
| | 51:6 | FATAL_INFO_HPA | Bits 51:6 of the shared host physical address (including HKID) of where FATAL_INFO will be written by the TDX Module in case a TDX Module fatal error is detected. The host VMM should clear the FATAL_INFO to all-0 before calling TDH.SYS.INIT. A value of all-1 indicates FATAL_INFO should not be written. |
| | 53:52 | RESERVED | Must be 0 |
| | 55:54 | NOTIFICATION_INTR | Mode of notification interrupt to be broadcast by the TDX Module to all logical processors in case a TDX Module fatal error is detected. 0: No notification interrupt 1: Interrupt 2: NMI 3: SMI |
| | 63:56 | NOTIFICATION_VECTOR | Vector of the above notification interrupt, if the selected mode is 1 (interrupt). Value must be in the range 16:255. If other modes are selected, this field is ignored. |

5.4.66.2. Output Operands

Table 5.266: TDH.SYS.INIT Output Operands Definition

| Operand | Description | | |
|---------|--|-----------|---|
| RAX | SEAMCALL instruction return code – see 5.4.1 | | |
| RCX | Extended error information part 1 If RAX returns TDH_INCORRECT_CPUID_VALUE, RCX returns the applicable CPUID information as shown below. In all other cases, RCX returns 0. | | |
| | Bits | Name | Description |
| | 31:0 | LEAF | CPUID leaf number |
| | 63:32 | SUBLEAF | CPUID sub-leaf number: if sub-leaf is not applicable, value is -1 (0xFFFFFFFF). |
| RDX | Extended error information part 2 If RAX returns TDH_INCORRECT_CPUID_VALUE, RDX returns the value masks as shown below. A bit value of 1 indicates a bit position that was checked against the required value. In all other cases, RDX returns 0. | | |
| | Bits | Name | Description |
| | 31:0 | MASK_EAX | Mask of the value returned by CPUID in EAX |
| | 63:32 | MASK_EBX | Mask of the value returned by CPUID in EBX |
| R8 | Extended error information part 3 If RAX returns TDH_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 |
| R9 | Extended error information part 4 If RAX returns TDH_INCORRECT_CPUID_VALUE, R9 returns the expected values as shown below. In all other cases, R9 returns 0. | | |
| | Bits | Name | Description |
| | 31:0 | VALUE_EAX | Value expected to be returned by CPUID in EAX |
| | 63:32 | VALUE_EBX | Value expected to be returned by CPUID in EBX |
| R10 | Extended error information part 5 If RAX returns TDH_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 | | |

5.4.66.3. Leaf Function Description

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

5 **5.4.66.3.1. Overview**

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.

5.4.66.3.2. Enumeration

10 Support of TDH.SYS.INIT version higher than 0 is enumerated by TDX_FEATURES0.FATAL_DIAGNOSTICS (bit 37), readable by the host VMM using TDH.SYS.RD*.

5.4.66.3.3. 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)
- 15 • 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_CTL.ENERGY_FILTERING_ENABLE (bit 0) must be set to 1.

5.4.66.3.4. Leaf Function Latency

20 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.

5.4.66.4. Operands Information

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

Table 5.267: TDH.SYS.INIT Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------------------|-------------|---|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | R8 ³⁹ | HPA | FATAL_INFO_HPA | FATAL_INFO | RW | Shared | 64 bytes | N/A | N/A | N/A |
| Implicit | N/A | N/A | All Intel TDX Module internal variables | N/A | RW | Hidden | N/A | Exclusive | N/A | N/A |

25

5.4.66.5. Completion Status Codes

Table 5.268: TDH.SYS.INIT Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|---------------------------------------|-------------|
| TDX_LIMIT_CPUID_MAXVAL_SET | |
| TDX_CPUID_LEAF_1F_FORMAT_UNRECOGNIZED | |

³⁹ Applicable only if the requested version number (in RAX) is higher than 0.

| Completion Status Code | Description |
|---------------------------------|---|
| TDX_CPUID_LEAF_1F_NOT_SUPPORTED | |
| TDX_CPUID_LEAF_0D_INCONSISTENT | |
| TDX_INCORRECT_CPUID_VALUE | Additional information is provided in RCX – R10 |
| TDX_INCORRECT_MSR_VALUE | |
| TDX_INVALID_WBINVD_SCOPE | |
| TDX_RND_NO_ENTROPY | Random number generation (e.g., RDRAND or RDSEED) failed because the hardware random number generator did not have enough entropy. The host VMM should retry the operation. |
| TDX_SMRR_LOCK_NOT_SUPPORTED | |
| TDX_SMRR_NOT_LOCKED | |
| TDX_SMRR_NOT_SUPPORTED | |
| TDX_SMRR_OVERLAPS_CMR | |
| TDX_SUCCESS | TDH.SYS.INIT is successful. |
| TDX_SYS_BUSY | The operation was invoked when another TDX Module operation was in progress. The operation may be retried. |
| TDX_SYS_SHUTDOWN | |
| TDX_SYS_INIT_NOT_PENDING | |

5.4.67. TDH.SYS.KEY.CONFIG Leaf

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

5.4.67.1. Input Operands

Table 5.269: TDH.SYS.KEY.CONFIG Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 31 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |

5

5.4.67.2. Output Operands

Table 5.270: TDH.SYS.KEY.CONFIG Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.67.3. Leaf Function Description

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

5.4.67.3.1. Overview

TDH.SYS.KEY.CONFIG performs package-scope Intel TDX global private key configuration.

5.4.67.3.2. Failure Conditions

TDH.SYS.KEY.CONFIG may fail due to the following conditions:

- A conflict with a PCONFIG instruction running on the same package. In this case, a TDX_OPERAND_BUSY status is returned.
- Failure to generate a random key, typically caused by an entropy error of the CPU's random number generator, and may be impacted by RDSEED, RDRAND or PCONFIG executing on other LPs. In this case, a TDX_KEY_GENERATION_FAILED is returned.

20

5.4.67.3.3. 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.

5.4.67.4. Operands Information

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

25

Table 5.271: TDH.SYS.KEY.CONFIG Operands Information

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|---|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Implicit | N/A | N/A | All Intel TDX Module internal variables | N/A | RW | Hidden | N/A | Exclusive | N/A | N/A |

5.4.67.5. Completion Status Codes

Table 5.272: TDH.SYS.KEY.CONFIG Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|--------------------------------|---|
| TDX_KEY_CONFIGURED | |
| TDX_KEY_GENERATION_FAILED | Failed to generate a random key. This is typically caused by an entropy error of the CPU's random number generator, and may be impacted by RDSEED, RDRAND or PCONFIG executing on other LPs. The operation should be retried. |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. Specifically, key configuration may fail due to a concurrently running PCONFIG instruction. |
| TDX_SUCCESS | TDH.SYS.KEY.CONFIG is successful. |
| TDX_SYS_BUSY | The operation was invoked when another TDX Module operation was in progress. The operation may be retried. |
| TDX_SYS_KEY_CONFIG_NOT_PENDING | |
| TDX_SYS_SHUTDOWN | |

5

5.4.68. TDH.SYS.LP.INIT Leaf

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

5.4.68.1. Input Operands**Table 5.273: TDH.SYS.LP.INIT Input Operands Definition**

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 35 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |

5

5.4.68.2. Output Operands**Table 5.274: TDH.SYS.LP.INIT Output Operands Definition**

| Operand | Description | | |
|---------|---|----------|---|
| RAX | SEAMCALL instruction return code – see 5.4.1 | | |
| RCX | Extended error information part 1 If RAX returns TDX_INCONSISTENT_CPUID_FIELD, RCX returns the applicable CPUID information as shown below. In all other cases, RCX returns 0. | | |
| | Bits | Name | Description |
| | 31:0 | LEAF | CPUID leaf number |
| | 63:32 | SUBLEAF | CPUID sub-leaf number: if sub-leaf is not applicable, value is -1 (0xFFFFFFFF). |
| RDX | Extended error information part 2 If RAX returns TDX_INCONSISTENT_CPUID_FIELD, RDX returns the value masks as shown below. A bit value of 1 indicates a bit position that was checked against the same CPUID leaf value checked during TDH.SYS.INIT. In all other cases, RDX returns 0. | | |
| | Bits | Name | Description |
| | 31:0 | MASK_EAX | Mask of the value returned by CPUID in EAX |
| | 63:32 | MASK_EBX | Mask of the value returned by CPUID in EBX |
| R8 | Extended error information part 3 If RAX returns TDX_INCONSISTENT_CPUID_FIELD, R8 returns the value masks as shown below. A bit value of 1 indicates a bit position that was checked against the same CPUID leaf value checked during TDH.SYS.INIT. In all other cases, R8 returns 0. | | |
| | Bits | Name | Description |

| Operand | 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 | | |

5.4.68.3. Leaf Function Description

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

5 **5.4.68.3.1. Overview**

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.

5.4.68.3.2. Special Environment Requirements

10 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_CTL.ENERGY_FILTERING_ENABLE (bit 0) must be set to 1.

15 **5.4.68.3.3. 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.

5.4.68.4. Operands Information

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

Table 5.275: TDH.SYS.LP.INIT Operands Information

| Explicit/Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-------------------|------|----------|---|---------------|--------|------------------|-------------|--------------------------|--------------|--------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Implicit | N/A | N/A | All Intel TDX Module internal variables | N/A | RW | Hidden | N/A | Shared | N/A | N/A |

5.4.68.5. Completion Status Codes

Table 5.276: TDH.SYS.LP.INIT Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------------|--|
| TDX_INCONSISTENT_CPUID_FIELD | Additional information is provided in RCX – R8 |
| TDX_INCONSISTENT_MSR | |
| TDX_INCORRECT_MSR_VALUE | |
| TDX_INVALID_PKG_ID | |

| Completion Status Code | Description |
|-----------------------------|---|
| TDX_RND_NO_ENTROPY | Random number generation (e.g., RDRAND or RDSEED) failed because the hardware random number generator did not have enough entropy. The host VMM should retry the operation. |
| TDX_SUCCESS | TDH.SYS.LP.INIT is successful. |
| TDX_SYS_BUSY | The operation was invoked when another TDX Module operation was in progress. The operation may be retried. |
| TDX_SYS_LP_INIT_DONE | |
| TDX_SYS_LP_INIT_NOT_PENDING | |
| TDX_SYS_SHUTDOWN | |

5.4.69. TDH.SYS.LP.SHUTDOWN Leaf (Deprecated)

This function is deprecated; it is provided for backward compatibility.

5.4.69.1. Input Operands**Table 5.277: TDH.SYS.LP.SHUTDOWN Input Operands Definition**

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 44 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |

5

5.4.69.2. Output Operands**Table 5.278: TDH.SYS.LP.SHUTDOWN Output Operands Definition**

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.69.3. Leaf Function Description

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

TDH.SYS.LP.SHUTDOWN does nothing.

5.4.69.4. Completion Status Codes**Table 5.279: TDH.SYS.LP.SHUTDOWN Completion Status Codes (Returned in RAX) Definition**

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

15

5.4.70. TDH.SYS.RD Leaf

Read a TDX Module global-scope metadata field.

5.4.70.1. Input Operands

Table 5.280: TDH.SYS.RD Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 34 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RDX | <p>Field identifier – see 3.10</p> <p>The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0.</p> <p>WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored.</p> <p>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.</p> | | |

5

5.4.70.2. Output Operands

Table 5.281: TDH.SYS.RD Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| RDX | If the input field identifier was -1, RDX returns the first readable field identifier. Else, in case of an error, RDX returns -1. On success, RDX returns the next readable field identifier. A value of -1 indicates no next field identifier is available. |
| R8 | Contents of the field In case of no success, as indicated by RAX, R8 returns 0. |
| Other | Unmodified |

5.4.70.3. Leaf Function Description

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

5.4.70.3.1. Overview

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 -1 as an input, until RDX returns -1. A status code of TDX_METADATA_FIELD_SKIP indicates that the returned value is not applicable. Alternatively, the host VMM can use TDH.SYS.RDALL.

15

5.4.70.3.2. Enumeration

Availability of TDH.SYS.RD is enumerated by TDSYSINFO_STRUCT.SYS_RD, returned by TDH.SYS.INFO (see 3.3.6). It is also enumerated by TDX_FEATURES0.ENHANCED_METADATA (bit 3), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.SYS.RD returns a TDX_OPERAND_INVALID(RAX) status.

5 5.4.70.4. Operands Information

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

Table 5.282: TDH.SYS.RD Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|--|------|-------------|----------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| There are no relevant memory operands. | | | | | | | | | | |

10 5.4.70.5. Completion Status Codes

Table 5.283: TDH.SYS.RD Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|--|---|
| TDX_METADATA_FIELD_ID_INCORRECT | |
| TDX_METADATA_FIELD_NOT_READABLE | |
| TDX_METADATA_FIRST_FIELD_ID_IN_CONTEXT | Indicates that the first field ID in context is returned |
| TDX_METADATA_FIELD_SKIP | Indicates that the field value being read is not applicable and needs to be skipped. If called in a loop, use RDX as the identifier of the next field to be read, if any. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDH.SYS.RD is successful. |
| TDX_SYS_SHUTDOWN | |
| TDX_SYSINITLP_NOT_DONE | |

5.4.71. TDH.SYS.RDALL Leaf

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

5.4.71.1. Input Operands

Table 5.284: TDH.SYS.RDALL Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 37 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| 63:24 | Reserved | Must be 0 | |
| RDX | The physical address (including HKID bits) of a 4KB page where a metadata list will be returned In case of error, some field value entries might not contain valid data. | | |
| R8 | Initial field identifier – see 3.10 If R8’s value is -1, then TDG.SYS.RDALL will start from the first global-scope metadata field identifier. Else, LAST_ELEMENT_IN_FIELD, LAST_FIELD_IN_SEQUENCE, WRITE_MASK_VALID and CONTEXT_CODE fields are ignored. The FIELD_CODE must be the code of the first element of a metadata field. | | |

5

5.4.71.2. Output Operands

Table 5.285: TDH.SYS.RDALL Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| R8 | Next field identifier. A value of -1 means all applicable field identifiers have been returned in the metadata list. In case of an error, as indicated by RAX, R8 returns -1. |
| Other | Unmodified |

5.4.71.3. Leaf Function Description

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

5.4.71.3.1. Overview

TDH.SYS.RDALL reads all host-readable TDX Module global-scope metadata fields into a metadata list in the provided page.

If one or more applicable fields do not fit in the provided list buffer, the function can be invoked in a loop, each invocation providing an initial field identifier returned as the next field identifier of the previous invocation, as shown in the following example:

1. NEXT_FIELD_ID = -1

- 2. Repeat:
 - 2.1. Set LIST_BUFFER to the next 4K buffer
 - 2.2. Invoke TDH.SYS.RDALL(RDX = LIST_BUFFER, RDX = NEXT_FIELD_ID)
 - 2.3. STATUS = RAX, NEXT_FIELD_ID = R8
- 5 Until ((STATUS is a non-recoverable error) or (NEXT_FIELD_ID is -1))

The function never returns an empty list if there’s no error.

5.4.71.3.2. Enumeration

Availability of TDH.SYS.RDALL is enumerated by TDSYSINFO_STRUCT.SYS_RD, returned by TDH.SYS.INFO (see 3.3.6). It is also enumerated by TDX_FEATURES0.ENHANCED_METADATA (bit 3), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.SYS.RDALL returns a TDX_OPERAND_INVALID(RAX) status.

5.4.71.4. Operands Information

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

Table 5.286: TDH.SYS.RDALL Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|---------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RDX | HPA | Metadata list | MD_LIST | RW | Shared | 4KB | None | None | None |

15

5.4.71.5. Completion Status Codes

Table 5.287: TDH.SYS.RDALL Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.SYS.RDALL is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |

5.4.72. TDH.SYS.S4_END Leaf

Help prevent replay attacks, by preventing future S4 restoration using the current global S4 session’s anti-replay nonce, which was generated during the S4 hibernation session.

5.4.72.1. Input Operands

5 **Table 5.288: TDH.SYS.S4_END Input Operands Definition**

| Operand | Name | Description | | |
|---------|------------------|---|----------------|--|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 54 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | | 63:24 | Reserved | Must be 0 |

5.4.72.2. Output Operands

Table 5.289: TDH.SYS.S4_END Output Operands Definition

| Operand | Name | Description |
|---------|--------|--|
| RAX | Status | SEAMCALL instruction return code – see 5.4.1 |
| Other | | Unmodified |

10 **5.4.72.3. Leaf Function Description**

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

5.4.72.3.1. Overview

15 TDH.SYS.S4_END generates a new anti-replay nonce. This helps prevent replay attack which may attempt to start a new S4 resumption session and resume the TDs using the anti-replay nonce generated during the S4 hibernation session.

5.4.72.3.2. Enumeration

Availability of TDH.SYS.S4_END is enumerated by TDX_FEATURES0.S4 (bit 13), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.SYS.S4_END returns a TDX_OPERAND_INVALID(RAX) status.

5.4.72.4. Operands Information

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

Table 5.290: TDH.SYS.S4_END Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|-------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| PL.S4_STATE | N/A | N/A | PL.S4_STATE | N/A | RW | Hidden | N/A | Exclusive | None | None |

5.4.72.5. Completion Status Codes

Table 5.291: TDH.SYS.S4_END Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------|---|
| TDX_SUCCESS | TDH.SYS.S4_END is successful. |
| TDX_OPERAND_BUSY | |
| TDX_SYS_NOT_READY | TDH.SYS.S4_END was called when TDX Module's lifecycle state is not SYS_READY. |

5

5.4.73. TDH.SYS.SHUTDOWN Leaf

Initiate Intel TDX Module shutdown and generate handoff data for the next Intel TDX Module.

5.4.73.1. Input Operands

Table 5.292: TDH.SYS.SHUTDOWN Input Operands Definition

| Operand | Name | Description | | |
|---------|------------------|---|------------------------|---|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 52 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | | 63:24 | Reserved | Must be 0 |
| RCX | HANDOFF | Handoff parameters | | |
| | | Bits | Field | Description |
| | | 15:0 | REQ_HV | Requested handoff version |
| | | 16 | AVOID_COMPAT_SENSITIVE | Flags that TDH.SYS.SHUTDOWN should fail if any TD is in a phase sensitive to update compatibility See below for enumeration details. |
| | | 63:7 | Reserved | Must be 0 |

5

5.4.73.2. Output Operands

Table 5.293: TDH.SYS.SHUTDOWN Output Operands Definition

| Operand | Name | Description |
|---------|--------|--|
| RAX | Status | SEAMCALL instruction return code – see 5.4.1 |
| Other | | Unmodified |

5.4.73.3. Leaf Function Description

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

5.4.73.3.1. Overview

TDH.SYS.SHUTDOWN initiates Intel TDX Module shutdown and generates handoff data for the next Intel TDX Module. The host VMM should call TDH.SYS.SHUTDOWN when no concurrent LP is in SEAM mode, i.e., all SEAMCALL interface functions except the current one have returned to the host VMM.

15

Following TDH.SYS.SHUTDOWN, no further TDX Module interface functions can be called until a new module is installed.

5.4.73.3.2. Enumeration

Availability of TDH.SYS.SHUTDOWN is enumerated by TDX_FEATURES0.TD_PRESERVING (bit 1), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.SYS.SHUTDOWN returns a TDX_OPERAND_INVALID(RAX) status.

20

Availability of the AVOID_COMPAT_SENSITIVE input flag is enumerated by TDX_FEATURES0.UPDATE_COMPATIBILITY.

5.4.73.3.3. Handoff Version Considerations

The REQ_HV input parameter specifies the required version of the handoff data prepared by TDH.SYS.UPDATE, which will be consumed by the TDX Module that will be installed. REQ_HV must comply with the following conditions:

- 5 • If the current module supports downgrade on update (NO_DOWNGRADE is 0), then REQ_HV must be no lower than MIN_UPDATE_HV, the minimal update handoff version supported by the TDX Module, and no higher than MODULE_HV, the current handoff version of the TDX Module.
- Else, REQ_HV must be the same as MODULE_HV, the current handoff version of the TDX Module.

NO_DOWNGRADE, MIN_UPDATE_HV and MODULE_HV can be read by the host VMM using TDH.SYS.RD*.

10 For details, see the [TDX Module Base Spec] section titled “TD-Preserving TDX Module Update”.

5.4.73.3.4. Update Compatibility Considerations

TD-preserving TDX Module update may cause failure of some TD management operations in some cases, if it happens during phases of the TD lifecycle that are sensitive to update compatibility. For details, see the [TDX Module Base Spec] section titled “Compatibility Aspects of TD-Preserving Update”.

15 If supported by the TDX Module, the host VMM can set the AVOID_COMPAT_SENSITIVE flag to request the TDX Module to fail TDH.SYS.UPDATE if any of the TDs are currently in a state that is impacted by the update-sensitive cases. The compatibility checks done by TDH.SYS.UPDATE do not include the following cases:

- If any TD was initialized by an older TDX Module that did enumerate TDX_FEATURES0.UPDATE_COMPATIBILITY as 1, TDH.SYS.SHUTDOWN does not check for a TD build in progress condition for that TD.
- 20 • If any TD migration session is in progress, it was initialized by an older TDX Module that did enumerate TDX_FEATURES0.UPDATE_COMPATIBILITY as 1, TDH.SYS.SHUTDOWN does not check for an interrupted TD migration function condition for that TD.

5.4.73.4. Operands Information

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

Table 5.294: TDH.SYS.SHUTDOWN Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|---|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Implicit | N/A | N/A | All Intel TDX Module internal variables | N/A | RW | Hidden | N/A | Exclusive | N/A | N/A |

5.4.73.5. Completion Status Codes

Table 5.295: TDH.SYS.SHUTDOWN Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------|--|
| TDX_OPERAND_INVALID | The requested handoff version is invalid. |
| TDX_SUCCESS | TDH.SYS.SHUTDOWN is successful. |
| TDX_SYS_BUSY | The operation was invoked when another TDX Module operation was in progress. The operation may be retried. |

| Completion Status Code | Description |
|------------------------------------|---|
| TDX_SYS_NOT_READY | TDH.SYS.SHUTDOWN was called when TDX Module's lifecycle state is not SYS_READY. |
| TDX_UPDATE_COMPATIBILITY_SENSITIVE | TDH.SYS.SHUTDOWN detected that there is one or more ongoing flows which may not be compatible with an updated TDX Module. |

5.4.74. TDH.SYS.TDMR.INIT Leaf

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

5.4.74.1. Input Operands**Table 5.296: TDH.SYS.TDMR.INIT Input Operands Definition**

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 36 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | The physical base address of a TDMR (HKID bits must be 0) | | |

5

5.4.74.2. Output Operands**Table 5.297: TDH.SYS.TDMR.INIT Output Operands Definition**

| Operand | Description |
|---------|---|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| RDX | On successful completion, RDX returns the TDMR next-to-initialize address. This is the physical address of the last byte that has been initialized so far, rounded down to 1GB. In all other cases, RDX returns 0. |
| Other | Unmodified |

5.4.74.3. Leaf Function Description

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

5.4.74.3.1. Overview

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.

5.4.74.3.2. Enumeration

Support of dynamic PAMT is enumerated by TDX_FEATURES0.DYNAMIC_PAMT (bit 36), readable by the host VMM using TDH.SYS.RD*.

5.4.74.4. Operands Information

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

Table 5.298: TDH.SYS.TDMR.INIT Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|--|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | TDMR | Blob | None | None | 1GB | Exclusive | N/A | N/A |
| Implicit | N/A | HPA | PAMT region associated with TDMR | Blob | RW | Hidden | N/A | Exclusive | N/A | N/A |

5.4.74.5. Completion Status Codes

Table 5.299: TDH.SYS.TDMR.INIT Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------------|--|
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDH.SYS.TDMR.INIT is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TDMR_ALREADY_INITIALIZED | |

5

5.4.75. TDH.SYS.UPDATE Leaf

Unreleased Feature: Some of the text in this section is related to Non-Blocking Export, a feature which has not been released yet at the time of writing of this document. Details related to that feature serve as a preview and are subject to change.

- 5 Update the TDX Module and restore its internal variable from the handoff data prepared by the previous Intel TDX Module using TDH.SYS.SHUTDOWN.

5.4.75.1. Input Operands

Table 5.300: TDH.SYS.UPDATE Input Operands Definition

| Operand | Name | Description | | |
|---------|------------------|--|----------------------|---|
| RAX | Leaf and Version | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 53 |
| | | 23:16 | Version Number | Selects the SEAMCALL interface function version Versions 0 and 1 are supported. See enumeration details below. |
| | | 63:24 | Reserved | Must be 0 |
| R9 | FEATURES_ENABLE0 | If the requested version in RAX is 1 or higher, R9 specifies TDX Module feature enabling flags, formatted similarly to TDX_FEATURES0, readable by TDH.SYS.RD*. A bit may be set to 1 if the corresponding TDX_FEATURES0 bit is 1. There are limitations on how features can be enabled vs. the previous TDX Modules that ran on the platform since the initial TDX Module installation, as detailed below. Else, R9 is ignored. | | |
| | | Bits | Name | Description |
| | | 6 | TDX_CONNECT | Enables TDX Connect. TDX Connect cannot be disabled if it was enabled for the pre-update TDX Module. |
| | | 41 | NON_BLOCKING_EXPORT | Controls TD export mode: 0: Write-blocking based export. This value is valid only if the pre-update TDX module was configured for write-blocking based export. 1: Non-blocking export. This value is valid only if write-blocking based export session has never happened during the lifecycle of any of the previous TDX modules from which a TD-preserving update was done. |
| | | 47 | UPDATE_COMPATIBILITY | Enables per-TD compatibility checks with internal metadata created by the previous TDX Module. |
| | | Bits whose value in TDX_FEATURES0 is 0 | | Must be 0 |
| | | Other bits | | Ignored |
| R10 | FEATURES_ENABLE1 | If the requested version in RAX is 1 or higher, R10 is reserved for additional TDX Module feature enabling flags. Its value must be 0. Else, R10 is ignored. | | |

| Operand | Name | Description |
|---------|------|--|
| R11 | RTC | If the requested version in RAX is 1 or higher, R11 provides a real-time clock (RTC) value – the number of seconds passes since 1/1/1970. RTC value must be lower than 2^{62} . A value of 0 indicates that no RTC is provided. If FEATURE_ENABLES0.TDX_CONNECT (R9 bit 6) is 1, R11 must not be 0. Else, R11 is ignored. |

5.4.75.2. Output Operands

Table 5.301: TDH.SYS.UPDATE Output Operands Definition

| Operand | Name | Description |
|---------|--------|--|
| RAX | Status | SEAMCALL instruction return code – see 5.4.1 |
| Other | | Unmodified |

5 5.4.75.3. Leaf Function Description

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

5.4.75.3.1. Overview

TDH.SYS.UPDATE reads the handoff data prepared by the previous Intel TDX Module using TDH.SYS.SHUTDOWN.

10 5.4.75.3.2. Enumeration

Availability of TDH.SYS.UPDATE is enumerated by TDX_FEATURES0.TD_PRESERVING (bit 1), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.SYS.UPDATE returns a TDX_OPERAND_INVALID(RAX) status.

Update compatibility check support is enumerated by TDX_FEATURES0.UPDATE_COMPATIBILITY (bit 47).

15 TDH.SYS.UPDATE version 1 is supported if the value of any of the following TDX_FEATURES0 bits is 1, indicating that their respective features are supported:

- TDX_CONNECT (bit 6)
- NRX (bit 39)
- NON_BLOCKING_EXPORT (bit 41)
- UPDATE_COMPATIBILITY (bit 47)

20 5.4.75.3.3. Update Compatibility Checks

If requested by the host VMM using the input flag FEATURES_ENABLES0.UPDATE_COMPATIBILITY (bit 47), TDH.SYS.UPDATE checks compatibility of intermediate TD state saved by the previous TDX Module. See the discussion in the [Base FAS] section titled “Compatibility Aspects of TD-Preserving Update”.

5.4.75.3.4. Update Failure Scenarios

25 TDX Module update may fail in the following cases:

- No valid handoff data
- The old module’s handoff data’s version is too old for the current TDX Module
- The old module’s handoff data’s version is newer than the current TDX Module’s handoff data version
- Update compatibility checks, enabled by the input flag FEATURES_ENABLES0.UPDATE_COMPATIBILITY (bit 47), failed

30 On such failures the host VMM is expected to do one of the following:

- Request the Persistent SEAMLDR to update to another TDX Module (UPDATE scenario). If that update is successful, existing TDs are preserved.
- Keep the current TDX Module and continue with the non-update sequence (TDH.SYS.CONFIG, TDH.SYS.KEY.CONFIG etc.). In this case all existing TDs are lost.

5.4.75.3.5. TD Export Mode Configuration

As a rule, TD export mode (write-blocking export or non-blocking export) cannot be modified on TDX module update. However, there is an exception: Since write-blocking export is the default configuration, if write-blocking export was not used before, the updated module can be configured to non-blocking export.

5 5.4.75.4. Operands Information

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

Table 5.302: TDH.SYS.UPDATE Operands Information

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|---|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Implicit | N/A | N/A | All Intel TDX Module internal variables | N/A | RW | Hidden | N/A | Exclusive | N/A | N/A |

10 5.4.75.5. Completion Status Codes

Table 5.303: TDH.SYS.UPDATE Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|---------------------------|---|
| TDX_CONNECT_INVALID_STATE | Applicable only for TDX Module and CPUs that support TDX Connect |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDH.SYS.UPDATE is successful. |
| TDX_SYS_BUSY | The operation was invoked when another TDX Module operation was in progress. The operation may be retried. |
| TDX_SYS_STATE_INCORRECT | TDH.SYS.SHUTDOWN was called when the TDX Module's lifecycle state is not SYSINIT_DONE or some LPs have not yet been initialized by TDH.SYS.LP.INIT. |
| TDX_SYS_INVALID_HANDOFF | The handoff data in SEAM range is invalid. |

5.4.76. TDH.VP.ADDCX Leaf

Add a physical page to a TDVPS.

5.4.76.1. Input Operands

Table 5.304: TDH.VP.ADDCX Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 4 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | The physical address of a page where the TDCX page will be added (HKID bits must be 0) | | |
| RDX | The physical address of a TDVPR page (HKID bits must be 0) | | |

5

5.4.76.2. Output Operands

Table 5.305: TDH.VP.ADDCX Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.76.3. Leaf Function Description

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

5.4.76.3.1. Overview

TDH.VP.ADDCX adds a physical page to a TDVPS, as a child of a given TDVPR.

5.4.76.3.2. Dynamic PAMT

15 If the TDX Module is configured for dynamic PAMT, the PAMT hierarchy can be built on demand. A TDX_MISSING_PAMT_PAGE_PAIR status indicates that a PAMT page pair is missing for the new TDCX page. The host VMM may add it using TDH.PHYMEM.PAMT.ADD and retry the operation.

5.4.76.4. Operands Information

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

Table 5.306: TDH.VP.ADDCX Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------|-------------|----------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ⁴⁰ |
| Explicit | RCX | HPA | TDCX page | Blob | RW | Opaque | 4KB | Exclusive | Shared | Shared | Exclusive |
| Explicit | RDX | HPA | TDVPR page | Blob | RW | Opaque | 4KB | Exclusive | Shared | Shared | None |
| Implicit | N/A | HPA | TDR page | TDR | RW | Opaque | N/A | Shared | None | None | None |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | N/A | N/A | None |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared | N/A | N/A | None |

5.4.76.5. Completion Status Codes

Table 5.307: TDH.VP.ADDCX Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_MISSING_PAMT_PAGE_PAIR | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.VP.ADDCX is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |
| TDX_TDCX_NUM_INCORRECT | |
| TDX_VCPU_STATE_INCORRECT | |

5

⁴⁰ ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

5.4.77. TDH.VP.CREATE Leaf

Create a guest TD VCPU and its TDVPS root page (TDVPR).

5.4.77.1. Input Operands

Table 5.308: TDH.VP.CREATE Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 10 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | The physical address of a page where TDVPR will be added (HKID bits must be 0) | | |
| RDX | The physical address of the owner TDR page (HKID bits must be 0) | | |

5

5.4.77.2. Output Operands

Table 5.309: TDH.VP.CREATE Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.77.3. Leaf Function Description

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

5.4.77.3.1. Overview

TDH.VP.CREATE begins the build sequence of a new guest TD VCPU. It adds a TDVPR page as a child of a TDR page.

5.4.77.3.2. Dynamic PAMT

15 If the TDX Module is configured for dynamic PAMT, the PAMT hierarchy can be built on demand. A TDX_MISSING_PAMT_PAGE_PAIR status indicates that a PAMT page pair is missing for the new TDVPR page. The host VMM may add it using TDH.PHYMEM.PAMT.ADD and retry the operation.

5.4.77.4. Operands Information

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

Table 5.310: TDH.VP.CREATE Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | | |
|-----------------------|------|-------------|----------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|--------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB | ACT 2MB ⁴¹ |
| Explicit | RCX | HPA | TDVPR page | Blob | RW | Opaque | 4KB | Exclusive | Shared | Shared | Exclusive |
| Explicit | RDX | HPA | TDR page | TDR | RW | Opaque | 4KB | Shared | Shared | Shared | None |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | 4KB | Shared(i) | N/A | N/A | None |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared | N/A | N/A | None |

5.4.77.5. Completion Status Codes

Table 5.311: TDH.VP.CREATE Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_MISSING_PAMT_PAGE_PAIR | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.VP.CREATE is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |

5

⁴¹ ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

5.4.78. TDH.VP.ENTER Leaf

Unreleased Feature: Some of the text in this section is related to Enhanced Interrupt Virtualization, a feature which has not been released yet at the time of writing of this document. Details related to that feature serve as a preview and are subject to change.

5 Enter TDX non-root operation.

From the host VMM’s point of view, TDH.VP.ENTER is a complex operation that normally involves TD entry followed by a TD exit. Therefore, input and output operands are specified by multiple tables below.

5.4.78.1. Input Operands

TDH.VP.ENTER input format depends on how the previous TDH.VP.ENTER was terminated. There are two cases:

- 10
- Initial entry or following a previous asynchronous TD exit
 - Following a previous TDCALL(TDG.VP.VMCALL)

5.4.78.1.1. Input Format for Initial Entry or Following a Previous Asynchronous TD Exit

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

15 **Table 5.312: TDH.VP.ENTER Input Operands Format #1 Definition: Initial Entry or Following a Previous Asynchronous TD Exit**

| Operand | Description | | |
|---------|---|--------------------------|---|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 10 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | VCPU handle and flags | | |
| | Bit(s) | Name | Description |
| | 11:0 | RESERVED | Must be 0 |
| | 51:12 | TDVPR_HPA | Bits 51:12 of the physical address of the TD VCPU’s TDVPR page (HKID bits must be 0) |
| | 52 | HOST_RECOVERABILITY_HINT | Applicable only following a previous trap-like asynchronous TD exit , where bit 60 (HOST_RECOVERABILITY_HINT) of the previous TDH.VP.ENTER completion status (returned in RAX) was set to 1. In all other cases this bit must be 0. 0: The host VMM hints that the guest-side function may possibly be retried (e.g., the host may have corrected some conditions). 1: The host VMM hints that the error is probably not recoverable. This bit is reflected to the guest TD in bit 60 of RAX. |
| | 53 | RESUME_L1 | For partitioned TDs, indicates that the L1 VMM should be resumed. Applicable after TD exits from an L2 VM. 0: TDH.VP.ENTER resumes the L2 VM it last exited from. 1: TDH.VP.ENTER resumes L1 VMM, even if the previous TD exit was from an L2 VM. |

| Operand | Description | | |
|---------|-------------|----------|---|
| | 57:54 | ON | Indicates an outstanding posted interrupt notification: Bit 54: Outstanding notification for L1 Bit 55: Outstanding notification for L2 VM #1 Bit 56: Outstanding notification for L2 VM #2 Bit 57: Outstanding notification for L2 VM #3 See enumeration details below. |
| | 63:58 | RESERVED | Must be 0 |

5.4.78.1.2. Input Format following a Previous TDCALL(TDG.VP.VMCALL)

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

5

Table 5.313: TDH.VP.ENTER Input Operands Format #2 Definition: Following a Previous TDCALL(TDG.VP.VMCALL)

| Operand | Description | | |
|-----------------------------------|--|-----------|---|
| RAX | SEAMCALL instruction leaf and version numbers – see 5.4.1 | | |
| RCX | VCPU handle and flags | | |
| | Bit(s) | Name | Description |
| | 11:0 | RESERVED | Must be 0 |
| | 51:12 | TDVPR_HPA | Bits 51:12 of the physical address of the TD VCPU's TDVPR page (HKID bits must be 0) |
| | 52 | RESERVED | Must be 0 |
| | 53 | RESUME_L1 | For partitioned TDs, indicates that the L1 VMM should be resumed. Applicable after TD exits from an L2 VM. 0: TDH.VP.ENTER resumes the L2 VM it last exited from. 1: TDH.VP.ENTER resumes L1 VMM, even if the previous TD exit was from an L2 VM. |
| | 57:54 | ON | Indicates an outstanding posted interrupt notification: Bit 54: Outstanding notification for L1 Bit 55: Outstanding notification for L2 VM #1 Bit 56: Outstanding notification for L2 VM #2 Bit 57: Outstanding notification for L2 VM #3 See enumeration details below. |
| | 63:58 | RESERVED | Must be 0 |
| RBX, RDX, RBP, RSI, RDI, R8 – R15 | If the corresponding bit of RCX at the previous TD exit (i.e., previous TDH.VP.ENTER termination) was set to 1, the register value is passed as-is to the guest TD – see the description of TDG.VP.VMCALL in 5.5.26 for details. Else, the register value is not used as an input. If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP can't be used to pass values to the guest TD. See the enumeration note below. | | |
| XMM0 – XMM15 | If the corresponding bit of RCX at the previous TD exit (i.e., previous TDH.VP.ENTER termination) was set to 1, the register value is passed as-is to the guest TD – see the description of TDG.VP.VMCALL in 5.5.26 for details. Else, the register value is not used as an input. | | |

5.4.78.2. Output Operands

TDH.VP.ENTER output format depends on how the function was terminated. There are multiple cases:

1. Error (No TD Entry)
2. Asynchronous TD exit following a TD entry (with a VMX architectural exit reason)
3. Asynchronous TD exit following a TD entry (with a non-VMX TD exit status)
4. Asynchronous TD exit following a TD entry (with cross-TD exit details)
5. TD exit due to a TDCALL(TDG.VP.VMCALL) following a TD entry
6. TD exit due to a guest TD request

All the TD exit cases formats share some fields, as described below.

5.4.78.2.1. Output Format #1: Error (No TD Entry)

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

Table 5.314: TDH.VP.ENTER Output Operands Format #1 Definition: On Error (No TD Entry)

| Operand | Description | | | |
|----------------|---|----------------------------------|---|--|
| RAX | Status | SEAMCALL instruction return code | | |
| | | Bit(s) | Name | Description |
| | | 63 | ERROR | Set to 1 |
| | | 47:32 | CLASS and DETAILS_L1 | None of the values detailed in the table below |
| | Other | | See the function completion status definition in 5.4.1. | |
| Other GPRs | Unmodified | | | |
| Extended State | Any extended state that the TD is allowed to use (per TDCS.XFAM) may be cleared to its architectural RESET state. | | | |

5.4.78.2.2. Common Output Format on TD Exits

The following table details the common format of TDH.VP.ENTER output operands when TD entry succeeds, and later a TD exit occurs. The following tables provide information for each specific case.

Table 5.315: TDH.VP.ENTER Common Output Operands Format on TD Exits Following a TD Entry

| Operand | Name | Description | | |
|------------------------------|-------------------------|--|--------------------|---|
| RAX | Status | SEAMCALL instruction return code – see the function completion status definition in 5.4.1. | | |
| RCX | Common Exit Information | Index of the VM from which the TD exit occurred | | |
| | | Bit(s) | Name | Description |
| | | 31:0 | Format Dependent | See specific output formats below |
| | | 33:32 | VM | Index of the VM that was running at the time of TD exit |
| | 63:34 | RESERVED | Reserved, set to 0 | |
| RBX, RDX, RSI, RDI, R8 – R15 | Format Dependent | See specific output formats below | | |

| Operand | Name | Description |
|----------------|------|---|
| RBP | None | If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP is unmodified. See the enumeration note below. Else, RBP may be modified. |
| Extended State | | Any extended state that the TD is allowed to use (per TDCS.XFAM) is cleared to its architectural RESET state. In case of a TDCALL(TDG.VP.VMCALL) following a TD entry, XMM may contain output operands. See below for details. |

5.4.78.2.3. Output Format #2: Asynchronous TD Exits Following a TD Entry (with a VMX Architectural Exit Reason)

The following table details TDH.VP.ENTER output operands when TD entry succeeds, and later an **asynchronous TD exit** occurs due to a **VMX architectural exit reason**.

Table 5.316: TDH.VP.ENTER Output Operands Format #2 Definition: On Asynchronous TD Exits Following a TD Entry (with a VMX Architectural Exit Reason)

| Operand | Name | Description | | |
|---------|------------------|---|---|---|
| RAX | Status | SEAMCALL instruction return code | | |
| | | Bit(s) | Name | Description |
| | | 31:0 | DETAILS_L2: Exit Reason | VMCS exit reason Note: Exit reason TDCALL (77) is a special case, indicating a synchronous TD exit initiated by TDG.VP.VMCALL; see below. |
| | | 47:32 | CLASS and DETAILS_L1 | May have the following values: <ul style="list-style-type: none"> TDX_SUCCESS, indicating a normal TD exit TDX_NON_RECOVERABLE_VCPU, indicating that the VCPU is disabled TDX_NON_RECOVERABLE_TD, indicating that the TD is disabled TDX_NON_RECOVERABLE_TD_NON_ACCESSIBLE, indicating that the TD is disabled, and its private memory can't be accessed TDX_TD_EXIT_ON_L2_VM_EXIT and TDX_TD_EXIT_ON_L2_TO_L1, indicating a debug TD exit on L2 transitions |
| | Other | | See the function completion status definition in 5.4.1. | |
| RCX | Exit Information | Index of the VM from which the TD exit occurred | | |
| | | Bit(s) | Name | Description |
| | | 31:0 | EXIT_QUALIFICATION | VMCS exit qualification bits 31:0 Note: VMCS exit qualification bits 63:32 are always 0. When exit is due to an EPT violation, bits 12:7 are cleared to 0. |
| | | 33:32 | VM | Index of the VM that was running at the time of TD exit |
| | 62:34 | RESERVED | Reserved, set to 0 | |

| Operand | Name | Description | | |
|--------------------------|---|--|-----------------|---|
| | | 63 | IMM_RESUME_HINT | Indicates that the host VMM should resume the TD VCPU as soon as possible to handle a pending event. See below for enumeration and configuration details. |
| RDX | Extended Exit Qualification | Additional non-VMX, TDX-specific information – see 3.7.1 | | |
| R8 | Guest Physical Address | When exit is due to EPT violation or EPT misconfiguration, format is similar to the VMCS guest-physical address, except that bits 11:0 are cleared to 0. In other cases, R8 is cleared to 0. | | |
| R9 | VM-Exit Interruption Information | When exit is due to a vectored event, the 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 | Reserved | Cleared to 0 | | |
| RBP | None | If the TD’s CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP is unmodified. See the enumeration note below. Else, RBP is cleared to 0. | | |
| Extended State | Any extended state that the TD is allowed to use (per TDCS.XFAM) is cleared to its architectural RESET state. | | | |

5.4.78.2.4. Output Format #3: Asynchronous TD Exits Following a TD Entry (with a non-VMX TD Exit Status)

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

5 **Table 5.317: TDH.VP.ENTER Output Operands Format #3 Definition: On Asynchronous TD Exits Following a TD Entry (with a non-VMX TD Exit Status)**

| Operand | Name | Description | | |
|---------|------------------|---|----------------------|--|
| RAX | Status | SEAMCALL instruction return code | | |
| | | Bit(s) | Name | Description |
| | | 47:32 | CLASS and DETAILS_L1 | May have the following values: <ul style="list-style-type: none"> • TDX_HOST_PRIORITY_BUSY_TIMEOUT • TDX_NON_RECOVERABLE_TD_CORRUPTED_MD • TDX_TD_EXIT_BEFORE_L2_ENTRY • TDX_TDCALL_RATE_LIMIT • TDX_IPI_REQUEST • TDX_NV_NOT_UNIQUE |
| | Other | See the function completion status definition in 5.4.1. | | |
| RCX | Exit Information | TD exit information | | |
| | | Bit(s) | Name | Description |
| | | 31:0 | RESERVED | Reserved, set to 0 |

| Operand | Name | Description | | |
|------------------------------|---|---|-----------------|---|
| | | 33:32 | VM | Index of the VM that was running at the time of TD exit |
| | | 62:34 | RESERVED | Reserved, set to 0 |
| | | 63 | IMM_RESUME_HINT | Indicates that the host VMM should resume the TD VCPU as soon as possible to handle a pending event See below for enumeration and configuration details. |
| RDX, RBX, RSI, RDI, R8 – R15 | Reserved | Cleared to 0 Note: In the future, if this format will be used with new status codes, these GPRs may be used to return additional information. | | |
| RBP | None | If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP is unmodified. See the enumeration note below. Else, RBP is cleared to 0. | | |
| Extended State | Any extended state that the TD is allowed to use (per TDCS.XFAM) is cleared to its architectural RESET state. | | | |

5.4.78.2.5. Output Format #4: Asynchronous TD Exits Following a TD Entry (with Cross-TD Exit Details)

The following table details TDH.VP.ENTER output operands when TD entry succeeds, and later an asynchronous TD exit occurs due to a **cross-TD operation**, i.e., the current TD operating on another TD.

5 **Table 5.318: TDH.VP.ENTER Output Operands Format #4 Definition: On Asynchronous TD Exits Following a TD Entry (with Cross-TD Exit Details)**

| Operand | Name | Description | | |
|---------|------------------|----------------------------------|----------------------|---|
| RAX | Status | SEAMCALL instruction return code | | |
| | | Bit(s) | Name | Description |
| | | 47:32 | CLASS and DETAILS_L1 | May have the following values: <ul style="list-style-type: none"> TDX_CROSS_TD_FAULT, indicating a fault-like asynchronous TD exit, with non-VMX cross-TD status. TDX_CROSS_TD_TRAP, indicating a trap-like asynchronous TD exit, with non-VMX cross-TD status. |
| | | Other | | See the function completion status definition in 5.4.1. |
| RCX | Exit Information | TD exit information | | |
| | | Bit(s) | Name | Description |
| | | 32:0 | RESERVED | Reserved, set to 0 |
| | | 33:32 | VM | Index of the VM that was running at the time of TD exit |
| | | 62:34 | RESERVED | Reserved, set to 0 |
| | | 63 | IMM_RESUME_HINT | Indicates that the host VMM should resume the TD VCPU as soon as possible to handle a pending event See below for enumeration and configuration details. |

| Operand | Name | Description |
|-------------------------|---|--|
| RDX | Cross-TD Status | Status code of the error which caused the TD exit, using the same format as TDCALL instruction return code |
| R8 | Target TD | HPA of the TDR page of the TD which was the target of the cross-TD operation |
| RBX, RSI, RDI, R9 – R15 | Reserved | Cleared to 0 |
| RBP | None | If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP is unmodified. See the enumeration note below. Else, RBP is cleared to 0. |
| Extended State | Any extended state that the TD is allowed to use (per TDCS.XFAM) is cleared to its architectural RESET state. | |

5.4.78.2.6. Output Format #5: TD Exit due to TDCALL(TDG.VP.VMCALL) Following a TD Entry

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

5 **Table 5.319: TDH.VP.ENTER Output Operands Format #5 Definition: On TDCALL(TDG.VP.VMCALL) Following a TD Entry**

| Operand | Name | Description | | |
|-----------------------------------|------------------|---|-------------------------|--|
| RAX | Status | SEAMCALL instruction return code | | |
| | | Bit(s) | Name | Description |
| | | 31:0 | DETAILS_L2: Exit Reason | VMCS exit reason, indicating TDCALL (77) |
| | | 47:32 | CLASS and DETAILS_L1 | Indicating TDX_SUCCESS |
| | | Other | | See the function completion status definition in 5.4.1. |
| RCX | Exit Information | TD exit information | | |
| | | Bit(s) | Name | Description |
| | | 31:0 | PARAMS_MASK | Value as passed into TDCALL(TDG.VP.VMCALL) by the guest TD: indicates which part of the guest TD GPR and XMM state is passed as-is to the VMM and back. For details, see the description of TDG.VP.VMCALL in 5.5.26. |
| | | 33:32 | VM | Index of the VM that was running at the time of TD exit |
| | | 62:34 | RESERVED | Reserved, set to 0 |
| | | 63 | IMM_RESUME_HINT | Indicates that the host VMM should resume the TD VCPU as soon as possible to handle a pending event See below for enumeration and configuration details. |
| RBX, RDX, RBP, RDI, RSI, R8 – R15 | GPRs | If the corresponding bit in RCX is set to 1, the register value is passed as-is from the guest TD's input to TDG.VP.VMCALL. Else, the register value cleared to 0. | | |

| Operand | Name | Description |
|---------------------------|------|--|
| | | If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP can't be used to pass values from the guest TD and is not modified from its input value. See the enumeration note below. |
| XMM0 – XMM15 | XMMs | If the corresponding bit in RCX is set to 1, the register value is passed as-is from the guest TD's input to TDG.VP.VMCALL. Else, the register value cleared to 0. |
| Extended State except XMM | | Any extended state, except XMM, that the TD is allowed to use (per TDCS.XFAM) is cleared to its architectural RESET state. |

5.4.78.2.7. Output Format #6: TD Exit due to a Guest TD Request

The following table details TDH.VP.ENTER output operands when TD entry succeeds, and later on the TD requests calls a TDX Module guest-side function that eventually results in a request from the host VMM.

5 **Table 5.320: TDH.VP.ENTER Output Operands Format #6 Definition: On TD Exit due to a Guest TD Request**

| Operand | Name | Description | | |
|------------------------------|------------------|---|----------------------|---|
| RAX | Status | SEAMCALL instruction return code | | |
| | | Bit(s) | Name | Description |
| | | 47:32 | CLASS and DETAILS_L1 | May have the following values: <ul style="list-style-type: none"> TDX_IOTLB_INV_REQUEST⁴² |
| | | Other | | See the function completion status definition in 5.4.1. |
| RCX | Exit Information | TD exit information | | |
| | | Bit(s) | Name | Description |
| | | 31:0 | RESERVED | Reserved, set to 0 |
| | | 33:32 | VM | Index of the VM that was running at the time of TD exit |
| 63:34 | RESERVED | Reserved, set to 0 | | |
| RDX, RBX, RSI, RDI, R8 – R15 | Reserved | Cleared to 0 Note: In the future, if this format will be used with new status codes, these GPRs may be used to return additional information. | | |
| RBP | None | If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP is unmodified. See the enumeration note below. Else, RBP is cleared to 0. | | |
| Extended State | | Any extended state that the TD is allowed to use (per TDCS.XFAM) is cleared to its architectural RESET state. | | |

5.4.78.3. Leaf Function Description

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

⁴² This status code is applicable if the TDX Module supports TDX Connect, as enumerated by TDX_FEATURES0.TDX_CONNECT (bit 6), readable using TDH.SYS.RD*.

5.4.78.3.1. Overview

TDH.VP.ENTER enters TDX non-root operation. It returns immediately if TD entry fails. If TD entry succeeded, TDH.VP.ENTER returns when TD exit is initiated.

For partitioned TDs, the TD VCPU may operate in the L1 VM or one of the L2 VMs, if any. TD exit may be initiated from each of the TD's VMs. If the last TD exit was from an L2 VM, TDH.VP.ENTER resumes the same L2 VM, unless the RESUME_L1 input flag is set to 1, instructing TDH.VP.ENTER to resume the L1 VM.

5.4.78.3.2. Enumeration

Control of RBP usage as an input/output parameter by the TD's CONFIG_FLAG.NO_RBP_MOD is enumerated by TDX_FEATURES0.NO_RBP_MOD (bit 18), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, then RBP can be used by TDG.VP.VMCALL to pass information between the guest TD and the host VMM, although highly discouraged since it contradicts normal calling conventions ABI.

Support of the IMM_RESUME_HINT output flag is enumerated by TDX_FEATURES0.ENHANCED_INTR_STATE (bit 40).

Support of the ON input flags in enumerated by TDX_FEATURES0.ENHANCED_INTR_VIRTUALIZATION (bit 45).

5.4.78.3.3. CPU State Preservation Following a Successful TD Entry and a TD Exit

Following a successful TD entry and a TD exit, some CPU state is modified:

- Registers CR2, DR0, DR1, DR2, DR3, DR6 and DR7 are set to their architectural INIT value.
- XCRO is set to the TD's user-mode feature bits of XFAM (bits 7:0, 9).

5.4.78.3.3.1. MSR Preservation Description Files

Most of the MSR virtualization information is provided in separate files:

- **msr_preservation.pdf** provides a human-readable MSR preservation table. The table lists MSRs whose value may not be preserved across TD entry and exit.
- **msr_preservation.csv** provides the same table in a Comma-Separated Values (CSV) format. This file can be imported, if required, to tools such as MS Excel for further processing.

The tables below describe the information provided by those files. Additional information about specific MSRs is provided in the following sections.

In addition to the above files, **msr_preservation.json**, a JSON format file with the same information is provided.

Table 5.321: MSR Preservation Table Description

| Column | Description |
|-------------------------------------|---|
| First (H) | Index of the first MSR in the range |
| Last (H) | Index of the last MSR in the range |
| Size (H) | Number of MSRs in the range |
| MSR Architectural Name | MSR name |
| MSR Value after TDH.VP.ENTER | Impact of TD entry and exit on MSR value; see the table below |

Table 5.322: MSR Value after TDH.VP.ENTER Column Details

| MSR Value after TDH.VP.ENTER | Description |
|------------------------------|--|
| INIT | The TDX Module sets the MSR to its RESET value. |
| Init(condition) | If the condition is true, the TDX Module sets the MSR to its RESET value. Else the MSR value is unmodified. |
| Modified(condition) | If the condition is true, the MSR value may be modified. Else the MSR value is unmodified. |

5.4.78.3.4. Special Environment Requirements

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

5.4.78.3.5. Guest TD State Loading on VM Entry Failure

TDH.VP.ENTER may fail loading guest TD state in the cases shown in the table below. TDH.VP.ENTER returns with information detailing the failure case. Such failures may happen due to the following reasons:

- The TD is being debugged (its ATTRIBUTES.DEBUG bit is set) and the debugger sets some wrong guest state value using TDH.VP.WR. For a debuggable TD, the completion status (in RAX[63:32]) is set in such cases to TDX_SUCCESS, and the details are provided as described below. The debugger may update the VCPU state using TDH.VP.WR and invoke TDH.VP.ENTER again.
- The TD has been migrated, and some of its state is not compatible with the destination platform. The TDX Module does its best effort to check guest state values during import, but there might still be cases where incompatible guest TD state gets migrated. For a non-debuggable TD, the completion status (in RAX[63:32]) is set in such cases to TDX_NON_RECOVERABLE_TD, and the details are provided as described below. The host VMM should tear down the TD.

Table 5.323: Guest State Loading Errors

| Guest State Loading Error | VM Exit Reason in RAX[31:0] | Extended Exit Qualification in RDX |
|---|---|--|
| Error while loading guest MSR values from TDVPS | 34: VM-entry failure due to MSR loading | TD_ENTRY_MSR_LOAD_FAILURE with the MSR index |
| Error while loading CPU extended state from TDVPS | 33: VM-entry failure due to invalid guest state | TD_ENTRY_XSTATE_LOAD_FAILURE |
| VM entry (VMLAUNCH or VMRESUME) which loads guest state from VMCS | 33: VM-entry failure due to invalid guest state | NONE |

5.4.78.3.6. IMM_RESUME_HINT

If supported by the TDX Module, IMM_RESUME_HINT serves as a hint to the host VMM that it should resume the TD as soon as possible, in order for it to process virtual interrupts. Its operation can be configured by the host VMM to be set if one or more TDVPS.VCPU_STATE_DETAILS bits are set. For details, see 4.2.2.3.

5.4.78.3.7. Outstanding Notification (ON)

If supported by the TDX Module, the ON input flags indicate that there is an outstanding posted interrupt set in a PID residing in shared memory (either as a regular PID or as a Shared PID), associated with the current VCPU. This indication serves as a hint, useful mainly when no notification interrupt is configured for that PID. If set, the TDX Module checks the PID's ON bit and handles any posted interrupt requests in the PID's PIR field as part of the TD entry flow.

If TD entry is to an L2 VM, i.e., the last TD exit was from L2 and the host VMM did not set the RESUME_L1 input flag, the TDX Module resumes L1 in the following cases:

ON is set for L1: The TDX Module checks the posted interrupt priority (RVI) vs. L1's current processor priority (Virtual APIC's PPR). If the posted interrupt priority is higher, i.e., L1 will immediately handle the interrupt, the TDX Module emulates an L2→L1 exit and resumes L1.

ON is set for any L2 VM: The TDX Module injects a wakeup interrupt to L1, if configured, and sets the applicable TDCS.WAKEUP_SENT flag. It then does an implicit L2→L1 exit and resumes L1. L1 is expected to handle the wakeup interrupt and enter the targeted L2 VM(s).

5.4.78.3.8. Leaf Function Latency

In some cases (e.g., suspected single/zero step attack mitigation), TDH.VP.ENTER execution time may be longer than most TDX Module interface functions execution time. No interrupts (including NMI and SMI) are processed by the logical processor during that time.

5.4.78.3.9. VCPU Association

TDH.VP.ENTER associates the target TD VCPU with the current LP. This requires that the VCPU will not be associated with another LP. For details, see the [TDX Module Base Spec].

5.4.78.4. Operands Information

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

Table 5.324: TDH.VP.ENTER Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|-----------------------------|------------------|--------|---------------------|----------------|---------------------------|-------------------------|-------------------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | TDVPR page | TDVPS | RW | Opaque | 4KB | Shared(c) ⁴³ | Shared(c) ⁴³ | Shared(c) ⁴³ |
| Implicit | N/A | HPA | TDR page | TDR | RW | Opaque | N/A | Shared(c) ⁴³ | N/A | N/A |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i,c) ⁴³ | N/A | N/A |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared(t) ⁴⁴ | N/A | N/A |
| Implicit | N/A | N/A | TDCS TLB Tracking Fields | N/A | RW | Opaque | N/A | Shared(t) ⁴⁴ | N/A | N/A |
| Implicit | N/A | N/A | SEPT tree | N/A | R | Opaque | N/A | Shared(t) ⁴⁵ | N/A | N/A |

5

5.4.78.5. Completion Status Codes

Table 5.325: TDH.VP.ENTER Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|----------------------------------|--|
| TDX_INCONSISTENT_MSR | IA32_TSC_ADJUST MSR value is different than the value sample by TDH.SYS.INIT. |
| TDX_INCORRECT_MSR_VALUE | |
| TDX_L2_EXIT_HOST_ROUTED_ASYNC | |
| TDX_L2_EXIT_HOST_ROUTED_TDVMCALL | |
| TDX_NON_RECOVERABLE_TD | TDH.VP.ENTER launched or resumed TD VCPU operation (TDX non-root mode) – followed later by a TD exit. The TD state is non-recoverable – further TD entry is prohibited. Exit reason is in RAX bits 31:0. |
| TDX_NON_RECOVERABLE_VCPU | TDH.VP.ENTER launched or resumed TD VCPU operation (TDX non-root mode) – followed later by a TD exit. The TD VCPU state is non-recoverable – further TD entry to this VCPU is prohibited. Exit reason is in RAX bits 31:0. |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |

⁴³ TDVPS, TDR, TDCS and TDCS.OP_STATE are locked (in Shared mode) for the whole duration of running in TDX non-root mode; the locks are released on TD exit.

⁴⁴ Locking of OP_STATE and the TLB tracking fields is temporary; the locks are released before VM entry into the TD VCPU.

⁴⁵ Locking of SEPT tree is temporary; the lock is released before VM entry into the TD VCPU. Earlier releases of the TDX Module, which didn't support Non-Blocking Export, locked the SEPT tree in an Exclusive mode.

| Completion Status Code | Description |
|-------------------------------------|---|
| TDX_OPERAND_BUSY | <p>Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.</p> <p>Note the special case where the indicated operand is TLB_EPOCH. This may happen due to a conflict with TDH.MEM.TRACK or TDH.EXPORT.PAUSE. The host VMM may retry TDH.VP.ENTER.</p> <p>Another special case is where the indicated operand is SEPT_TREE. In some cases, TDH.VP.ENTER may acquire exclusive access on the SEPT tree for a short period of time and may fail due to a concurrent operation. The host VMM should retry TDH.VP.ENTER.</p> |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.VP.ENTER launched or resumed TD VCPU operation (TDX non-root mode) – followed later by a TD exit. Exit reason is in RAX bits 31:0. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |
| TDX_TDCX_NUM_INCORRECT | |
| TDX_TSC_ROLLBACK | |
| TDX_VCPU_ASSOCIATED | |
| TDX_VCPU_STATE_INCORRECT | |

5.4.79. TDH.VP.FLUSH Leaf

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

5.4.79.1. Input Operands**Table 5.326: TDH.VP.FLUSH Input Operands Definition**

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 18 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| 63:24 | Reserved | Must be 0 | |
| RCX | The physical address of a TDVPR page (HKID bits must be 0) | | |

5

5.4.79.2. Output Operands**Table 5.327: TDH.VP.FLUSH Output Operands Definition**

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.79.3. Leaf Function Description

- 10 **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.

5.4.79.4. Operands Information

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

Table 5.328: TDH.VP.FLUSH Operands Information Definition

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

5.4.79.5. *Completion Status Codes*

Table 5.329: TDH.VP.FLUSH Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_LIFECYCLE_STATE_INCORRECT | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.VP.FLUSH is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |
| TDX_TDCX_NUM_INCORRECT | |
| TDX_VCPU_NOT_ASSOCIATED | |

5.4.80. TDH.VP.INIT Leaf

Initialize the saved state of a TD VCPU.

5.4.80.1. Input Operands

Table 5.330: TDH.VP.INIT Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|---|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 22 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version May be 0 or 1 (see the enumeration note below) |
| 63:24 | Reserved | Must be 0 | |
| RCX | The physical address of a TDVPR page (HKID bits must be 0) | | |
| RDX | Initial value of the guest TD VCPU RCX | | |
| R8 | If the version number provided in RAX[23:16] is 0, R8 is ignored. Else, R8 provides the following information: | | |
| | Bits | Field | Description |
| | 31:0 | X2APIC_ID | VCPU's virtual x2APIC ID Must be unique across all VCPUs of the current TD. |
| | 63:32 | Reserved | Must be 0 |

5

5.4.80.2. Output Operands

Table 5.331: TDH.VP.INIT Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| Other | Unmodified |

5.4.80.3. Leaf Function Description

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

5.4.80.3.1. Overview

TDH.VP.INIT initializes the saved state of a VCPU in the TDVPR and TDPX pages.

5.4.80.3.2. Enumeration

- 15 TDH.VP.INIT's support of version 1 or higher is enumerated by TDX_FEATURES0.TOPOLOGY_ENUM (bit 20), readable by TDH.SYS.RD* (see 3.3.3.1), being set to 1. If not supported, calling TDH.VP.INIT with a version number higher than 0 returns a TDX_OPERAND_INVALID(RAX) status.

5.4.80.3.3. 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.

5.4.80.3.4. VCPU Association

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

5.4.80.4. Operands Information

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

Table 5.332: TDH.VP.INIT Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|----------------|------------------|--------|---------------------|----------------|--|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | TDVPR page | TDVPS | RW | Opaque | 4KB | Exclusive(h) | Shared | Shared |
| Implicit | N/A | HPA | TDR page | TDR | R | Opaque | 4KB | Exclusive(h)/ Shared(h) ⁴⁶ | N/A | N/A |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | 4KB | Exclusive(i)/ Shared(i) ⁴⁶ | N/A | N/A |
| Implicit | N/A | N/A | TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Exclusive(i)/ Shared(h) ⁴⁷ | N/A | N/A |

10

5.4.80.5. Completion Status Codes

Table 5.333: TDH.VP.INIT Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_MAX_VCPUS_EXCEEDED | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |

⁴⁶ For backward compatibility, if TDH.VP.INIT is called with version == 0, then TDR is acquired in shared mode. Else, TDR is acquired in exclusive mode. TDCS is implicitly acquired with the same concurrency mode as TDR.

⁴⁷ If TDH.VP.INIT is called with version == 0, then TDCS.OP_STATE is acquired in shared mode. Else, TDCS.OP_STATE is implicitly acquired in exclusive mode, since TDR (and thus the whole TD) is acquired in exclusive mode.

| Completion Status Code | Description |
|----------------------------|----------------------------|
| TDX_SUCCESS | TDH.VP.INIT is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCX_NUM_INCORRECT | |
| TDX_VCPU_ASSOCIATED | |
| TDX_VCPU_STATE_INCORRECT | |
| TDX_X2APIC_ID_NOT_UNIQUE | |

5.4.81. TDH.VP.RD Leaf

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

5.4.81.1. Input Operands**Table 5.334: TDH.VP.RD Input Operands Definition**

| Operand | Description | | |
|---------|--|----------------|---|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 26 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Version number may be 0 or 1. See the enumeration details below. |
| | 63:24 | Reserved | Must be 0 |
| RCX | The physical address of a TDVPR page (HKID bits must be 0) | | |
| RDX | <p>Field identifier – see 3.10</p> <p>The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0.</p> <p>WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored.</p> <p>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.</p> | | |

5

5.4.81.2. Output Operands**Table 5.335: TDH.VP.RD Output Operands Definition**

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| RDX | <p>For TDH.VP.RD version 0, RDX is unmodified.</p> <p>For TDH.VP.RD version 1 or higher:</p> <ul style="list-style-type: none"> If the input field identifier was -1, RDX returns the first readable field identifier. <p>Else, in case of an error, RDX returns -1. On success, RDX returns the next readable field identifier. A value of -1 indicates no next field identifier is available.</p> |
| R8 | <p>Field content</p> <p>In case of no success, as indicated by RAX, R8 returns 0.</p> |
| Other | Unmodified |

5.4.81.3. Leaf Function Description

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

5.4.81.3.1. Overview

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).

5 If version 1 or higher is specified in RAX, RDX returns the next host-side readable field identifier. This may be used by the host VMM to dump the host readable VCPU metadata. To read all the available fields, the host VMM can invoke TDH.VP.RD in a loop, starting with field identifier -1 as an input, until RDX returns -1. A status code of TDX_METADATA_FIELD_SKIP indicates that the returned value is not applicable.

5.4.81.3.2. Enumeration

10 Availability of TDH.VP.RD version 1 is enumerated by TDX_FEATURES0.ENHANCED_METADATA (bit 3), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.VP.RD with a version number higher than 0 returns a TDX_OPERAND_INVALID(RAX) status.

5.4.81.3.3. VCPU Association

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

15 **5.4.81.4. Operands Information**

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

Table 5.336: TDH.VP.RD Operands Information Definition

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

20 **5.4.81.5. Completion Status Codes**

Table 5.337: TDH.VP.RD Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|--|---|
| TDX_METADATA_FIELD_ID_INCORRECT | |
| TDX_METADATA_FIELD_NOT_READABLE | |
| TDX_METADATA_FIELD_SKIP | Indicates that the field value being read is not applicable and needs to be skipped. If called in a loop, use RDX as the identifier of the next field to be read, if any. |
| TDX_METADATA_FIRST_FIELD_ID_IN_CONTEXT | Indicates that the first field ID in context is returned |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |

| Completion Status Code | Description |
|-------------------------------------|--------------------------|
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |
| TDX_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.VP.RD is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TDCS_NOT_ALLOCATED | |
| TDX_TDCX_NUM_INCORRECT | |
| TDX_VCPU_ASSOCIATED | |
| TDX_VCPU_STATE_INCORRECT | |

5.4.82. TDH.VP.WR Leaf

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

5.4.82.1. Input Operands**Table 5.338: TDH.VP.WR Input Operands Definition**

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | SEAMCALL instruction leaf number and version, see 5.4.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the SEAMCALL interface function: 43 |
| | 23:16 | Version Number | Selects the SEAMCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | The physical address of a TDVPR page (HKID bits must be 0) | | |
| RDX | Field identifier – see 3.10 The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0. WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored. | | |
| R8 | 64b value to write to the field | | |
| R9 | A 64b write mask to indicate which bits of the value in R8 are to be written to the field | | |

5

5.4.82.2. Output Operands**Table 5.339: TDH.VP.WR Output Operands Definition**

| Operand | Description |
|---------|--|
| RAX | SEAMCALL instruction return code – see 5.4.1 |
| R8 | Previous content of the field In case of an error, as indicated by RAX, R8 returns 0. |
| Other | Unmodified |

5.4.82.3. Leaf Function Description

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

5.4.82.3.1. Overview

15 TDH.VP.WR writes a TDVPS field, given its field code. The value (R8) is written as specified by the write mask (R9). Writing is subject to the field's internal write mask (per the TD's ATTRIBUTES.DEBUG bit). Writing of specific fields is also subject to additional rules as detailed in 4.2.

Table 5.340: Metadata Field Write Rules

| Write Mask Bit in R9 | Internal Write Mask Bit | Value Bit in R8 |
|----------------------|-------------------------|---|
| 0 | N/A | Silently ignored |
| 1 | 0 | Must be the same as the current field's bit |
| 1 | 1 | Written to the current field's bit |

TDH.VP.WR returns the previous content of the field masked by the field's readability (per the TD's ATTRIBUTES.DEBUG bit).

5.4.82.3.2. VCPU Association

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

5.4.82.4. Operands Information

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

Table 5.341: TDH.VP.WR Operands Information Definition

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

5.4.82.5. Completion Status Codes

Table 5.342: TDH.VP.WR Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------------|--|
| TDX_METADATA_FIELD_ID_INCORRECT | |
| TDX_METADATA_FIELD_NOT_READABLE | |
| TDX_METADATA_FIELD_NOT_WRITABLE | |
| TDX_METADATA_FIELD_VALUE_NOT_VALID | |
| TDX_METADATA_WR_MASK_NOT_VALID | |
| TDX_OP_STATE_INCORRECT | |
| TDX_OPERAND_ADDR_RANGE_ERROR | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_PAGE_METADATA_INCORRECT | |

| Completion Status Code | Description |
|-----------------------------------|--------------------------|
| TDX_PAGE_METADATA_INCORRECT | |
| TDX_SUCCESS | TDH.VP.WR is successful. |
| TDX_SYS_NOT_READY | |
| TDX_SYS_SHUTDOWN | |
| TDX_TD_FATAL | |
| TDX_TD_KEYS_NOT_CONFIGURED | |
| TDX_TD_VMCS_FIELD_NOT_INITIALIZED | |
| TDX_TDCS_NOT_ALLOCATED | |
| TDX_TDCX_NUM_INCORRECT | |
| TDX_TDVPD_FIELD_NOT_WRITABLE | |
| TDX_VCPU_ASSOCIATED | |
| TDX_VCPU_STATE_INCORRECT | |
| TDX_TD_VMCS_FIELD_NOT_INITIALIZED | |

5.5. Guest-Side (TDCALL) Interface Functions

The TDCALL instruction causes a VM exit to the Intel TDX Module. It is used to call guest-side Intel TDX functions, either local or a TD exit to the host VMM, as selected by RAX.

5.5.1. TDCALL Instruction (Common)

- 5 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.

5.5.1.1. Input Operands

Table 5.343: TDCALL Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|---|
| RAX | Leaf and version numbers, as defined in the [TDX Module Base Spec]. See Table 5.345 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. | | |

10 5.5.1.2. Output Operands

Table 5.344: TDCALL Output Operands Definition

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

5.5.1.3. Instruction Description

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

This section describes how TDCALL leaf functions are implemented by the Intel TDX Module.

The TDCALL instruction itself is specified in the [TDX Arch Extensions Spec]. There are multiple cases where TDCALL may fail. Failures may result in an exception (#UD, #GP(0)). Failure cases include, among others, the following:

- CPU mode is incorrect
- 20 • Privilege level is not 0

TDCALL results in a VM exit to the TDX Module. On VM exit, the Intel TDX Module performs the following checks:

1. If the CPU mode is not 64b ((IA32_EFER.LMA == 1) && (CS.L == 1)), the Intel TDX Module injects a #GP(0) fault into the guest TD.
 2. If the leaf number in RAX is not supported by the Intel TDX Module, it returns a TDX_OPERAND_INVALID(0) status code in RAX.
- 25

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.

5.5.1.4. TDCALL Leaf Numbers

The table below summarizes the guest-side interface functions leaf numbers. Note the following:

- For support of specific interface functions by specific TDX Module releases, see the detailed description of each function.
- TDX Connect interface function leaf numbers are defined in the [TDX Connect ABI Spec].

Table 5.345: 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 guest-readable TDX Module global-scope metadata fields |
| 13 | TDG.SYS.RDM | Read multiple TDX Module global-scope metadata fields |
| 14 | TDG.VM.RDM | Read multiple TD-scope metadata fields |
| 15 | TDG.VM.WRM | Write multiple TD-scope metadata fields |
| 16 | TDG.VP.RDM | Read multiple VCPU-scope metadata fields |
| 17 | TDG.VP.WRM | Write multiple VCPU-scope metadata fields |
| 18 | TDG.SERVTD.RD | Read a target TD metadata field |
| 19 | TDG.SERVTD.RDM | Read multiple target TD metadata fields |
| 20 | TDG.SERVTD.WR | Write a target TD metadata field |
| 21 | TDG.SERVTD.WRM | Write multiple target TD metadata fields |
| 22 | TDG.MR.VERIFYREPORT | Verify a cryptographic report of a TD, generated on the current platform |
| 23 | TDG.MEM.PAGE.ATTR.RD | Read the GPA mapping and attributes of a TD private page |
| 24 | TDG.MEM.PAGE.ATTR.WR | Write the attributes of a private page |
| 25 | TDG.VP.ENTER | Enter L2 VCPU operation |
| 26 | TDG.VP.INVEPT | Invalidate cached EPT translations for selected L2 VMs |
| 27 | TDG.VP.INVGLA | Invalidate cached translations for selected pages in an L2 VM |
| 28 | TDG.MR.ASSIGNSVNS | Assign SVN values given TDSIGSTRUCTS and reference TD measurement values |
| 29 | TDG.MR.KEY.GET | Get a persistent key, customized to the TD's measurements and policy |
| 30 | TDG.MEM.PAGE.RELEASE | Release a private page, enabling the host VMM to remove it |

5.5.1.5. Completion Status Codes

Table 5.346: TDCALL Completion Status Codes (Returned in RAX) Definition

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

5.5.2. TDG.INTR.POST

Unreleased Feature: This section is related to Enhanced Interrupt Virtualization, a feature which has not been released yet at the time of writing of this document. Details provided below serve as a preview and are subject to change.

- 5 This function is used by L1 to post a virtual interrupt to an arbitrary VCPU of an L2 VM.

5.5.2.1. Input Operands

Table 5.347: TDG.INTR.POST Input Operands Definition

| Operand | Name | Description | | |
|---------|-------------------|---|----------------|--|
| RAX | Leaf and Version | TDCALL instruction leaf number and version, see 5.4.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the TDCALL interface function: 32 |
| | | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | | 63:24 | Reserved | Must be 0 |
| RCX | VM_FLAGS | VM identifier and flags | | |
| | | Bits | Name | Description |
| | | 51:0 | Reserved | Reserved: must be 0 |
| | | 53:52 | VM | L2 virtual machine index (must be 1 or higher) |
| | | 63:54 | Reserved | Reserved: must be 0 |
| RDX | DESTINATION_INDEX | Index in the destination L2 VM's PIDPT Value must be lower than the number of entries in that PIDPT. | | |
| R8 | VECTOR | The vector of the interrupt to be posted VECTOR's value must be between 31 and 255. | | |

5.5.2.2. Output Operands

10

Table 5.348: TDG.INTR.POST Output Operands Definition

| Operand | | Description |
|---------|--------|--------------------------------|
| RAX | Status | TDCALL instruction return code |
| Other | | Unmodified |

5.5.2.3. Leaf Function Description

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

15 5.5.2.3.1. Overview

TDG.INTR.POST, called by L1, posts a virtual inter-processor interrupt to requested L2 VM VCPU.

IPI must have been properly configured to that L2 VM. See the [Interrupt Virtualization Spec] chapter titled "Guest TD's L1 Operation as a VMM of L2 VMs" for details.

5.5.2.3.2. Enumeration

Support of TDG.INTR.POST is enumerated by TDX_FEATURES0.ENHANCED_INTR_VIRTUALIZATION (bit 45), readable by TDG.SYS.RD*.

5.5.2.3.3. Non-Running Destination VCPU Case

- 5 If TDG.INTR.POST detects that the destination VCPU is not currently running, it causes a TD exit, indicating an TDX_IPI_REQUEST. The host VMM can schedule the destination VCPU, which will handle the IPI on TD entry. See the description of TDH.VP.ENTER for details.

From the calling L1 perspective, this case behaves as a successful completion of TDG.INTR.POST. When the host VMM resumes the TD (TDH.VP.ENTER), L1 sees a TDX_SUCCESS status in RAX.

10 5.5.2.4. Operands Information

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

Table 5.349: TDG.INTR.POST Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|-----------------|---------------|--------|---------------------|----------------|-----------------------------|
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | R | Opaque | N/A | Shared(i) |

15 5.5.2.5. Completion Status Codes

Table 5.350: TDG.INTR.POST Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|---|------------------------------|
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDG.INTR.POST is successful. |
| Unreleased Feature: This list is not comprehensive | |

5.5.3. TDG.MEM.PAGE.ACCEPT Leaf

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

5.5.3.1. Input Operands

5

Table 5.351: TDG.MEM.PAGE.ACCEPT Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 6 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | EPT mapping information: | | |
| | Bits | Name | Description |
| | 2:0 | Level | Level of the Secure EPT leaf entry that maps the private page to be accepted: either 0 (4KB) or 1 (2MB) – see 3.6.1. |
| | 11:3 | Reserved | Reserved: must be 0 |
| | 51:12 | GPA | Bits 51:12 of the guest physical address of the private page to be accepted |
| | 63:52 | Reserved | Reserved: must be 0 |

5.5.3.2. Output Operands

Table 5.352: TDG.MEM.PAGE.ACCEPT Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | TDCALL instruction return code – see 5.5.1 |
| Other | Unmodified |

10

5.5.3.3. Leaf Function Description

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

5.5.3.3.1. Overview

15

TDG.MEM.PAGE.ACCEPT accepts a pending private page, previously added by TDH.MEM.PAGE.AUG or released by TDG.MEM.PAGE.RELEASE, into the TD. It initializes the page to 0. If page attributes have been set by the guest TD, using TDG.MEM.PAGE.ATTR.WR, while the page was pending, they become effective when the page is accepted.

5.5.3.3.2. SEPT Mapping Size Considerations

In most cases, the guest TD is unaware of how TD private pages are mapped by the host VMM in SEPT. However, TDG.MEM.PAGE.ACCEPT operation specifies a page mapping size and may fail if the specified size is different than the actual mapping size.

- If the page is mapped at a lower level than requested, the function returns TDX_PAGE_SIZE_MISMATCH. The guest may re-invoke TDG.MEM.PAGE.ACCEPT specifying a 4KB page size.
- If the page is mapped at a higher level than requested, this results in an EPT violation TD exit, with extended exit qualification indicating the error SEPT entry level and state, and the guest-requested mapping level. The host VMM is expected to demote the page, then re-enter the guest TD so TDG.MEM.PAGE.ACCEPT is re-invoked.

5.5.3.3.3. Other Conditions that Prevent Page Acceptance

- If the page has already been accepted, the function returns TDX_PAGE_ALREADY_ACCEPTED.
- If the page is not PENDING nor PENDING_EXPORTED_DIRTY, this results in an EPT violation TD exit, with extended exit qualification indicating the error SEPT entry level and state, and the guest-requested accept level.

5.5.3.3.4. Interruptibility

If, during its execution, TDG.MEM.PAGE.ACCEPT detects that an external interrupt is pending, it may resume the guest TD with the CPU state unmodified. The progress so far is recorded in the page’s Secure EPT entry. This allows the external interrupt to be recognized, causing a TD exit or a posted interrupt delivery. Typically, TDG.MEM.PAGE.ACCEPT will be re-invoked and continue its work.

Guest TD software is not directly involved. Guest TD should not precede the TDCALL with an STI instruction or a MOV to SS instruction. Posted interrupts may be delivered when the TDCALL flow is interrupted.

5.5.3.4. Operands Information

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

Table 5.353 TDG.MEM.PAGE.ACCEPT Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|------------------|------------------|--------|---------------------|----------------------------------|-----------------------------|
| Explicit | RCX | GPA | TD private page | Blob | RW | Private | 2 ^{12+9*Level} Bytes | None |
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | RW | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | Secure EPT tree | N/A | RW | Private | N/A | None |
| Implicit | N/A | GPA | Secure EPT entry | SEPT Entry | RW | Private | N/A | Exclusive(h) |

5.5.3.5. Completion Status Codes

Table 5.354: TDG.MEM.PAGE.ACCEPT Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------|---|
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. Specifically, it may indicate that a concurrent TDG.MEM.PAGE.ACCEPT is using the same Secure EPT entry |

| Completion Status Code | Description |
|---------------------------|--|
| TDX_PAGE_ALREADY_ACCEPTED | |
| TDX_PAGE_SIZE_MISMATCH | Requested page size is 2MB, but the page GPA is not mapped at 2MB size |
| TDX_SUCCESS | TDG.MEM.PAGE.ACCEPT is successful. |

5.5.4. TDG.MEM.PAGE.ATTR.RD Leaf

Read the GPA mapping and attributes of a TD private page or a private MMIO page⁴⁸.

5.5.4.1. Input Operands

Table 5.355: TDG.MEM.PAGE.ATTR.RD Input Operands Definition

| Operand | Name | Description | | |
|---------|------------------|---|----------------|--|
| RAX | Leaf and Version | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the TDCALL interface function: 23 |
| | | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 | |
| RCX | GPA | Guest physical address | | |

5

5.5.4.2. Output Operands

Table 5.356: TDG.MEM.PAGE.ATTR.RD Output Operands Definition

| Operand | Name | Description | | |
|---------|-------------|--|-------------|--|
| RAX | STATUS | TDCALL instruction return code – see 5.5.1 | | |
| RCX | GPA_MAPPING | Actual GPA mapping of the page: | | |
| | | Bits | Name | Description |
| | | 2:0 | LEVEL | Level of the Secure EPT leaf entry that maps the private page: either 0 (4KB), 1 (2MB) or 2 (1GB) – see 3.6.1. |
| | | 11:3 | RESERVED | Reserved: set to 0 |
| | | 51:12 | GPA | Bits 51:12 of the guest physical start address of the private page Depending on the level, the following least significant bits are always 0: Level 0 (EPTE): None Level 1 (EPDE): Bits 20:12 Level 2 (EPDPTE): Bits 29:12 |
| | | 60:52 | RESERVED | Reserved: set to 0 |
| | | 61 | MMIO | Flags that the page is MMIO |
| 62 | PENDING | Flags that the page is PENDING This is applicable to all PENDING states; if the TDX Module supports TDX Connect, it is also applicable to MMIO_PENDING pages. | | |

⁴⁸ Applicable only if the TDX Module supports TDX Connect.

| Operand | Name | Description | | |
|---------|----------|--|----------|--------------------|
| | | 63 | RESERVED | Reserved: set to 0 |
| RDX | GPA_ATTR | Guest-visible page attributes. See the GPA_ATTR definition in 3.6.3. | | |
| Other | | Unmodified | | |

5.5.4.3. Leaf Function Description

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

5 **5.5.4.3.1. Overview**

TDH.MEM.PAGE.ATTR.RD reads the GPA mapping and attributes of a TD private page. Given a GPA (which can be anywhere within a page) it returns the actual mapping level – either 0 (4KB), 1 (2MB) or 2 (1GB) – and the guest-readable attributes. TDH.MEM.PAGE.ATTR.RD can read the attributes of a PENDING page.

10 GPA mapping level is exposed to the guest TD since page acceptance (TDH.MEM.PAGE.ACCEPT) and page attributes modifications and L2 page aliases management (TDH.MEM.PAGE.ATTR.WR) are done at mapping granularity.

5.5.4.3.2. Enumeration

Availability of TDG.MEM.PAGE.ATTR.RD is enumerated by TDX_FEATURES0.TD_PARTITIONING (bit 7), readable by TDG.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.MEM.PAGE.ATTR.RD returns a TDX_OPERAND_INVALID(RAX) status.

15 Support of TDX Connect is enumerated by TDX_FEATURES0.TDX_CONNECT (bit 6) and TDX_FEATURES0.TDX_CONNECT_PARTITIONING (bit 32).

5.5.4.3.3. EPT Violation

If the requested GPA is not guest-readable and not pending acceptance, TDH.MEM.PAGE.ATTR.RD causes an EPT violation TD exit.

20 **5.5.4.4. Operands Information**

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

Table 5.357 TDG.MEM.PAGE.ATTR.RD Memory Operands Information Definition

| Explicit/Implicit | Reg. | Addr. Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-------------------|------|------------|-----------------------|---------------|--------|------------------|-------------|--------------------------|
| Explicit | RCX | GPA | TD private page | Blob | RW | Private | None | None |
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | Secure EPT tree | N/A | R | Private | N/A | None |
| Implicit | N/A | N/A | L2 Secure EPT trees | N/A | RW | Private | N/A | None |
| Implicit | N/A | GPA | Secure EPT entry | SEPT Entry | R | Private | N/A | Exclusive(h) |
| Implicit | N/A | GPA | L2 Secure EPT entries | SEPT Entry | RW | Private | N/A | Exclusive(i) |

5.5.4.5. Completion Status Codes

Table 5.358: TDG.MEM.PAGE.ATTR.RD Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------|-------------------------------------|
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDG.MEM.PAGE.ATTR.RD is successful. |

5.5.5. TDG.MEM.PAGE.ATTR.WR Leaf

Write the attributes of a private page or a private MMIO page⁴⁹. Create or remove L2 page aliases as required.

5.5.5.1. Input Operands

Table 5.359: TDG.MEM.PAGE.ATTR.WR Input Operands Definition

| Operand | Name | Description | | |
|---------|------------------|---|----------------|---|
| RAX | Leaf and Version | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the TDCALL interface function: 24 |
| | | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | | 63:24 | Reserved | Must be 0 |
| RCX | GPA_MAPPING | GPA mapping information: | | |
| | | Bits | Name | Description |
| | | 2:0 | LEVEL | Level of the Secure EPT leaf entry that maps the private page: either 0 (4KB), 1 (2MB) or 2 (1GB) – see 3.6.1. |
| | | 11:3 | RESERVED | Reserved: must be 0 |
| | | 51:12 | GPA | Bits 51:12 of the guest physical address of the private page Depending on the level, the following least significant bits must be 0: Level 0 (EPTE): None Level 1 (EPDE): Bits 20:12 Level 2 (EPDPTE): Bits 29:12 |
| 63:52 | RESERVED | Reserved: must be 0 | | |
| RDX | GPA_ATTR | Guest-visible page attributes. See the GPA_ATTR definition in 3.6.3. To avoid writing the attributes of a certain VM, all 16 attribute bits (GPA_ATTR_SINGLE_VM) for that VM should be set to 0. Attribute bits for non-existent VMs must be 0. | | |
| R8 | ATTR_FLAGS | Attributes masks and invalidate EPT flags | | |
| | | Bits | Field | Description |
| | | 15:0 | RESERVED | Must be 0 |
| | | 30:16 | ATTR_MASK1 | A bit value of 1 indicates that the applicable attributes bit is to be written. Otherwise, the attributes bit is unmodified. Must be 0 if the TD has no VM #1. |
| | | 31 | INVEPT1 | Invalidate EPT for L2 VM #1 Must be 0 if the TD has no VM #1. |

⁴⁹ Applicable only if the TDX Module supports TDX Connect.

| Operand | Name | Description | | |
|---------|------|-------------|------------|---|
| | | 46:32 | ATTR_MASK2 | A bit value of 1 indicates that the applicable attributes bit is to be written. Otherwise, the attributes bit is unmodified. Must be 0 if the TD has no VM #2. |
| | | 47 | INVEPT2 | Invalidate EPT for L2 VM #2 Must be 0 if the TD has no VM #2. |
| | | 62:48 | ATTR_MASK3 | A bit value of 1 indicates that the applicable attributes bit is to be written. Otherwise, the attributes bit is unmodified. Must be 0 if the TD has no VM #3. |
| | | 63 | INVEPT3 | Invalidate EPT for L2 VM #3 Must be 0 if the TD has no VM #3. |

5.5.5.2. Output Operands

Table 5.360: TDG.MEM.PAGE.ATTR.WR Output Operands Definition

| Operand | Name | Description | | |
|---------|-------------|--|----------|--|
| RAX | STATUS | TDCALL instruction return code – see 5.5.1 | | |
| RCX | GPA_MAPPING | GPA mapping of the page: | | |
| | | Bits | Name | Description |
| | | 2:0 | LEVEL | The requested level of the Secure EPT leaf entry that maps the private page: either 0 (4KB), 1 (2MB) or 2 (1GB) – see 3.6.1. If the returned STATUS is TDX_PAGE_SIZE_MISMATCH, the actual level is lower than the requested one. |
| | | 11:3 | RESERVED | Reserved: set to 0 |
| | | 51:12 | GPA | Bits 51:12 of the requested guest physical start address of the private page Depending on LEVEL, the following least significant bits are always 0: Level 0 (EPTE): None Level 1 (EPDE): Bits 20:12 Level 2 (EPDPTE): Bits 29:12 |
| | | 60:52 | RESERVED | Reserved: set to 0 |
| | | 61 | MMIO | Flags that the page is MMIO This bit is only applicable if TDX Module supports TDX Connect, and the returned STATUS is TDX_SUCCESS. |
| | | 62 | PENDING | Flags that the page is PENDING This is applicable to all PENDING states; if the TDX Module supports TDX Connect, it is also applicable to MMIO_PENDING pages. This bit is only applicable if the returned STATUS is TDX_SUCCESS. |

| Operand | Name | Description |
|---------|----------|--|
| | | 63 RESERVED Reserved: set to 0 |
| RDX | GPA_ATTR | <p>On success, if the attribute bits (GPA_ATTR_SINGLE_VM) for a specific VM were 0 on input, they remain unmodified. For other VMs, RDX returns the updated guest-visible page attributes.</p> <p>In case of an error, RDX returns the current value of page attributes when possible. If the current attributes for a certain VM have not been read, that VM's attributes VALID bit returns 0.</p> <p>See the GPA_ATTR definition in 3.6.3.</p> |
| Other | | Unmodified |

5.5.5.3. Leaf Function Description

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

5 5.5.5.3.1. Overview

TDH.MEM.PAGE.ATTR.WR writes the specified set of attributes of a TD private page, including L2 page alias attributes. Only the bits selected by the attributes mask are updated. The private page can be either writable by the TD (MAPPED or EXPORTED_DIRTY) or pending acceptance (PENDING or PENDING_EXPORTED_DIRTY). If the page is pending acceptance, the written attributes will become effective when the page is later accepted by the guest TD, using TDG.MEM.PAGE.ACCEPT.

TDH.MEM.PAGE.ATTR.WR ignores any VM's GPA attributes set whose VALID bit is 0.

TDH.MEM.PAGE.ATTR.WR creates or removes L2 page aliases as required:

- If any of the requested L2 attributes VALID bit is set, and the R, W, Xs, Xu and PWA bits combination has a legal, non-0 value, then if the L2 page alias does not exist, it is created. The rules for checking the legal combination of attributes bits are described in 3.6.3.1.
- If any of the requested L2 attributes VALID bit is set, and the R, W, Xs, Xu and PWA bits are all 0, then if the L2 page alias exists, it is removed.

Note: For the above operations, the Xu and PWA bits are always considered, regardless of the L2 VMCS setting of the "mode-based execute control for EPT" (MBEC) and "EPT paging-write control" VM-execution controls.

20 5.5.5.3.2. Enumeration

Availability of TDG.MEM.PAGE.ATTR.WR is enumerated by TDX_FEATURES0.TD_PARTITIONING (bit 7), readable by TDG.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.MEM.PAGE.ATTR.WR returns a TDX_OPERAND_INVALID(RAX) status.

Support of TDX Connect is enumerated by TDX_FEATURES0.TDX_CONNECT (bit 6) and TDX_FEATURES0.TDX_CONNECT_PARTITIONING (bit 32).

25 5.5.5.3.3. SEPT Mapping Size Considerations

In most cases, the guest TD is unaware of how TD private pages are mapped by the host VMM in SEPT. However, TDG.MEM.PAGE.ATTR.WR operation specifies a page mapping size and may fail if the specified size is different than the actual mapping size.

- If the page is mapped at a lower level than requested, the function returns TDX_PAGE_SIZE_MISMATCH. The guest may re-invoke TDG.MEM.PAGE.ATTR.WR specifying the actual mapping size as returned in RCX.
- If the page is mapped at a higher level than requested, this results in an EPT violation TD exit, with extended exit qualification indicating the error SEPT entry level and state, and the guest-requested mapping level. The host VMM is expected to demote the page, then re-enter the guest TD so TDG.MEM.PAGE.ATTR.WR is re-invoked.

5.5.5.3.4. Other Conditions that Prevent Page Attributes Modifications

- If the page is not guest-writable and is not pending, this results in an EPT violation TD exit, indicating a failed write operation.
- If an L2 SEPT walk fails, meaning there's a missing non-leaf L2 SEPT page, the operation depends on the setting of the host writable TDCS field VM_CTL5, which is an array of 4 bitmaps, one per VM (only L2 VMs are applicable). Bit 0 controls the operation on L2 SEPT walk fails in TDCALL flows:
 - The default value of 0 means that a TDX_L2_SEPT_WALK_FAILED status is returned to the L1 VMM.
 - If the value is 1, the TDX Module does an EPT violation TD exit, indicating a failed write operation exit, with extended exit qualification indicating the error L2 SEPT level and VM index. The host VMM may then add the missing L2 SEPT page using TDH.MEM.SEPT.ADD.

In any of the above cases, the page attributes are not modified.

5.5.5.4. Operands Information

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

Table 5.361 TDG.MEM.PAGE.ATTR.WR Memory Operands Information Definition

| Explicit/ Implicit | Reg. | Addr. Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|---------------|-----------------------|------------------|--------|---------------------|----------------------------------|-----------------------------|
| Explicit | RCX | GPA | TD private page | Blob | RW | Private | 2 ^{12+9*Level} Bytes | None |
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | L1 Secure EPT tree | N/A | R | Private | N/A | None |
| Implicit | N/A | N/A | L2 Secure EPT trees | N/A | RW | Private | N/A | None |
| Implicit | N/A | GPA | L1 Secure EPT entry | SEPT Entry | RW | Private | N/A | Exclusive(h) |
| Implicit | N/A | GPA | L2 Secure EPT entries | SEPT Entry | RW | Private | N/A | Exclusive(i) |

5.5.5.5. Completion Status Codes

Table 5.362: TDG.MEM.PAGE.ATTR.WR Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------|--|
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_BUSY | |
| TDX_PAGE_ATTR_INVALID | The combination of page attributes to be set, after considering the requested attributes, the requested attributes mask and the current page attributes, is invalid. |
| TDX_PAGE_SIZE_MISMATCH | Requested page size does not match its GPA mapping size |
| TDX_SUCCESS | TDG.MEM.PAGE.ATTR.WR is successful. |

5.5.6. TDG.MEM.PAGE.RELEASE Leaf

Release a private page, enabling the host VMM to remove it.

5.5.6.1. Input Operands

Table 5.363: TDG.MEM.PAGE.RELEASE Input Operands Definition

| Operand | Name | Description | | |
|---------|------------------|---|----------------|---|
| RAX | Leaf and Version | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the TDCALL interface function: 30 |
| | | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | | 63:24 | Reserved | Must be 0 |
| RCX | GPA_MAPPING | GPA mapping information: | | |
| | | Bits | Name | Description |
| | | 2:0 | LEVEL | Level of the Secure EPT leaf entry that maps the private page: either 0 (4KB), 1 (2MB) or 2 (1GB) – see 3.6.1. |
| | | 11:3 | RESERVED | Reserved: must be 0 |
| | | 51:12 | GPA | Bits 51:12 of the guest physical address of the private page Depending on the level, the following least significant bits must be 0: Level 0 (EPTE): None Level 1 (EPDE): Bits 20:12 Level 2 (EPDPTE): Bits 29:12 |
| 63:52 | RESERVED | Reserved: must be 0 | | |

5

5.5.6.2. Output Operands

Table 5.364: TDG.MEM.PAGE.RELEASE Output Operands Definition

| Operand | Name | Description | | |
|---------|-------------|---|-------------|---|
| RAX | STATUS | TDCALL instruction return code –see 5.5.1 | | |
| RCX | GPA_MAPPING | GPA mapping of the page: | | |
| | | Bits | Name | Description |
| | | 2:0 | LEVEL | The requested level of the Secure EPT leaf entry that maps the private page: either 0 (4KB), 1 (2MB) or 2 (1GB) – see 3.6.1. If the returned STATUS is TDX_PAGE_SIZE_MISMATCH, the actual level is lower than the requested one. |
| | | 11:3 | RESERVED | Reserved: set to 0 |

| Operand | Name | Description | | |
|---------|------|-------------|----------|--|
| | | 51:12 | GPA | Bits 51:12 of the guest physical start address of the private page Depending on LEVEL, the following least significant bits are always 0: Level 0 (EPTE): None Level 1 (EPDE): Bits 20:12 Level 2 (EPDPTE): Bits 29:12 |
| | | 60:52 | RESERVED | Reserved: set to 0 |
| | | 61 | MMIO | Flags that the page is MMIO This bit is only applicable if TDX Module supports TDX Connect, and the returned STATUS is either TDX_SUCCESS or TDX_EPT_STATE_INCORRECT. |
| | | 62 | PENDING | Flags that the page is PENDING This is applicable to all PENDING states; if the TDX Module supports TDX Connect, it is also applicable to MMIO_PENDING pages. This bit is only applicable if the returned STATUS is either TDX_SUCCESS or TDX_EPT_STATE_INCORRECT. |
| | | 63 | RESERVED | Reserved: set to 0 |
| Other | | Unmodified | | |

5.5.6.3. Leaf Function Description

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

5 5.5.6.3.1. Overview

TDG.MEM.PAGE.RELEASE releases a TD private page and puts it in a PENDING state, allowing the host VMM to remove it even if the TD is enabled for TDX Connect. Releasing a page prevents creating new address translations to that page; to remove the released page, the host VMM must first block the page using TDH.MEM.RANGE.BLOCK and perform TLB shutdown as described in the [Base Spec].

10 5.5.6.3.2. Enumeration

TDX Module support of TDG.MEM.PAGE.RELEASE is enumerated by TDX_FEATURES0.PAGE_RELEASE (bit 38), readable by TDG.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.MEM.PAGE.RELEASE returns a TDX_OPERAND_INVALID(RAX) status.

5.5.6.3.3. Configuration by the Host VMM

15 TDG.MEM.PAGE.RELEASE is available to the guest TD if CONFIG_FLAGS.PAGE_RELEASE is 1; this happens if the host VMM set either CONFIG_FLAGS.PAGE_RELEASE or CONFIG_FLAGS.TDX_CONNECT to 1. If not configured as available, calling TDG.MEM.PAGE.RELEASE returns a TDX_OPERAND_INVALID(RAX) status.

CONFIG_FLAGS is readable by the guest TD using TDG.VM.RD.

5.5.6.3.4. SEPT Mapping Size Considerations

In most cases, the guest TD is unaware of how TD private pages are mapped by the host VMM in SEPT. However, TDG.MEM.PAGE.RELEASE operation specifies a page mapping size and may fail if the specified size is different than the actual mapping size.

- If the page is mapped at a lower level than requested, the function returns TDX_PAGE_SIZE_MISMATCH. The guest may re-invoke TDG.MEM.PAGE.RELEASE specifying the actual mapping size as returned in RCX.
 - If the page is mapped at a higher level than requested, this results in an EPT violation TD exit, with extended exit qualification indicating the error SEPT entry level and state, and the guest-requested mapping level. The host VMM is expected to demote the page, then re-enter the guest TD so TDG.MEM.PAGE.RELEASE is re-invoked.
- Note:** 1GB pages may be released by TDG.MEM.PAGE.RELEASE. However, TDG.MEM.PAGE.ACCEPT will not be able to directly accept them. Calling TDH.MEM.PAGE.ACCEPT on 1GB pages results in an EPT violation TD exit, and the host VMM is expected to demote the page using TDH.MEM.PAGE.DEMOTE.

5.5.6.3.5. Other Conditions that Prevent Page Release

- The guest TD should not attempt to release a page that is not allocated to its GPA space. In such cases, a TDX_EPT_ENTRY_FREE status is returned.
- The guest TD should not attempt to release PENDING pages. If the page state is any of the PENDING states, a TDX_EPT_ENTRY_STATE_INCORRECT status is returned to the guest TD, and RCX.PENDING is set.
- The guest TD should not attempt to release MMIO pages. If the page state is any of the MMIO states, a TDX_EPT_ENTRY_STATE_INCORRECT status is returned to the guest TD, and RCX.MMIO is set.
- If the page is not accessible due to a host VMM operation (that the guest is TD not aware of), such as blocked by TDH.MEM.RANGE.BLOCK or write-blocked by TDH.EXPORT.BLOCKW, TDG.MEM.PAGE.RELEASE results in an EPT violation TD exit, indicating a failed write operation. The host VMM is expected to unblock the page and re-enter the TD.

5.5.6.4. Operands Information

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

Table 5.365 TDG.MEM.PAGE.RELEASE 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(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | L1 Secure EPT tree | N/A | R | Private | N/A | None |
| Implicit | N/A | N/A | L2 Secure EPT trees | N/A | RW | Private | N/A | None |
| Implicit | N/A | GPA | L1 Secure EPT entry | SEPT Entry | RW | Private | N/A | Exclusive(h) |
| Implicit | N/A | GPA | L2 Secure EPT entries | SEPT Entry | RW | Private | N/A | Exclusive(i) |

5.5.6.5. Completion Status Codes

Table 5.366: TDG.MEM.PAGE.RELEASE Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------|-------------|
| TDX_EPT_ENTRY_FREE | |
| TDX_EPT_ENTRY_STATE_INCORRECT | |

| Completion Status Code | Description |
|------------------------|--|
| TDX_OPERAND_INVALID | |
| TDX_OPERAND_BUSY | |
| TDX_PAGE_PENDING | |
| TDX_PAGE_SIZE_MISMATCH | The requested page size does not match its GPA mapping size. |
| TDX_SUCCESS | TDG.MEM.PAGE.RELEASE is successful. |

5.5.7. TDG.MR.ASSIGNSVNS

Unreleased Feature: This section is related to TD Signing, a feature which has not been released yet at the time of writing of this document. Details provided below serve as a preview and are subject to change.

TDG.MR.ASSIGNSVNS allows a TD author to assign Product Identifiers and Security Version Numbers to TD images that they produce. These values are assigned by shipping a signed TDSIGSTRUCT containing reference values for the TD measurements, and associated SVNs. This TDSIGSTRUCT is provided to TDG.MR.ASSIGNSVNS for processing during TD startup.

5.5.7.1. Input Operands

Table 5.367: TDG.MR.ASSIGNSVNS Input Operand Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 28 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | 2048B-aligned guest physical address of TDSIGSTRUCT containing the signer and SVNs for the TD. | | |

5.5.7.2. Output Operands

Table 5.368: TDG.MR.ASSIGNSVNS Output Operand Definition

| Operand | Description |
|---------|--|
| RAX | TDCALL instruction return code – see 5.5.1 |
| Other | Unmodified |

5.5.7.3. Leaf Function Description

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

5.5.7.3.1. Overview

TDG.MR.ASSIGNSVNS takes GPA of a signed TDSIGSTRUCT containing reference values for a subset of available TD measurements, and an associated Product ID and SVN to be assigned if those reference values match the measurements of the TD. The hash of the public key of the TDSIGSTRUCT is recorded. TDG.MR.ASSIGNSVNS can be called multiple times to process a chain of TDSIGSTRUCT similar to a certificate chain starting with the leaf. The public key of the signer of the final (or root of the chain) TDSIGSTRUCT is also recorded.

5.5.7.3.2. Enumeration

Availability of TDG.MR.ASSIGNSVNS is enumerated by TDX_FEATURES0.TD_SIGNING_AND_SVN (bit 22), readable by TDG.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.MR.ASSIGNSVNS returns a TDX_OPERAND_INVALID(RAX) status.

5.5.7.3.3. Rate Restriction

Due to the signature verification operation, TDG.MR.ASSIGNSVNS execution time may be longer than most TDX Module interface functions execution times. To avoid long latencies where, e.g., response to hardware interrupts is delayed, the rate at which the guest TD can call this interface function is limited. If the guest TD calls this function less than about

200usec from the time it has last called TDG.MR.ASSIGNSVNS from the same VCPU, the TDX Module induces a TD exit with a TDX_TDCALL_RATE_LIMIT status. This rate limit only applies if the last TDG.MR.ASSIGNSVNS actually spent time on signature verification; it does not apply to cases such as busy resources. The host VMM can then call TDH.VP.ENTER to resume the guest operation.

5 5.5.7.4. Operands Information

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

Table 5.369: TDG.MR.ASSIGNSVNS Operands Information Definition

| Explicit/ Implicit | Reg | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|-----|-------------|-----------------|---------------|--------|---------------------|----------------|-----------------------------|
| Explicit | RCX | GPA | Input | TDSIGSTRUCT | R | Private/ Shared | 2048B | None |
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS.RTMR | SHA384_HASH | N/A | Opaque | N/A | Exclusive |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | None | Opaque | N/A | Shared(i) |

10 5.5.7.5. Completion Status Codes

Table 5.370: TDG.MR.ASSIGNSVNS Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------|--|
| TDX_OPERAND_INVALID | Alignment or access checks failed |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_SIGSTRUCT_MISMATCH | One of the reference measurements in the TDSIGSTRUCT did not match the current measurements in the TDCS. |
| TDX_SUCCESS | TDG.MR.ASSIGNSVNS is successful. |

5.5.8. TDG.MR.KEY.GET

Unreleased Feature: This section is related to Sealing, a feature which has not been released yet at the time of writing of this document. Details provided below serve as a preview and are subject to change.

TDG.MR.KEY.GET allows a TD to request a persistent key be derived for the TD, customized to the TD's measurements and policy.

5.5.8.1. Input Operands

Table 5.371: TDG.MR.KEY.GET Input Operand Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 29 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | 128B-aligned guest private physical address of TDKEYREQUEST | | |
| RDX | 32B-aligned guest private physical address for key output | | |

5.5.8.2. Output Operands

Table 5.372: TDG.MR.KEY.GET Output Operand Definition

| Operand | Description |
|---------|--|
| RAX | TDCALL instruction return code – see 5.5.1 |
| Other | Unmodified |

5.5.8.3. Leaf Function Description

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

5.5.8.3.1. Overview

TDG.MR.KEY.GET allows a TD to request a persistent key be derived for the TD, customized to the TD's measurements and policy.

5.5.8.3.2. Interaction with TD Migration

By default, using hardware-based keys by a migratable TD is not allowed. A migratable TD may allow using hardware keys by setting TDCS.TD_CTL.S.ALLOW_HW_KEYS to 1. The value of the key returned by TDG.MR.KEY.GET is different on each platform, i.e., a key request done after migration will not return the same key as a key request done prior to migration.

5.5.8.3.3. Enumeration

Availability of TDG.MR.KEY.GET is enumerated by TDX_FEATURES0.SEALING (bit 12), readable by TDG.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.MR.KEY.GET returns a TDX_OPERAND_INVALID(RAX) status.

5.5.8.4. Operands Information

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

Table 5.373: TDG.MR.KEY.GET Operands Information Definition

| Explicit/ Implicit | Reg | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|-----|-------------|-----------------|---------------|--------|---------------------|----------------|-----------------------------|
| Explicit | RCX | GPA | Input Request | TDKEYREQUEST | R | Private | 128B | None |
| Explicit | RDX | GPA | Output Key | 256-bit Key | RW | Private | 32B | None |
| Implicit | N/A | N/A | TDR page | TDR | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS.RTMR | SHA384_HASH | R | Opaque | N/A | Shared |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | None | Opaque | N/A | Shared(i) |

5

5.5.8.5. Completion Status Codes

Table 5.374: TDG.MR.KEY.GET Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------------|--|
| TDX_MR_KEY_GET_NOT_SUPPORTED | TDG.MR.KEY.GET is supported for a Migratable TDs only if it set TD_CTL.ALLOW_HW_KEYS to 1. |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDG.MR.ASSIGNSVNS is successful. |
| TDX_SVN_INVALID | SVN requested is invalid for current configuration. |

5.5.9. TDG.MR.REPORT Leaf

Unreleased Feature: Some of the text in this section is related to TD Signing, a feature which has not been released yet at the time of writing of this document. Details provided below serve as a preview and are subject to change.

5 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.

5.5.9.1. Input Operands

Table 5.375: TDG.MR.REPORT Input Operands Definition

| Operand | Description | | |
|---------|---|-----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 4 |
| | 23:16 | Version Number | Selects the TDCALL interface function version May be 0 or 1 (see enumeration details below) |
| | 63:24 | Reserved | Must be 0 |
| RCX | Guest physical address of newly created report structure. <ul style="list-style-type: none"> For version 0, the buffer must be aligned on 1024B. For version 1, the buffer must be aligned on 2048B. | | |
| 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 |
| R9 | If the version number is 1 or higher, then R9 provides the buffer size for the newly created report structure. This size must be at least 1280 bytes (see 3.9.2). If the version number is 0, R9 is ignored, and the buffer size is implicitly 1024 bytes. | | |

10

5.5.9.2. Output Operands

Table 5.376: TDG.MR.REPORT Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | TDCALL instruction return code – see 5.5.1 |
| Other | Unmodified |

5.5.9.3. Leaf Function Description

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

5.5.9.3.1. Overview

This function creates a TDREPORT_STRUCT structure that contains the measurements/configuration information of the guest TD that called the function, measurements/configuration information of the Intel TDX Module and a REPORTMACSTRUCT. The REPORTMACSTRUCT is integrity-protected with a MAC, and it contains the hash of the measurements and configuration as well as additional REPORTDATA provided by the TD software.

The created TDREPORT_STRUCT version (REPORTTYPE.VERSION) is the lowest version that contains all the TD's reported information:

- If supported, and the TD has been assigned with any SVNs or signers (i.e., any of the TDCS fields MRCONOFIGSVN, MROWNERCONFIGSVN, MRSIGROOT, MRSIGNER, ISVPROID, and ISVSVN are not 0), then the version is 2. Else,
- If any service TD has been bound or pre-bound (i.e., SERVTD_HASH is not 0), then the version is 1.
- Else, the version is 0.

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

If the TD's ATTRIBUTES.SERVTD_EXT is 1, TDREPORT.TDINFO.SERVTD_HASH will be populated with SERVTD_EXT_HASH rather than SERVTD_HASH.

Note: Although not enforced by TDG.MR.REPORT, the guest TD should normally place REPORTDATA in private memory to help ensure secure report generation.

5.5.9.3.2. Enumeration

Support of TDG.MR.REPORT version 1 is enumerated by TDX_FEATURES0.TD_SIGNING_AND_SVN (bit 22), readable by TDG.SYS.RD.

5.5.9.3.3. Interruptibility

If, during its execution, TDG.MR.REPORT detects that an external interrupt is pending, it may resume the guest TD with the CPU state unmodified. The progress so far is recorded internally. This allows the external interrupt to be recognized, causing a TD exit or a posted interrupt delivery. Typically, TDG.MR.REPORT will be re-invoked and continue its work.

Guest TD software is not directly involved. Guest TD should not precede the TDCALL with an STI instruction or a MOV to SS instruction. Posted interrupts may be delivered when the TDCALL flow is interrupted.

5.5.9.4. Operands Information

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

Table 5.377: TDG.MR.REPORT Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|-------------------|-----------------|--------|----------------------------------|--|-----------------------------|
| Explicit | RCX | GPA | Output report | TDREPORT_STRUCT | RW | Private/ Shared | Version 0: 1024B Version 1: 2048B | None |
| Explicit | RDX | GPA | Input report data | REPORTDATA | R | Private/ Shared ⁵⁰ | 64B | None |
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS.RTMR | SHA384_HASH | N/A | Opaque | N/A | Shared |

⁵⁰ Although not enforced by TDG.MR.REPORT, the guest TD should normally place REPORTDATA in private memory to help ensure secure report generation.

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|-----------------|---------------|--------|---------------------|-------------|-----------------------------|
| Implicit | N/A | N/A | TDVPS structure | TDVPS | None | Opaque | N/A | Shared(i) |

5.5.9.5. Completion Status Codes

Table 5.378: TDG.MR.REPORT Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------------------|--|
| TDX_INSUFFICIENT_TDREPORT_SPACE | TDG.MR.REPORT was provided with a buffer space that is too small for the generated TDREPORT_STRUCT |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDG.MR.REPORT is successful. |
| TDX_TDG_MR_REPORT_VERSION_MISMATCH | TDG.MR.REPORT was called with a version number that doesn't allow creating a report matching the TD's configuration |

5.5.10. TDG.MR.RTMR.EXTEND Leaf

Extend a TDCS.RTMR measurement register.

5.5.10.1. Input Operands

Table 5.379: TDG.MR.RTMR.EXTEND Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 2 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | 64B-aligned guest physical address of a 48B extension data buffer | | |
| RDX | Index of the measurement register to be extended | | |

5

5.5.10.2. Output Operands

Table 5.380: TDG.MR.RTMR.EXTEND Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | TDCALL instruction return code – see 5.5.1 |
| Other | Unmodified |

5.5.10.3. Leaf Function Description

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

TDG.MR.RTMR.EXTEND extends one of the RTMR measurement registers in TDCS with the provided extension data in memory.

5.5.10.4. Operands Information

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

Table 5.381 TDG.MR.RTMR.EXTEND Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|----------------|---------------|--------|---------------------|----------------|-----------------------------|
| Explicit | RCX | GPA | EXTEND_DATA | Blob | R | Private | 64B | None |
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS.RTMR | SHA384_HASH | N/A | Opaque | N/A | Exclusive |
| Implicit | N/A | N/A | TDVPR page | TDVPS | None | Opaque | N/A | Shared(i) |

5.5.10.5. Completion Status Codes

Table 5.382: TDG.MR.RTMR.EXTEND Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------|--|
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDG.MR.RTMR.EXTEND is successful. |

5.5.11. TDG.MR.VERIFYREPORT

Verify a cryptographic REPORTMACSTRUCT that describes the contents of a TD, to determine that it was created on the current TEE on the current platform.

5.5.11.1. Input Operands

Table 5.383: TDG.MR.VERIFYREPORT Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 22 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | 256B-aligned guest physical address of the REPORTMACSTRUCT to be verified. | | |

5.5.11.2. Output Operands

Table 5.384: TDG.MR.VERIFYREPORT Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | TDCALL instruction return code – see 5.5.1 |
| Other | Unmodified |

5.5.11.3. Leaf Function Description

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

5.5.11.3.1. Overview

TDG.MR.VERIFYREPORT computes a MAC over the provided REPORTMACSTRUCT structure; it then checks that the computed value is the same as the MAC field of that structure.

5.5.11.3.2. Enumeration

Availability of TDG.MR.VERIFYREPORT is enumerated by TDX_FEATURES0.LOCAL_ATTESTATION (bit 8), readable by TDG.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.MR.VERIFYREPORT returns a TDX_OPERAND_INVALID(RAX) status.

5.5.11.3.3. Retry on Failure

As described in the [Base Spec], there can be cases where report verification fails due to, e.g., microcode update or migration of the reporting TD and the verifying TD to another platform. In such cases it is recommended that a fresh report will be generated by the reporting TD, using TDG.MR.REPORT, and that TDG.MR.VERIFYREPORT will be called again.

5.5.11.4. Operands Information

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

Table 5.385: TDG.MR.VERIFYREPORT Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|-----------------|-----------------|--------|---------------------|----------------|-----------------------------|
| Explicit | RCX | GPA | Input report | REPORTMACSTRUCT | R | Private | 256B | None |
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS.RTMR | SHA384_HASH | N/A | Opaque | N/A | Shared |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | None | Opaque | N/A | Shared(i) |

5.5.11.5. Completion Status Codes**Table 5.386: TDG.MR.VERIFYREPORT Completion Status Codes (Returned in RAX) Definition**

| Completion Status Code | Description |
|-----------------------------|--|
| TDX_INVALID_CPUSVN | See the above note about retrying the operation. |
| TDX_INVALID_REPORTMACSTRUCT | See the above note about retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDG.MR.VERIFYREPORT is successful. |

5

5.5.12. TDG.SERVTD.RD Leaf

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

5.5.12.1. Input Operands

Table 5.387: TDG.SERVTD.RD Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 18 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | Binding handle | | |
| RDX | <p>Field identifier – see 3.10</p> <p>The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0.</p> <p>WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored.</p> <p>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.</p> | | |
| R10 | Target TD's TD_UUID bits 63:0 | | |
| R11 | Target TD's TD_UUID bits 127:64 | | |
| R12 | Target TD's TD_UUID bits 191:128 | | |
| R13 | Target TD's TD_UUID bits 255:192 | | |

5

5.5.12.2. Output Operands

Table 5.388: TDG.SERVTD.RD Output Operands Definition

| Operand | Description |
|---------|---|
| RAX | TDCALL instruction return code – see 5.5.1 |
| RDX | <p>RDX returns the next readable field identifier. A value of -1 indicates no next field identifier is available.</p> <p>If the input field identifier was -1, RDX returns the first readable field identifier. In case of another error, RDX returns -1.</p> |
| R8 | <p>Contents of the field</p> <p>In case of an error, as indicated by RAX, R8 returns 0.</p> |
| R10 | Updated target TD's TD_UUID bits 63:0 – see the description below. |
| R11 | Updated target TD's TD_UUID bits 127:64 – see the description below. |
| R12 | Updated target TD's TD_UUID bits 191:128 – see the description below. |

| Operand | Description |
|---------|---|
| R13 | Updated target TD's TD_UUID bits 255:192 – see the description below. |
| Other | Unmodified |

5.5.12.3. Leaf Function Description

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

5 5.5.12.3.1. Overview

TDG.SERVTD.RD reads a metadata field (control structure field) of a target TD.

RDX returns the next host-side readable field identifier. This may be used by the Service TD to dump the target TD metadata readable by the Service TD. To read all the available fields, the service TD can invoke TDG.SERVTD.RD in a loop, starting with field identifier -1 as an input, until RDX returns -1. A status code of TDX_METADATA_FIELD_SKIP indicates that the returned value is not applicable.

5.5.12.3.2. Enumeration

Availability of TDG.SERVTD.RD is enumerated by TDX_FEATURES0.SERVICE_TD (bit 2), readable by TDG.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.SERVTD.RD returns a TDX_OPERAND_INVALID(RAX) status.

5.5.12.3.3. TD_UUID Update

15 TD_UUID is updated when the target TD is imported. If the service TD binding to the target TD happened before the target TD was imported, the TD_UUID provided in R13:R10 may no longer be correct. In this case, if the TD_UUID provided in R13:R10 is equal to the pre-import TD_UUID of the target TD, TDG.SERVTD.RD returns TDX_TARGET_UUID_UPDATED status in RAX, and updates R13:R10 with the imported value of TD_UUID. The caller should retry the operation with the new TD_UUID.

20 5.5.12.3.4. Cross-TD Traps

Failure to access the metadata of the target TD may result in a cross-TD trap TD exit to the host VMM. This TD exit is trap like, meaning it happens after TDG.SERVTD.RD has completed its operation. On the following TDH.VP.ENTER, the host VMM may set a HOST_RECOVERABILITY_HINT flag, indicating that TDG.SERVTD.RD may be retried. From the guest TD's perspective, this flag appears in bit 60 of the status code returned in RAX. See the [TDX Module Base Spec] for details.

25 5.5.12.4. Operands Information

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

Table 5.389 TDG.SERVTD.RD Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|--|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | Target TD's TDR page (from binding handle) | TDR | R | Opaque | N/A | Shared(h) | Shared(h) | Shared(h) |
| Implicit | N/A | N/A | Service (this) TD's TDR page | TDR | None | Opaque | N/A | Shared(i) | None | None |
| Implicit | N/A | N/A | Service (this) TD's TDCS structure | TDCS | R | Opaque | N/A | Shared(i) | None | None |
| Implicit | N/A | N/A | Service (this) TD's TDCS.RTMR | SHA384_HASH | N/A | Opaque | N/A | Shared | N/A | N/A |

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|-------------------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Implicit | N/A | N/A | Service (this) TD's TDVPS structure | TDVPS | None | Opaque | N/A | Shared(i) | Shared(i) | Shared(i) |
| Implicit | N/A | N/A | Target TD's TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | None | None |
| Implicit | N/A | N/A | Target TD's TDCS.OP_STATE | OP_STATE | RW | Opaque | N/A | Shared(h) | N/A | N/A |
| Implicit | N/A | N/A | Target TD's Binding table | | R | Opaque | N/A | Shared(h) | None | None |
| Implicit | N/A | N/A | Target TD's TD metadata | N/A | R | Opaque | N/A | None | None | None |

5.5.12.5. Completion Status Codes

Table 5.390: TDG.SERVTD.RD Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|--|---|
| TDX_METADATA_FIELD_ID_INCORRECT | |
| TDX_METADATA_FIELD_NOT_READABLE | |
| TDX_METADATA_FIELD_SKIP | Indicates that the field value being read is not applicable and needs to be skipped. If called in a loop, use RDX as the identifier of the next field to be read, if any. |
| TDX_METADATA_FIRST_FIELD_ID_IN_CONTEXT | Indicates that the first field ID in context is returned |
| TDX_METADATA_FIELD_VALUE_NOT_VALID | |
| TDX_OP_STATE_INCORRECT | This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_OPERAND_ADDR_RANGE_ERROR | This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_PAGE_METADATA_INCORRECT | This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |

| Completion Status Code | Description |
|-------------------------------|--|
| TDX_SERVTD_INFO_HASH_MISMATCH | This service TD's info hash doesn't match the service TD info hash in the target TD's binding information. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_SERVTD_NOT_BOUND | This service TD is not bound to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_SERVTD_UUID_MISMATCH | This service TD's TD_UUID doesn't match the service TD UUID in the target TD's binding information. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_SUCCESS | TDG.SERVTD.RD is successful. |
| TDX_TARGET_UUID_MISMATCH | The target TD's TD_UUID value provided in R13:R10 doesn't match the actual value. |
| TDX_TARGET_UUID_UPDATED | The target TD's TD_UUID value provided in R13:R10 doesn't match the current actual value, but it does match the TD_UUID that target TD had before it was imported. In this case, the current TD_UUID value is provided in R13:R10, and the operation can be retried. |
| TDX_TD_FATAL | This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_TD_KEYS_NOT_CONFIGURED | This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_TDCS_NOT_ALLOCATED | This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |

5.5.13. TDG.SERVTD.REBIND.APPROVE

The TDG.SERVTD.REBIND.APPROVE API is called by the currently bound service TD to approve a new Service TD to be bound to the target TD.

5.5.13.1. Input Operands

Table 5.391: TDG.SERVTD.REBIND.APPROVE Input Operand Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 33 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | Old Binding Handle | | |
| RDX | Rebind Session Token [63:0] | | |
| R8 | Rebind Session Token [127:64] | | |
| R9 | Rebind Session Token [195:128] | | |
| R10 | Target TD's TD_UUID bits 63:0 | | |
| R11 | Target TD's TD_UUID bits 127:64 | | |
| R12 | Target TD's TD_UUID bits 191:128 | | |
| R13 | Target TD's TD_UUID bits 255:192 | | |
| R14 | Rebind Session Token [255:196] | | |

5.5.13.2. Output Operands

Table 5.392: TDG.SERVTD.REBIND.APPROVE Output Operand Definition

| Operand | Description |
|---------|---|
| RAX | TDCALL instruction return code – see 5.5.13.5 |
| R10 | Updated target TD's TD_UUID bits 63:0 – see the description below. |
| R11 | Updated target TD's TD_UUID bits 127:64 – see the description below. |
| R12 | Updated target TD's TD_UUID bits 191:128 – see the description below. |
| R13 | Updated target TD's TD_UUID bits 255:192 – see the description below. |
| Other | Unmodified |

5.5.13.3. Leaf Function Description

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

5.5.13.3.1. Overview

TDG.SERVTD.REBIND.APPROVE allows the calling Service TD to approve a new Service TD to be bound to the target TD currently bound to it. It sets the target TD's TDCS.REBIND_SESSION_TOKENS[slot] to the provided input rebind session token, where the slot is the current service TD's binding slot.

- 5 For Service TD rebinding details, see the [Base Spec].

5.5.13.3.2. Enumeration

Availability of TDG.SERVTD.REBIND.APPROVE is enumerated by TDX_FEATURES0.SERVTD_REBIND (bit 48), readable by TDG.SYS.RD*. If not supported, calling TDG.SERVTD.REBIND.APPROVE returns a TDX_OPERAND_INVALID(RAX) status.

5.5.13.3.3. TD_UUID Update

- 10 TD_UUID is updated when the target TD is imported. If the service TD binding to the target TD happened before the target TD was imported, the TD_UUID provided in R13:R10 may no longer be correct. In this case, if the TD_UUID provided in R13:R10 is equal to the pre-import TD_UUID of the target TD, TDG.SERVTD.REBIND.APPROVE returns TDX_TARGET_UUID_UPDATED status in RAX, and updates R13:R10 with the imported value of TD_UUID. The caller should retry the operation with the new TD_UUID.

15 5.5.13.3.4. Cross-TD Traps

Failure to access the metadata of the target TD may result in a cross-TD trap TD exit to the host VMM. This TD exit is trap like, meaning it happens after TDG.SERVTD.RD has completed its operation. On the following TDH.VP.ENTER, the host VMM may set a HOST_RECOVERABILITY_HINT flag, indicating that TDG.SERVTD.REBIND.APPROVE may be retried. From the guest TD's perspective, this flag appears in bit 60 of the status code returned in RAX. See the [TDX Module Base Spec] for details.

20 5.5.13.4. Operands Information

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

Table 5.393: TDG.SERVTD.REBIND.APPROVE Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|--|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | Target TD's TDR page (from binding handle) | TDR | RW | Opaque | N/A | Shared(h) | Shared(h) | Shared(h) |
| Implicit | N/A | N/A | Service (this) TD's TDR page | TDR | None | Opaque | N/A | Shared(i) | None | None |
| Implicit | N/A | N/A | Service (this) TD's TDCS structure | TDCS | R | Opaque | N/A | Shared(i) | None | None |
| Implicit | N/A | N/A | Service (this) TD's TDCS.RTMR | SHA384_HASH | N/A | Opaque | N/A | Shared | N/A | N/A |
| Implicit | N/A | N/A | Service (this) TD's TDVPS structure | TDVPS | None | Opaque | N/A | Shared(i) | Shared(i) | Shared(i) |
| Implicit | N/A | N/A | Target TD's TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | None | None |
| Implicit | N/A | N/A | Service (this) TD's TDCS.OP_STATE | OP_STATE | R | Opaque | N/A | Shared(h) | N/A | N/A |
| Implicit | N/A | N/A | Target TD's Binding table | | R | Opaque | N/A | Exclusive(h) | None | None |

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|----------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Implicit | N/A | N/A | Target TD's TD metadata | N/A | RW | Opaque | N/A | None | None | None |

5.5.13.5. Completion Status Codes

Table 5.394: TDG.SERVTD.REBIND.APPROVE Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|-------------------------------|--|
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDG.SERVTD.REBIND.APPROVE is successful. |
| TDX_OP_STATE_INCORRECT | This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_OPERAND_ADDR_RANGE_ERROR | This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_PAGE_METADATA_INCORRECT | This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_SERVTD_INFO_HASH_MISMATCH | This service TD's info hash doesn't match the service TD info hash in the target TD's binding information. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_SERVTD_NOT_BOUND | This service TD is not bound to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_SERVTD_UUID_MISMATCH | This service TD's TD_UUID doesn't match the service TD UUID in the target TD's binding information. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_SUCCESS | TDG.SERVTD.REBIND.APPROVE is successful. |

| Completion Status Code | Description |
|--------------------------|--|
| TDX_TARGET_UUID_MISMATCH | The target TD's TD_UUID value provided in R13:R10 doesn't match the actual value. |
| TDX_TARGET_UUID_UPDATED | The target TD's TD_UUID value provided in R13:R10 doesn't match the current actual value, but it does match the TD_UUID that target TD had before it was imported. In this case, the current TD_UUID value is provided in R13:R10, and the operation can be retried. |
| TDX_TD_FATAL | This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |

5.5.14. TDG.SERVTD.WR Leaf

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

5.5.14.1. Input Operands

Table 5.395: TDG.SERVTD.WR Input Operands Definition

| Operand | Description | | | | | | | | | | | | |
|---------|---|----------------|--|-------------|------|-------------|---|-------|----------------|--|-------|----------|-----------|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | | | | | | | | | | | |
| | <table border="1"> <thead> <tr> <th>Bits</th> <th>Field</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>15:0</td> <td>Leaf Number</td> <td>Selects the TDCALL interface function: 19</td> </tr> <tr> <td>23:16</td> <td>Version Number</td> <td>Selects the TDCALL interface function version Must be 0</td> </tr> <tr> <td>63:24</td> <td>Reserved</td> <td>Must be 0</td> </tr> </tbody> </table> | Bits | Field | Description | 15:0 | Leaf Number | Selects the TDCALL interface function: 19 | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 | 63:24 | Reserved | Must be 0 |
| | Bits | Field | Description | | | | | | | | | | |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 19 | | | | | | | | | | |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 | | | | | | | | | | |
| 63:24 | Reserved | Must be 0 | | | | | | | | | | | |
| RCX | Binding handle | | | | | | | | | | | | |
| RDX | <p>Field identifier – see 3.10</p> <p>The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0.</p> <p>WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored.</p> <p>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.</p> | | | | | | | | | | | | |
| R8 | Data to write to the field | | | | | | | | | | | | |
| R9 | A 64b write mask to indicate which bits of the value in R8 are to be written to the field | | | | | | | | | | | | |
| R10 | Target TD's TD_UUID bits 63:0 | | | | | | | | | | | | |
| R11 | Target TD's TD_UUID bits 127:64 | | | | | | | | | | | | |
| R12 | Target TD's TD_UUID bits 191:128 | | | | | | | | | | | | |
| R13 | Target TD's TD_UUID bits 255:192 | | | | | | | | | | | | |

5.5.14.2. Output Operands

Table 5.396: TDG.SERVTD.WR Output Operands Definition

| Operand | Description |
|---------|---|
| RAX | TDCALL instruction return code – see 5.5.1 |
| R8 | <p>Previous contents of the field</p> <p>In case of an error, R8 returns 0.</p> |
| R10 | Updated target TD's TD_UUID bits 63:0 – see the description below. |
| R11 | Updated target TD's TD_UUID bits 127:64 – see the description below. |
| R12 | Updated target TD's TD_UUID bits 191:128 – see the description below. |

| Operand | Description |
|---------|---|
| R13 | Updated target TD's TD_UUID bits 255:192 – see the description below. |
| Other | Unmodified |

5.5.14.3. Leaf Function Description

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

5 **5.5.14.3.1. Overview**

TDG.SERVTD.WR writes a metadata field (control structure field) of a target TD. The value (R8) is written as specified by the write mask (R9). Writing is subject to the field's internal write mask (per the TD's ATTRIBUTES.DEBUG bit). Writing of specific fields is also subject to additional rules.

Table 5.397: Metadata Field Write Rules

| Write Mask Bit in R9 | Internal Write Mask Bit | Value Bit in R8 |
|----------------------|-------------------------|---|
| 0 | N/A | Silently ignored |
| 1 | 0 | Must be the same as the current field's bit |
| 1 | 1 | Written to the current field's bit |

10

5.5.14.3.2. Enumeration

Availability of TDG.SERVTD.WR is enumerated by TDX_FEATURES0.SERVICE_TD (bit 2), readable by TDG.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.SERVTD.WR returns a TDX_OPERAND_INVALID(RAX) status.

5.5.14.3.3. TD_UUID Update

15 TD_UUID is updated when the target TD is imported. If the service TD binding to the target TD happened before the target TD was imported, the TD_UUID provided in R13:R10 may no longer be correct. In this case, if the TD_UUID provided in R13:R10 is equal to the pre-import TD_UUID of the target TD, TDG.SERVTD.WR returns TDX_TARGET_UUID_UPDATED status in RAX, and updates R13:R10 with the imported value of TD_UUID. The caller should retry the operation with the new TD_UUID.

20 **5.5.14.3.4. Cross-TD Traps**

Failure to access the metadata of the target TD may result in a cross-TD trap TD exit to the host VMM. This TD exit is trap like, meaning it happens after TDG.SERVTD.WR has completed its operation. On the following TDH.VP.ENTER, the host VMM may set a HOST_RECOVERABILITY_HINT flag, indicating that TDG.SERVTD.WR may be retried. From the guest TD's perspective, this flag appears in bit 60 of the status code returned in RAX. See the [TDX Module Base Spec] for details.

25 **5.5.14.4. Operands Information**

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

Table 5.398 TDG.SERVTD.WR Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|--|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Explicit | RCX | HPA | Target TD's TDR page (from binding handle) | TDR | RW | Opaque | N/A | Shared(h) | Shared(h) | Shared(h) |

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions | | |
|-----------------------|------|-------------|-------------------------------------|------------------|--------|---------------------|----------------|--------------------------|-----------------|-----------------|
| | | | | | | | | Operand | Contain. 2MB | Contain. 1GB |
| Implicit | N/A | N/A | Service (this) TD's TDR page | TDR | None | Opaque | N/A | Shared(i) | None | None |
| Implicit | N/A | N/A | Service (this) TD's TDCS structure | TDCS | R | Opaque | N/A | Shared(i) | None | None |
| Implicit | N/A | N/A | Service (this) TD's TDCS.RTMR | SHA384_ HASH | N/A | Opaque | N/A | Shared | N/A | N/A |
| Implicit | N/A | N/A | Service (this) TD's TDVPS structure | TDVPS | None | Opaque | N/A | Shared(i) | Shared(i) | Shared(i) |
| Implicit | N/A | N/A | Target TD's TDCS structure | TDCS | RW | Opaque | N/A | Shared(i) | None | None |
| Implicit | N/A | N/A | Service (this) TD's TDCS.OP_STATE | OP_STATE | R | Opaque | N/A | Shared(h) | N/A | N/A |
| Implicit | N/A | N/A | Target TD's Binding table | | R | Opaque | N/A | Shared(h) | None | None |
| Implicit | N/A | N/A | Target TD's TD metadata | N/A | RW | Opaque | N/A | None | None | None |

5.5.14.5. Completion Status Codes

Table 5.399: TDG.SERVTD.WR Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------------------|---|
| TDX_METADATA_FIELD_ID_INCORRECT | |
| TDX_METADATA_FIELD_NOT_READABLE | |
| TDX_METADATA_FIELD_NOT_WRITABLE | |
| TDX_METADATA_FIELD_VALUE_NOT_VALID | |
| TDX_METADATA_WR_MASK_NOT_VALID | |
| TDX_OP_STATE_INCORRECT | This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_OPERAND_ADDR_RANGE_ERROR | This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |

| Completion Status Code | Description |
|-------------------------------|--|
| TDX_PAGE_METADATA_INCORRECT | This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_SERVTD_INFO_HASH_MISMATCH | This service TD's info hash doesn't match the service TD info hash in the target TD's binding information. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_SERVTD_NOT_BOUND | This service TD is not bound to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_SERVTD_UUID_MISMATCH | This service TD's TD_UUID doesn't match the service TD UUID in the target TD's binding information. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_SUCCESS | TDG.SERVTD.WR is successful. |
| TDX_TARGET_UUID_MISMATCH | The target TD's TD_UUID value provided in R13:R10 doesn't match the actual value. |
| TDX_TARGET_UUID_UPDATED | The target TD's TD_UUID value provided in R13:R10 doesn't match the current actual value, but it does match the TD_UUID that target TD had before it was imported. In this case, the current TD_UUID value is provided in R13:R10, and the operation can be retried. |
| TDX_TD_FATAL | This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_TD_KEYS_NOT_CONFIGURED | This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |
| TDX_TDCS_NOT_ALLOCATED | This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved, and the service TD can retry the operation. |

5.5.15. TDG.SYS.RD Leaf

Read a TDX Module global-scope metadata field.

5.5.15.1. Input Operands

Table 5.400: TDG.SYS.RD Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 11 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RDX | <p>Field identifier – see 3.10</p> <p>The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0.</p> <p>WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored.</p> <p>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.</p> | | |

5

5.5.15.2. Output Operand

Table 5.401: TDG.SYS.RD Output Operands Definition

| Operand | Description |
|---------|---|
| RAX | TDCALL instruction return code – see 5.5.1 |
| RDX | If the input field identifier was -1, RDX returns the first readable field identifier. Else, in case of an error, RDX returns -1. On success, RDX returns the next readable field identifier. A value of -1 indicates no next field identifier is available. |
| R8 | Contents of the field In case of no success, as indicated by RAX, R8 returns 0. |
| Other | Unmodified |

5.5.15.3. Leaf Function Description

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

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

RDX returns the next guest-side readable field identifier. This may be used by the guest TD to enumerate the TDX Module's capabilities and configuration. To read all the available fields, the guest TD can invoke TDG.SYS.RD in a loop, starting with field identifier -1 as an input, until RDX returns -1. A status code of TDX_METADATA_FIELD_SKIP indicates that the returned value is not applicable. Alternatively, the guest TD can use TDG.SYS.RDALL.

15

5.5.15.4. Operands Information

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

Table 5.402 TDG.SYS.RD Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|-----------------|---------------|--------|---------------------|----------------|-----------------------------|
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | None | Opaque | N/A | Shared(i) |

5

5.5.15.5. Completion Status Codes

Table 5.403: TDG.SYS.RD Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|--|---|
| TDX_METADATA_FIELD_ID_INCORRECT | |
| TDX_METADATA_FIELD_NOT_READABLE | |
| TDX_METADATA_FIELD_SKIP | Indicates that the field value being read is not applicable and needs to be skipped. If called in a loop, use RDX as the identifier of the next field to be read, if any. |
| TDX_METADATA_FIRST_FIELD_ID_IN_CONTEXT | Indicates that the first field ID in context is returned |
| TDX_METADATA_FIELD_SKIP | Indicates that the field value being read is not applicable and needs to be skipped. If called in a loop, use RDX as the identifier of the next field to be read, if any. |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDG.SYS.RD is successful. |

5.5.16. TDG.SYS.RDALL Leaf

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

5.5.16.1. Input Operands

Table 5.404: TDG.SYS.RDALL Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 12 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RDX | The GPA of a 4KB buffer where a metadata list will be returned The buffer must be aligned on 4KB. In case of error, some field value entries might not contain valid data. | | |
| R8 | Initial field identifier – see 3.10 If R8's value is -1, then TDG.SYS.RDALL will start from the first global-scope metadata field identifier. Else, LAST_ELEMENT_IN_FIELD, LAST_FIELD_IN_SEQUENCE, WRITE_MASK_VALID and CONTEXT_CODE fields are ignored. The FIELD_CODE must be the code of the first element of a metadata field. | | |

5

5.5.16.2. Output Operands

Table 5.405: TDG.SYS.RDALL Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | TDCALL instruction return code – see 5.5.1 |
| R8 | Next field identifier. A value of -1 means all applicable field identifiers have been returned in the metadata list. In case of an error, as indicated by RAX, R8 returns -1. |
| Other | Unmodified |

5.5.16.3. Leaf Function Description

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

TDG.SYS.RDALL reads all host-readable TDX Module global-scope metadata fields into a metadata list in the provided page.

If one or more applicable fields do not fit in the provided list buffer, the function can be invoked in a loop, each invocation providing an initial field identifier returned as the next field identifier of the previous invocation, as shown in the following example:

1. NEXT_FIELD_ID = -1

2. Repeat:
 - 2.1. Set LIST_BUFFER to the next 4K buffer
 - 2.2. Invoke TDG.SYS.RDALL(RDX = LIST_BUFFER, RDX = NEXT_FIELD_ID)
 - 2.3. STATUS = RAX, NEXT_FIELD_ID = R8
5. Until ((STATUS is a non-recoverable error) or (NEXT_FIELD_ID is -1))

The function never returns an empty list if there's no error.

5.5.16.4. Operands Information

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

10 **Table 5.406: TDG.SYS.RDALL Operands Information Definition**

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|-----------------|---------------|--------|---------------------|----------------|-----------------------------|
| Explicit | RDX | GPA | Metadata List | MD_LIST | RW | Private | 4096 | None |
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | None | Opaque | N/A | Shared(i) |

5.5.16.5. Completion Status Codes

Table 5.407: TDG.SYS.RDALL Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------|--|
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDG.SYS.RDALL is successful. |

5.5.17. TDG.VM.RD Leaf

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

5.5.17.1. Input Operands

Table 5.408: TDG.VM.RD Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|---|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 7 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Version number may be 0 or 1. See the enumeration details below. |
| | 63:24 | Reserved | Must be 0 |
| RCX | Reserved, must be 0 | | |
| RDX | <p>Field identifier – see 3.10</p> <p>The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0.</p> <p>WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored.</p> <p>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.</p> | | |

5

5.5.17.2. Output Operands

Table 5.409: TDG.VM.RD Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | TDCALL instruction return code – see 5.5.1 |
| RDX | <p>For TDG.VM.RD version 0, RDX is unmodified.</p> <p>For TDG.VM.RD version 1 or higher:</p> <ul style="list-style-type: none"> If the input field identifier was -1, RDX returns the first readable field identifier. Else, in case of an error, RDX returns -1. On success, RDX returns the next readable field identifier. A value of -1 indicates no next field identifier is available. |
| R8 | <p>Contents of the field</p> <p>In case of no success, as indicated by RAX, R8 returns 0.</p> |
| Other | Unmodified |

5.5.17.3. Leaf Function Description

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

5.5.17.3.1. Overview

TDG.VM.RD reads a VM-scope metadata field (control structure field) of a TD.

If version 1 or higher is specified in RAX, RDX returns the next host-side readable field identifier. This may be used by the guest TD to dump the guest readable TD metadata. To read all the available fields, the guest TD can invoke TDG.VM.RD in a loop, starting with field identifier -1 as an input, until RDX returns -1. A status code of TDX_METADATA_FIELD_SKIP indicates that the returned value is not applicable.

5 5.5.17.3.2. Enumeration

Availability of TDG.VM.RD version 1 is enumerated by TDX_FEATURES0.ENHANCED_METADATA (bit 3), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.VM.RD with a version number higher than 0 returns a TDX_OPERAND_INVALID(RAX) status.

5.5.17.4. Operands Information

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

Table 5.410 TDG.VM.RD Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|---------------------------------|------------------|--------|---------------------|----------------|-----------------------------|
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TD metadata (guest-side access) | N/A | R | Opaque | N/A | Shared |

5.5.17.5. Completion Status Codes

15 **Table 5.411: TDG.VM.RD Completion Status Codes (Returned in RAX) Definition**

| Completion Status Code | Description |
|--|---|
| TDX_METADATA_FIELD_ID_INCORRECT | |
| TDX_METADATA_FIELD_NOT_READABLE | |
| TDX_METADATA_FIELD_SKIP | Indicates that the field value being read is not applicable and needs to be skipped. If called in a loop, use RDX as the identifier of the next field to be read, if any. |
| TDX_METADATA_FIRST_FIELD_ID_IN_CONTEXT | Indicates that the first field ID in context is returned |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDG.VM.RD is successful. |

5.5.18. TDG.VM.WR Leaf

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

5.5.18.1. Input Operands**Table 5.412: TDG.VM.WR Input Operands Definition**

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 8 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| 63:24 | Reserved | Must be 0 | |
| RCX | Reserved, must be 0 | | |
| RDX | Field identifier – see 3.10 The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0. WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored. | | |
| R8 | Data to write to the field | | |
| R9 | A 64b write mask to indicate which bits of the value in R8 are to be written to the field | | |

5

5.5.18.2. Output Operands**Table 5.413: TDG.VM.WR Output Operands Definition**

| Operand | Description |
|---------|---|
| RAX | TDCALL instruction return code – see 5.5.1 |
| R8 | Previous contents of the field In case of an error, as indicated by RAX, R8 returns 0. |
| Other | Unmodified |

5.5.18.3. Leaf Function Description

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

TDG.VM.WR writes a VM-scope metadata field (control structure field) of a TD. The value (R8) is written as specified by the write mask (R9). Writing is subject to the field's internal write mask (per the TD's ATTRIBUTES.DEBUG bit). Writing of specific fields is also subject to additional rules.

15

Table 5.414: Metadata Field Write Rules

| Write Mask Bit in R9 | Internal Write Mask Bit | Value Bit in R8 |
|----------------------|-------------------------|------------------|
| 0 | N/A | Silently ignored |

| Write Mask Bit in R9 | Internal Write Mask Bit | Value Bit in R8 |
|----------------------|-------------------------|---|
| 1 | 0 | Must be the same as the current field's bit |
| 1 | 1 | Written to the current field's bit |

5.5.18.4. Operands Information

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

5

Table 5.415 TDG.VM.WR Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|---------------------------------|------------------|--------|---------------------|----------------|-----------------------------|
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared |
| Implicit | N/A | N/A | TDCS structure | TDCS | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TD metadata (guest-side access) | N/A | R | Opaque | N/A | Shared |

5.5.18.5. Completion Status Codes

Table 5.416: TDG.VM.WR Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------------------|--|
| TDX_METADATA_FIELD_ID_INCORRECT | |
| TDX_METADATA_FIELD_NOT_READABLE | |
| TDX_METADATA_FIELD_NOT_WRITABLE | |
| TDX_METADATA_FIELD_VALUE_NOT_VALID | |
| TDX_METADATA_WR_MASK_NOT_VALID | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDG.VM.WR is successful. |

5.5.19. 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.

5.5.19.1. Input Operands

Table 5.417: TDG.VP.CPUIDVE.SET Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 5 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | Controls whether CPUID executed by the guest TD will cause #VE(CONFIG_PARAVIRT) unconditionally | | |
| | Bits | Name | Description |
| | 0 | SUPERVISOR | Flags that when CPL is 0, a CPUID executed by the guest TD will cause a #VE(CONFIG_PARAVIRT) unconditionally |
| | 1 | USER | Flags that when CPL > 0, a CPUID executed by the guest TD will cause a #VE(CONFIG_PARAVIRT) unconditionally |
| | 63:2 | RESERVED | Reserved: must be 0 |

5.5.19.2. Output Operands

Table 5.418: TDG.VP.CPUIDVE.SET Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | TDCALL instruction return code – see 5.5.1 |
| Other | Unmodified |

5.5.19.3. Leaf Function Description

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

This function controls whether execution of CPUID by the guest TD, when running in supervisor mode and/or in user mode, will unconditionally result in a #VE(CONFIG_PARAVIRT).

5.5.19.4. Operands Information

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

Table 5.419 TDG.VP.CPUIIDVE.SET Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|-----------------|---------------|--------|---------------------|----------------|-----------------------------|
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | RW | Opaque | N/A | Shared(i) |

5.5.19.5. Completion Status Codes**Table 5.420: TDG.VP.CPUIIDVE.SET Completion Status Codes (Returned in RAX) Definition**

| Completion Status Code | Description |
|------------------------|------------------------------------|
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDG.VP.CPUIIDVE.SET is successful. |

5

5.5.20. TDG.VP.ENTER Leaf

Enter L2 VCPU operation.

From the L1 VMM’s point of view, TDG.VP.ENTER is a complex operation that normally involves L1→L2 VM entry and L2→L1 VM exit; however, it may fail before L2 VM entry and may also involve TD exits and entries. Therefore, output operands are specified by multiple tables below.

5.5.20.1. Input Operands

Table 5.421: TDG.VP.ENTER Input Operands Definition

| Operand | Name | Description | | |
|---------|------------------|--|-------------------|---|
| RAX | Leaf and Version | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the TDCALL interface function: 25 |
| | | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | | 63:24 | Reserved | Must be 0 |
| RCX | VM_FLAGS | VM identifier and flags | | |
| | | Bits | Name | Description |
| | | 1:0 | INVD_TRANSLATIONS | Controls how TDG.VP.ENTER flushes the TLB context and extended paging structure (EPxE) caches associated with the L2 VM before entering the L2 VCPU |
| | | 51:2 | Reserved | Reserved: must be 0 |
| | | 53:52 | VM | L2 virtual machine index (must be 1 or higher) |
| | | 63:54 | Reserved | Reserved: must be 0 |
| RDX | GUEST_STATE_GPA | The GPA of a 256-bytes aligned L2_ENTER_GUEST_STATE structure - see 3.8.1 for details. | | |

Table 5.422: INVD_TRANSLATION Definition

| Value | Address Translation Invalidation | Underlying Mechanism | Comments |
|-------|--|--|--------------------------------|
| 0 | No invalidation | None | |
| 1 | Invalidate all TLB entries and extended paging-structure translations (EPxE) associated with the L2 VM being entered | INVEPT single-context invalidation (type 1) | |
| 2 | Invalidate all TLB entries associated with the L2 VM being entered | INVPID single-context invalidation (type 1) | See enumeration details below. |
| 3 | Invalidate TLB entries associated with the L2 VM being entered, excluding global translations | INVPID single-context invalidation, retaining global translations (type 3) | See enumeration details below. |

5.5.20.2. Output Operands

TDG.VP.ENTER output format depends on how the function was terminated.

5.5.20.2.1. Output Operands Format on No L1→L2 Entry

The following table details TDG.VP.ENTER output operands when the interface function returns **without entering the L2 VCPU** due to an error or some other condition.

Table 5.423: TDG.VP.ENTER Output Operands Definition on No L2 VM Entry

| Operand | Name | Description | | |
|---------|--------|----------------------------------|--|--|
| RAX | Status | SEAMCALL instruction return code | | |
| | | Bit(s) | Name | Description |
| | | 47:32 | CLASS and DETAILS_L1 | May have the following values: <ul style="list-style-type: none"> • TDX_PENDING_INTERRUPT, indicating that an interrupt is pending for L1 • Any other value not in the table below |
| | Other | | See the function completion status definition in 5.5.1 | |
| Other | | Unmodified | | |

5.5.20.2.2. Output Operands Format on Successful L1→L2 Entry Followed by L2→L1 Exit

The following table details TDG.VP.ENTER output operands when L2 VM entry succeeds, and later an L2 VM exit occurs due to a **VMX architectural exit reason**.

Table 5.424: TDG.VP.ENTER Output Operands Definition on an L2→L1 Exits Following an L1→L2 Entry

| Operand | Name | Description | | |
|---------|----------------------|-----------------------------------|--|--|
| RAX | Status | SEAMCALL instruction return code | | |
| | | Bit(s) | Name | Description |
| | | 31:0 | DETAILS_L2: Exit Reason | L2 VMCS exit reason |
| | | 47:32 | CLASS and DETAILS_L1 | May have the following values: <ul style="list-style-type: none"> • TDX_SUCCESS, indicating a normal L2→L1 exit • TDX_L2_EXIT_PENDING_INTERRUPT*, indicating an L2→L1 exit due to an interrupt posted to L1 or L2 • TDX_L2_EXIT_HOST_ROUTED_*, indicating a TD exit from L2 where the host VMM requested resumption of L1 Other values in the range 0x1100 through 0x111F are reserved for future additional status codes that indicate an L2→L1 exit following an L1→L2 entry. |
| | Other | | See the function completion status definition in 5.5.1 | |
| RCX | Exit Qualification | exit qualification from L2 VMCS | | |
| RDX | Guest Linear Address | guest-linear address from L2 VMCS | | |
| RSI | CS Info | CS selector, AR and limit | | |
| | | Bits | Details | |

| Operand | Name | Description | | | |
|---------|--|--|----------------------------------|-----------------------|--|
| | | 15:0 | CS Selector | | |
| | | 31:16 | CS AR bit 15:0 | | |
| | | 63:32 | CS Limit | | |
| RDI | CS Base | CS base address | | | |
| R8 | Guest Physical Address | guest-physical address from L2 VMCS | | | |
| R9 | VM-Exit Interruption Information | The following information is provided for L2 VM exits due to vectored events. In other cases, R9 content should be ignored. | | | |
| | | Bits | Details | | |
| | | 31:0 | VM-Exit Interruption Information | | |
| | | 63:32 | VM-Exit Interruption Error Code | | |
| R10 | IDT-Vectoring Information | The following information is provided for L2 VM exits that occur during event delivery. In other cases, R10 content should be ignored. | | | |
| | | Bits | Details | | |
| | | 31:0 | IDT-Vectoring Information | | |
| | | 63:32 | IDT-Vectoring Error Code | | |
| R11 | VM-Exit Instruction Information | The following information is provided for L2 VM exits due to instruction execution. In other cases, R11 content should be ignored. | | | |
| | | Bits | Details | | |
| | | 31:0 | VM-Exit Instruction Information | | |
| R12 | Additional Exit Information | Additional exit information | | | |
| | | Bits | Details | | |
| | | 1:0 | CPL (from GuestSS.AR.DPL) | | |
| R13 | Extended Exit Qualification | Extended exit qualification | | | |
| | | Bits | Details | | |
| | | 3:0 | Extended exit qualification type | | |
| | | | Value | Name | Description |
| | | | 0 | NONE | No extended exit qualification |
| | | | 6 | PENDING_EPT_VIOLATION | Extended exit qualification for an EPT violation due to L2 VM access to a PENDING page |
| Other | Reserved | | | | |
| 63:4 | Reserved, set to 0 | | | | |
| R14 | VM-Exit Extended Instruction Information | If APX is virtualized as supported (virtual CPUID(7,1).EDX[21] is 1), then R14 returns the L2 VMCS' VM-Exit Extended Instruction Information field. Else, R14 returns 0. | | | |

| Operand | Name | Description |
|-------------|---|--------------|
| RBX, R15 | None | Cleared to 0 |
| Other state | Any state that the L2 VM is allowed to use may be modified. | |

5.5.20.3. Leaf Function Description

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

5 **5.5.20.3.1. Overview**

TDG.VP.ENTER transitions the VCPU into L2 VM operation. The function returns either if failed to enter L2 VM or after successful entry to L2 VM and then exit from L2 VM.

5.5.20.3.2. Enumeration

10 Availability of TDG.VP.ENTER is enumerated by TDX_FEATURES0.TD_PARTITIONING (bit 7), readable by TDG.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.VP.ENTER returns a TDX_OPERAND_INVALID(RAX) status.

Available INVD_TRANSLATION values are enumerated by TDX_FEATURES0.L2_TLB_INVD_OPT (bit 19).

Availability of PENDING_EPT_VIOLATION indication in R13 is enumerated by TDX_FEATURES.PENDING_EPT_VIOLATION_V2 (bit 16).

5.5.20.3.3. CPU State Preservation Following a Successful L1→L2 VM Entry and an L2→L1 VM Exit

15 Following a successful L1→L2 VM entry and an L2→L1 VM exit, some CPU state is modified:

- General purpose register (GPR) values are not preserved.
- Any state that the L2 VM is allowed to use may be modified.

5.5.20.3.4. Non-Error Termination Conditions

The following table lists non-error TDG.VP.ENTER termination conditions:

20 **Table 5.425: TDG.VP.ENTER Termination Cases**

| Case | Status in RAX[63:32] | Description |
|-------------------------------|-------------------------------|---|
| Normal | TDX_SUCCESS | L1→L2 entry was successful, followed by an L2→L1 exit. L2 VM exit information is provided in output GPRs. |
| Host Requested L2 Exit | TDX_L2_EXIT_HOST_ROUTED_ASYNC | L1→L2 entry was successful. Later, following direct TD exit from L2, the host VMM requested resumption of L1. L2 CPU state is updated in the register list. L2 VM exit information is provided in output GPRs. This information was provided to the host VMM on TD exit; it may or may not be meaningful to the L1 VMM. In case of TDG.VP.VMCALL, the L2 CPU state is the state after completion of that function, e.g., GPR values are as returned by the host VMM as inputs to TDH.VP.ENTER. This condition is sticky. I.e., if resumption of L1 encountered a problem that required a TD exit (e.g., an EPT violation) the following TD entry resumes L1 and provides the same TDX_L2_EXIT_HOST_ROUTED status. |

| Case | Status in RAX[63:32] | Description |
|---|--|--|
| Host Requested L2 Exit following TDG.VP.VMCALL | TDX_L2_EXIT_HOST_ROUTED_TDVMCALL | <p>L1→L2 entry was successful. Later, following TDG.VP.VMCALL that caused a direct TD exit from L2, the host VMM requested resumption of L1.</p> <p>L2 CPU state is updated in the L2_ENTER_GUEST_STATE structure. The L2 CPU state is the state after completion of TDG.VP.VMCALL, e.g., GPR values are as returned by the host VMM as inputs to TDH.VP.ENTER.</p> <p>L2 VM exit information is provided in output GPRs. This information was provided to the host VMM on the last TD exit; it may or may not be meaningful to the L1 VMM.</p> <p>This condition is sticky. I.e., if resumption of L1 encountered a problem that required a TD exit (e.g., an EPT violation) the following TD entry resumes L1 and provides the same TDX_L2_EXIT_HOST_ROUTED_TDVMCALL status. Note that in such a case the L2 VM exit information reflects, e.g., the EPT violation, not the original TDG.VP.VMCALL exit.</p> |
| Pending Interrupt L2 Exit | TDX_L2_EXIT_PENDING_INTERRUPT | <p>L1→L2 entry was successful. Later, an L2→L1 exit happened due to an interrupt that was posted to L1. L2 CPU state is updated in the register list. L2 VM exit information is provided in output GPRs but is not necessarily meaningful to the L1 VMM (e.g., VM-exit interruption information is for the external interrupt that triggered the L2→L1 exit).</p> |
| Pending Interrupt L2 Exit | TDX_L2_EXIT_PENDING_INTERRUPT_TDVMCALL | <p>L1→L2 entry was successful. Later, following TDG.VP.VMCALL that caused a direct TD exit from L2, the host VMM resumed the TD. During TD resumption, a posted interrupt to either L1 or any L2 was detected by the TDX Module.</p> <p>L2 CPU state is updated in the L2_ENTER_GUEST_STATE structure. The L2 CPU state is the state after completion of TDG.VP.VMCALL, e.g., GPR values are as returned by the host VMM as inputs to TDH.VP.ENTER.</p> <p>L2 VM exit information is provided in output GPRs. This information was provided to the host VMM on the last TD exit; it may or may not be meaningful to the L1 VMM.</p> <p>This condition is sticky. I.e., if resumption of L1 encountered a problem that required a TD exit (e.g., an EPT violation) the following TD entry resumes L1 and provides the same TDX_L2_EXIT_PENDING_INTERRUPT_TDVMCALL status. Note that in such a case the L2 VM exit information reflects, e.g., the EPT violation, not the original TDG.VP.VMCALL exit.</p> |
| No L2 Entry | Other | <p>L1→L2 entry was aborted because of some condition, which may or may not be an error, as indicated by RAX bit 63.</p> <p>E.g., entry may have been aborted due to a pending virtual interrupt. In this case, the L1 VMM typically sets RFLAGS.IF to 1, handles the interrupt and then invokes TDG.VP.ENTER again.</p> |

5.5.20.4. Operands Information

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

Table 5.426: TDG.VP.ENTER Memory Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|-----------------|----------------------|--------|---------------------|----------------|-----------------------------|
| Explicit | RDX | GPA | Guest state | L2_ENTER_GUEST_STATE | RW | Private | 256B | None |
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | None | Opaque | N/A | Shared(i) |

5.5.20.5. Completion Status Codes**Table 5.427: TDG.VP.ENTER Completion Status Codes (Returned in RAX) Definition**

| Completion Status Code | Description |
|----------------------------------|--|
| TDX_L2_EXIT_HOST_ROUTED_ASYNC | L1→L2 entry succeeded. Later, following an asynchronous TD exit from L2, the host VMM requested resumption of L1. |
| TDX_L2_EXIT_HOST_ROUTED_TDVMCALL | L1→L2 entry succeeded. Later, following a TDG.VP.VMCALL TD exit from L2, the host VMM requested resumption of L1. |
| TDX_L2_EXIT_PENDING_INTERRUPT | L1→L2 entry succeeded, and later L2→L1 exit happened due to an interrupt that was posted to L1 and is pending. |
| TDX_OPERAND_INVALID | |
| TDX_PENDING_INTERRUPT | L1→L2 entry was aborted because an interrupt is pending for the L1 VMM. This indication is returned even if the L1 VMMs cleared RFLAGS.IF. |
| TDX_SUCCESS | |

5

5.5.21. TDG.VP.INFO Leaf

Get guest TD execution environment information.

5.5.21.1. Input Operands

Table 5.428: TDG.VP.INFO Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 1 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |

5

5.5.21.2. Output Operands

Table 5.429: TDG.VP.INFO Output Operands Definition

| Operand | Description | | |
|---------|--|------------|---|
| RAX | TDCALL instruction return code – see 5.5.1 – returns a constant value of TDX_SUCCESS (0) | | |
| RCX | Bits | Name | Description |
| | 5:0 | GPAW | The effective GPA width (in bits) for this TD (do not confuse with MAXPA). SHARED bit is at GPA bit GPAW-1. Only GPAW values 48 and 52 are possible. |
| | 63:6 | RESERVED | Reserved: 0 |
| RDX | The TD's ATTRIBUTES (provided as input to TDH.MNG.INIT) | | |
| R8 | Bits | Name | Description |
| | 31:0 | NUM_VCPUS | Number of Virtual CPUs that are usable (i.e., either active or ready) |
| | 63:32 | MAX_VCPUS | TD's maximum number of Virtual CPUs (provided as input to TDH.MNG.INIT) |
| R9 | Bits | Name | Description |
| | 31:0 | VCPU_INDEX | Virtual CPU index, starting from 0 and allocated sequentially on each successful TDH.VP.INIT |
| | 63:32 | RESERVED | Reserved for enumerating future Intel TDX Module capabilities, etc.: set to 0 |
| R10 | Bits | Name | Description |
| | 0 | SYS_RD | Indicates that the TDG.SYS.RD/RDM/RDALL functions are available. Further enumeration can be done using these functions. |
| | 63:1 | RESERVED | Reserved – set to 0 |
| R11 | Reserved for enumerating future Intel TDX Module capabilities, etc.: set to 0 | | |

| Operand | Description |
|---------|-------------|
| Other | Unmodified |

5.5.21.3. Leaf Function Description

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

- 5 TDG.VP.INFO provides the TD software with execution environment information – beyond information that is provided by CPUID.

5.5.21.4. Operands Information

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

10

Table 5.430: TDG.VP.INFO Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|-----------------|---------------|--------|---------------------|----------------|-----------------------------|
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | R | Opaque | N/A | Shared(i) |

5.5.21.5. Completion Status Codes

Table 5.431: TDG.VP.INFO Completion Status Codes (Returned in RAX) Definition

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

5.5.22. TDG.VP.INVEPT Leaf

Invalidate cached EPT translations for selected L2 VMs.

5.5.22.1. Input Operands

Table 5.432: TDG.VP.INVEPT Input Operands Definition

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 26 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | VM index bitmap Bit N value of 1 indicates a request to invalidate EPT for VM index N. N must be between 1 and the number of L2 VMs in this TD. | | |
| | Bits | Name | Description |
| | 0 | Reserved | Reserved: must be 0 |
| | 1 | L2_VM_1 | Invalidate EPT for L2 VM #1 |
| | 2 | L2_VM_2 | Invalidate EPT for L2 VM #2 |
| | 3 | L2_VM_3 | Invalidate EPT for L2 VM #3 |
| | 63:4 | Reserved | Reserved: must be 0 |

5

5.5.22.2. Output Operands

Table 5.433: TDG.VP.INVEPT Output Operands Definition

| Operand | Description |
|---------|--------------------------------|
| RAX | TDCALL instruction return code |
| Other | Unmodified |

5.5.22.3. Leaf Function Description

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

5.5.22.3.1. Overview

TDG.VP.INVEPT executes INVEPT to invalidate the EPT translations of the specified L2 VMs.

5.5.22.3.2. Enumeration

15 Availability of TDG.VP.INVEPT is enumerated by TDX_FEATURES0.TD_PARTITIONING (bit 7), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.VP.INVEPT returns a TDX_OPERAND_INVALID(RAX) status.

5.5.22.4. Operands Information

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

Table 5.434 TDG.VP.RD Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|-----------------|---------------|--------|---------------------|----------------|-----------------------------|
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | R | Opaque | N/A | Shared(i) |

5

5.5.22.5. Completion Status Codes

Table 5.435: TDG.VP.INVEPT Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------|------------------------------|
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDG.VP.INVEPT is successful. |

5.5.23. TDG.VP.INVGLA Leaf

Invalidate Guest Linear Address (GLA) mappings in the translation lookaside buffers (TLBs) and paging-structure caches for a specified L2 VM and a specified list of 4KB-aligned linear addresses.

5.5.23.1. Input Operands

5 **Table 5.436: TDG.VP.INVGLA Input Operands Definition**

| Operand | Name | Description | | |
|---------|---------------------------------|--|----------------|---|
| RAX | Leaf and Version | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | | Bits | Field | Description |
| | | 15:0 | Leaf Number | Selects the TDCALL interface function: 27 |
| | | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | | 63:24 | Reserved | Must be 0 |
| RCX | VM_AND_FLAGS | VM identifier and flags | | |
| | | Bits | Name | Description |
| | | 0 | LIST | 0: RDX contains a single GLA list entry 1: RDX contains the GPA and other information of a GLA list in memory. |
| | | 51:1 | Reserved | Reserved: must be 0 |
| | | 53:52 | VM | L2 virtual machine index (must be 1 or higher) |
| | | 63:54 | Reserved | Reserved: must be 0 |
| RDX | GLA_LIST_ENTRY or GLA_LIST_INFO | Depending on the LIST flag in RCX, RDX contains either of the following: <ul style="list-style-type: none"> A single GLA_LIST_ENTRY, specifying up to 512 consecutive guest linear addresses, each aligned on 4KB. GLA_LIST_INFO, specifying the GPA of a guest linear address (GLA) list in private memory. Each entry in the GLA list specifies up to 512 consecutive guest linear addresses, each aligned on 4KB. GLA_LIST_INFO also specifies the first and last GLA list entries to process. See 3.6.4 for details. | | |

5.5.23.2. Output Operands

Table 5.437: TDG.VP.INVGLA Output Operands Definition

| Operand | Name | Description |
|---------|---------------------------------|--|
| RAX | Status | TDCALL instruction return code |
| RDX | GLA_LIST_ENTRY or GLA_LIST_INFO | Depending on the LIST flag provided as input in RCX, RDX contains either of the following: <ul style="list-style-type: none"> If LIST was 0, RDX contains the single GLA_LIST_ENTRY provided as an input, unmodified. |

| Operand | Description |
|---------|---|
| | <ul style="list-style-type: none"> If LIST was 1, RDX contains the GLA_LIST_INFO provided as input, but with the FIRST_ENTRY and NUM_ENTRIES fields updated to reflect the number of entries processed so far. If all entries have been processed successfully, NUM_ENTRIES is set to 0. |
| Other | Unmodified |

5.5.23.3. Leaf Function Description

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

5 **5.5.23.3.1. Overview**

TDG.VP.INVGLA executes INVVPID type 0 to invalidate the cached translations for the specified list of 4KB page Guest Linear Addresses (GLA) of the specified L2 VM.

5.5.23.3.2. Enumeration

10 Availability of TDG.VP.INVGLA is enumerated by TDX_FEATURES0.TD_PARTITIONING (bit 7), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.VP.INVGLA returns a TDX_OPERAND_INVALID(RAX) status.

5.5.23.3.3. Interruptibility

If, during its execution, TDG.VP.INVGLA detects that an external interrupt is pending, it may resume the guest TD with the CPU state unmodified, except for the following:

- GLA_LIST_INFO in RDX is updated to reflect the GLAs processed so far.

15 This allows the external interrupt to be recognized, causing a VM exit or a posted interrupt delivery. Typically, TDG.VP.INVGLA will be re-invoked (since RIP has not changed) and continue its work. Guest TD software is not directly involved.

Guest TD software is not directly involved. Guest TD should not precede the TDCALL with an STI instruction or a MOV to SS instruction. Posted interrupts may be delivered when the TDCALL flow is interrupted.

20 **5.5.23.4. Operands Information**

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

Table 5.438: TDG.VP.INVGLA Operands Information Definition

| Explicit/Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-------------------|------|----------|-----------------|---------------|--------|------------------|-------------|--------------------------|
| Explicit | RDX | GPA | GLA list page | GLA_LIST | R | Private | 4096 | None |
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | R | Opaque | N/A | Shared(i) |

25 **5.5.23.5. Completion Status Codes**

Table 5.439: TDG.VP.INVGLA Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------|-------------|
| TDX_OPERAND_INVALID | |

| Completion Status Code | Description |
|------------------------|------------------------------|
| TDX_SUCCESS | TDG.VP.INVGLA is successful. |

5.5.24. TDG.VP.RD Leaf

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

5.5.24.1. Input Operands

Table 5.440: TDG.VP.RD Input Operands

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 9 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| 63:24 | Reserved | Must be 0 | |
| RCX | Reserved, must be 0 | | |
| RDX | <p>Field identifier – see 3.10</p> <p>The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0.</p> <p>WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored.</p> <p>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.</p> | | |

5

5.5.24.2. Output Operands

Table 5.441: TDG.VP.RD Output Operands Definition

| Operand | Description |
|---------|--|
| RAX | TDCALL instruction return code – see 5.5.1 |
| RDX | If the input field identifier was -1, RDX returns the first readable field identifier. Else, in case of an error, RDX returns -1. On success, RDX returns the next readable field identifier. A value of -1 indicates no next field identifier is available. |
| R8 | Contents of the field In case of no success, as indicated by RAX, R8 returns 0. |
| Other | Unmodified |

5.5.24.3. Leaf Function Description

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

5.5.24.3.1. Overview

TDG.VP.RD reads a VCPU-scope metadata field (control structure field) of a TD.

15 RDX returns the next host-side readable field identifier. This may be used by the guest TD to dump the guest readable VCPU metadata. To read all the available fields, the guest TD can invoke TDG.VP.RD in a loop, starting with field identifier

value of -1 as an input, until RDX returns -1. A status code of TDX_METADATA_FIELD_SKIP indicates that the returned value is not applicable.

5.5.24.3.2. Enumeration

Availability of TDG.VP.RD enumerated by TDX_FEATURES0.ENHANCED_METADATA (bit 3), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.VP.RD returns a TDX_OPERAND_INVALID(RAX) status.

5.5.24.4. Operands Information

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

Table 5.442 TDG.VP.RD Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|-----------------|---------------|--------|---------------------|----------------|-----------------------------|
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | R | Opaque | N/A | Shared(i) |

5.5.24.5. Completion Status Codes

Table 5.443: TDG.VP.RD Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|--|---|
| TDX_METADATA_FIELD_ID_INCORRECT | |
| TDX_METADATA_FIELD_NOT_READABLE | |
| TDX_METADATA_FIELD_SKIP | Indicates that the field value being read is not applicable and needs to be skipped. If called in a loop, use RDX as the identifier of the next field to be read, if any. |
| TDX_METADATA_FIRST_FIELD_ID_IN_CONTEXT | Indicates that the first field ID in context is returned |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDG.VP.RD is successful. |

5.5.25. 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 |
| Intel SDM, Vol. 3, Table 26-3 | Format of Interruptibility State |
| Intel SDM, Vol. 3, 28.7.1 | Interruptibility State |

Get Virtualization Exception Information for the recent #VE exception.

5.5.25.1. Input Operands

Table 5.444: TDG.VP.VEINFO.GET Input Operands Definition

| Operand | Description | | |
|---------|---|----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 3 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Version 0 is always supported. Versions 0 through 2 may be supported. See enumeration details below. |
| | 63:24 | Reserved | Must be 0 |

5.5.25.2. Output Operands

Table 5.445: TDG.VP.VEINFO.GET Output Operands Definition

| Operand | Description | | |
|---------|--|--------------|---|
| RAX | TDCALL instruction return code – see 5.5.1 | | |
| RCX | Bits | Name | Description |
| | 31:0 | Exit Reason | The 32-bit value that would have been saved into the VMCS as an exit reason if a VM exit had occurred instead of the virtualization exception |
| | 39:32 | #VE Category | If TDG.VP.VEINFO.GET was called with version 0, this field returns 0. Else, this field returns the #VE category, as defined in the [Base FAS]. |
| | 63:40 | Reserved | Reserved: 0 |
| | In case of an error, RCX returns 0. | | |
| RDX | Exit Qualification: the 64-bit value that would have been saved into the VMCS as an exit qualification if a legacy VM exit had occurred instead of the virtualization exception In case of an error, RDX returns 0. | | |
| R8 | If supported, and if TDG.VP.VEINFO.GET was called with version 2 or higher and the exit reason (in RCX[31:0]) is APIC Write, R8 returns the content written to the VAPIC page in the page offset provided in the Exit Qualification (in RDX[11:0]). See enumeration details below. Else, 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. | | |

| Operand | Description | | |
|-------------------------------------|---|---------------------------------|---|
| 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 |
| | The content of R10 is only applicable for TDX-extended #VE (injected by the TDX Module), where Exit Reason is not EPT violation (48). It should be ignored for EPT violations converted by the CPU to #VE. In case of an error, R10 returns 0. | | |
| R11 | If TDG.VP.VEINFO.GET was called with version 0 or 1, then R11 is unmodified. Else, if RAX returns TDX_SUCCESS and APX is virtualized as supported (virtual CPUID(7,1).EDX[21] is 1), then R11 returns Extended Instruction Information: the 64-bit value that would have been saved into the VMCS as an extended instruction information if a legacy VM exit had occurred instead of the virtualization exception. Else, R11 returns 0. | | |
| R12 | If TDG.VP.VEINFO.GET was called with version 0 or 1, then R12 is unmodified. Else, R12 returns the following information: | | |
| | Bits | Name | Description |
| | 31:0 | Interruptibility State | If supported, this field returns the 32-bit value that would have been saved into the VMCS Interruptibility State if a legacy VM exit had occurred instead of the virtualization exception. See enumeration details below. Else, this field returns 0. |
| | 63:32 | Reserved | Set to 0 |
| In case of an error, R12 returns 0. | | | |
| Other | Unmodified | | |

5.5.25.3. Leaf Function Description

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

5 **5.5.25.3.1. Overview**

TDG.VP.VEINFO.GET returns the virtualization exception information of a #VE exception that was previously delivered to the guest TD. For details, see the [Base Spec] section on Virtualization Information (#VE).

Note: VMCS fields are returned by TDG.VP.VEINFO.GET as-is. Some parts of the returned values may not be applicable for specific #VE cases. See the [Intel SDM] for details.

10 **5.5.25.3.2. Enumeration**

Support of TDG.VP.VEINFO.GET version 1 is enumerated by TDX_FEATURES0.VE_REDUCTION (bit 30).

Support of TDG.VP.VEINFO.GET version 2 is enumerated by either of the following bits of TDX_FEATURES0, readable by TDG.SYS.RD*, being set to 1:

- APX (bit 28)
- ENHANCED_INTR_VIRTUALIZATION (bit 45)
- 5 • VEINFO_INTR_STATE (bit 46)

VAPIC page write value return by R8 is enumerated by TDX_FEATURES0.ENHANCED_VIRTUALIZATION (bit 45).

Interruptibility State return by R12[31:0] is enumerated by TDX_FEATURES0.VEINFO_INTR_STATE (bit 46).

5.5.25.4. Operands Information

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

Table 5.446: TDG.VP.VEINFO.GET Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|-----------------|---------------|--------|---------------------|----------------|-----------------------------|
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | RW | Opaque | N/A | Shared(i) |

5.5.25.5. Completion Status Codes

Table 5.447: TDG.VP.VEINFO.GET Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------|---|
| TDX_NO_VE_INFO | There is no Virtualization Exception information. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDG.VP.VEINFO.GET is successful. |

15

5.5.26. TDG.VP.VMCALL Leaf

Perform a TD Exit to the host VMM.

5.5.26.1. Input Operands

Table 5.448: TDG.VP.VMCALL Input Operands Definition

| Operand | Description | | |
|-----------------------------------|---|----------------|---|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 0 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | <p>A bitmap that controls which part of the guest TD GPR and XMM state is passed as-is to the VMM and back</p> <p>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.</p> <p>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.</p> <p>The value of RCX is passed to the host VMM.</p> | | |
| | Bits | Name | Description |
| | 15:0 | GPR Mask | Controls the transfer of GPR values: Bit 0: RAX – must be 0 Bit 1: RCX – must be 0 Bit 2: RDX Bit 3: RBX Bit 4: RSP – must be 0 Bit 5: RBP – if the TD's CONFIG_FLAG.NO_RBP_MOD is 1, then this bit must be 0. See the enumeration note below. Bit 6: RSI Bit 7: RDI Bits 15:8: R15 – R8 |
| | 31:16 | XMM Mask | Controls the transfer of XMM15 – XMM0 register values |
| | 63:32 | Reserved | Reserved: must be 0 |
| RBX, RDX, RBP, RSI, RDI, R8 – R15 | <p>If the corresponding bit in RCX is set to 1, the register value passed as-is to the host VMM on SEAMRET.</p> <p>Else, the register value is not used as an input and is preserved.</p> <p>If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP can't be used to pass values to the host VMM. See the enumeration note below.</p> | | |
| XMM0 – XMM15 | <p>If the corresponding bit in RCX is set to 1, the register value passed as-is to the host VMM on SEAMRET.</p> <p>Else, the register value is not used as an input and is preserved.</p> | | |

5.5.26.2. Output Operands

Table 5.449: TDG.VP.VMCALL Output Operands Definition

| Operand | Description |
|--|---|
| RAX | TDCALL instruction return code: returns a constant value of TDX_SUCCESS (0) |
| RCX | Unmodified |
| RBX, RDX, RBP, RDI, RSI, R8 – R15 | If the corresponding bit in RCX is set to 1, the register value passed as-is from the host VMM's SEAMCALL(TDH.VP.ENTER) input. Else, the register value is unmodified. If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP can't be used to pass values from the host VMM and is not modified from its input value. See the enumeration note below. |
| XMM0 – XMM15 | If the corresponding bit in RCX is set to 1, the register value passed as-is from the host VMM's SEAMCALL(TDH.VP.ENTER) input. Else, the register value is unmodified. |
| Other | Unmodified |

5.5.26.3. Leaf Function Description

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

5.5.26.3.1. Overview

- 10 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.

5.5.26.3.2. Enumeration

- 15 Control of RBP usage as an input/output parameter by the TD's CONFIG_FLAG.NO_RBP_MOD is enumerated by TDX_FEATURES0.NO_RBP_MOD (bit 18), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, then RBP can be used by TDG.VP.VMCALL to pass information between the guest TD and the host VMM, although highly discouraged since it contradicts normal calling conventions ABI.

5.5.26.3.3. L2 VM Details

- 20 TDG.VP.VMCALL may be invoked by an L2 VM, if enabled by the L1 VMM for the current VCPU (by setting TDVPS.L2_CTL.S_ENABLE_TDVMCALL). If not enabled, then TDG.VP.VMCALL results in an L2 → L1 exit.

- On subsequent TD resumption, the host VMM may request resumption into L1 by setting TDH.VP.ENTER's RESUME_L1 flag. In this case, L1 is resumed (i.e., the TDG.VP.ENTER it has invoked is terminated) with a TDX_L2_EXIT_HOST_ROUTED_TDVMCALL status. The L2 VCPU state reflects the successful completion of TDG.VP.VMCALL.
- 25

5.5.26.4. Operands Information

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

Table 5.450: TDG.VP.VMCALL Operands Information Definition

| Explicit/ Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-----------------------|------|-------------|-----------------|---------------|--------|---------------------|----------------|-----------------------------|
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | RW | Opaque | N/A | Shared(i) |

5.5.26.5. Completion Status Codes**Table 5.451: 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 in a completion of SEAMCALL(TDH.VP.ENTER) on the host VMM side. Later, the host VMM executed SEAMCALL(TDH.VP.ENTER) again, and execution returned to the guest TD VCPU (in TDX non-root mode) completing TDG.VP.VMCALL. |

5

5.5.27. TDG.VP.WR Leaf

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

5.5.27.1. Input Operands

Table 5.452: TDG.VP.WR Input Operands

| Operand | Description | | |
|---------|--|----------------|--|
| RAX | TDCALL instruction leaf number and version, see 5.5.1 | | |
| | Bits | Field | Description |
| | 15:0 | Leaf Number | Selects the TDCALL interface function: 10 |
| | 23:16 | Version Number | Selects the TDCALL interface function version Must be 0 |
| | 63:24 | Reserved | Must be 0 |
| RCX | Reserved, must be 0 | | |
| RDX | Field identifier – see 3.10 The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0. WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and FIELD_SIZE components of the field identifier are ignored. | | |
| R8 | Data to write to the field | | |
| R9 | A 64b write mask to indicate which bits of the value in R8 are to be written to the field | | |

5

5.5.27.2. Output Operands

Table 5.453: TDG.VP.WR Output Operands Definition

| Operand | Description |
|---------|---|
| RAX | TDCALL instruction return code – see 5.5.1 |
| R8 | Previous contents of the field In case of an error, as indicated by RAX, R8 returns 0. |
| Other | Unmodified |

5.5.27.3. Leaf Function Description

10 [Intel SDM, Vol.3, Appendix A](#) [VMX Capabilities Reporting Facility](#)

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

5.5.27.3.1. Overview

15 TDG.VP.WR writes a VCPU-scope metadata field (control structure field) of a TD. The value (R8) is written as specified by the write mask (R9). Writing is subject to the field's internal write mask (per the TD's ATTRIBUTES.DEBUG bit).

Table 5.454: Metadata Field Write Rules

| Write Mask Bit in R9 | Internal Write Mask Bit | Value Bit in R8 |
|----------------------|-------------------------|---|
| 0 | N/A | Silently ignored |
| 1 | 0 | Must be the same as the current field's bit |
| 1 | 1 | Written to the current field's bit |

Writing of specific fields may also be subject to additional rules, e.g.:

- Writing of L2 VMCS fields is subject to the VMX capabilities reported by the applicable virtual values of IA32_VMX_* MSRs, as described in [Intel SDM, Vol.3, Appendix A].
- Guest CR0 and CR4 values are subject to the CR0/4 guest host mask and read shadow settings. For details, see the [TD Partitioning Spec].

5.5.27.3.2. Enumeration

Availability of TDG.VP.WR enumerated by TDX_FEATURES0.ENHANCED_METADATA (bit 3), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.VP.WR returns a TDX_OPERAND_INVALID(RAX) status.

5.5.27.4. Operands Information

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

Table 5.455 TDG.VM.WR Operands Information Definition

| Explicit/Implicit | Reg. | Ref Type | Resource | Resource Type | Access | Access Semantics | Align Check | Concurrency Restrictions |
|-------------------|------|----------|-----------------|---------------|--------|------------------|-------------|--------------------------|
| Implicit | N/A | N/A | TDR page | TDR | None | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDCS structure | TDCS | R | Opaque | N/A | Shared(i) |
| Implicit | N/A | N/A | TDVPS structure | TDVPS | RW | Opaque | N/A | Shared(i) |

5.5.27.5. Completion Status Codes

Table 5.456: TDG.VP.WR Completion Status Codes (Returned in RAX) Definition

| Completion Status Code | Description |
|------------------------------------|--|
| TDX_METADATA_FIELD_ID_INCORRECT | |
| TDX_METADATA_FIELD_NOT_READABLE | |
| TDX_METADATA_FIELD_NOT_WRITABLE | |
| TDX_METADATA_FIELD_VALUE_NOT_VALID | |
| TDX_METADATA_WR_MASK_NOT_VALID | |
| TDX_OPERAND_BUSY | Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. |
| TDX_OPERAND_INVALID | |
| TDX_SUCCESS | TDG.VP.WR is successful. |
| TDX_TD_VMCS_FIELD_NOT_INITIALIZED | |

