Intel® Virtualization Technology for Directed I/O

Architecture Specification

November 2025

Revision 5.10

Order Number: D51397-018

Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

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 which includes subject matter disclosed herein.

All product plans and roadmaps are subject to change without notice.

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

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exception that you may publish an unmodified copy. You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

Contents

Figures

Tables

Revision History

Date	Revision	Description
March 2006	Draft	• Preliminary Draft Specification
May 2007	1.0	• 1.0 Specification
September 2007	1.1	• Specification update for x2APIC support
September 2008	1.2	• Miscellaneous documentation fixes/clarifications, including BIOS support for NUMA, hot-plug
February 2011	1.3	• Fixed documentation errors; Added BIOS support to report X2APIC_OPT_OUT
January 2012	2.0	• Updated chapter 8 (BIOS requirements) to comprehend platforms with ACPI devices capable of generating DMA requests (such as Low Power Subsystem (LPSS) on client platforms).
August 2013	2.1	• Extended page group request with a stream response requested flag to request stream responses for page requests except the last request in group. • Added an Blocked-On-Fault field to page requests requesting stream response as a hint to indicate the respective fault caused a blocking condition on the endpoint device. • Clarified hardware behavior on page requests received when page request queue is full.
September 2013	2.2	• Added support for Shared Virtual Memory (SVM) capability. • Fixed ANDD structure definition in DMAR ACPI table to support 2-byte length field. • Fixed invalidation granularity encoding for extended IOTLB invalidation descriptor. • Updated bit positions of fields in PASID-State table entry.
October 2014	2.3	• Added support for Interrupt Posting capability support. • Clarified specific registers whose read completions are required to drain various types of interrupt requests generated by the remapping hardware. • Fixed typo in effective memory-type computation for first-level paging entry accesses when nested translations are enabled with Extended Memory Type disabled in second-level translation tables. • Fixed Page Request Status Register and Page Request Event Control Register descriptions to clarify that queueing of any page_req_desc in the page request queue results in hardware setting the Pending Page Request (PPR) field. • Fixed Supervisor Request Enable (SRE) field location from extended-context-entry to PASID-table entry, to distinguish privileged versus non-privileged PASIDs of a device. • Fixed Extended Access Flag Enable (EAFE) field location from PASID-table entry to extended-context-entry. • Relaxed context-entry programming considerations to clarify software requirement to ensure self-consistency when modifying present root, extended-root, context or extended-context entries. • Reserved Translation Type (TT) field encoding of 110b and 111b in extended-context-entries (previously documented incorrectly as PASID-only translation types).

Date	Revision	Description
June 2016	2.4	• Fixed location of PASID Support enumeration field in ECAP_REG from bit28 to bit 40. • Fixed typo in Section 4.2.3 to clarify that for translation-requests-with-PASID with PR=1, remapping hardware supporting supervisor-requests (SRS=1) return PRIV bit as always 1. Previous versions of the spec. incorrectly specified hardware returning PRIV bit as 1 only if the U/S field is 0 in at least one of the first-level paging-structure entries controlling the translation. • Clarified the ordering requirement to be followed by remapping hardware on page request descriptor writes and recoverable fault reporting event interrupt. • Updated Chapter 6 to include Device-TLB invalidation throttling support for SR-IOV devices. New Device-TLB Invalidation Throttling (DIT) capability field added to ECAP_REG. • Updated Chapter 6 to include a new Page-request Drain (PD) flag in inv_wait_dsc for page request draining. • Updated Chapter 7 to include details on page request and page response ordering and draining, including handling of terminal conditions on device with pending page faults. • Added ECAP_REG capability fields to report support for Device-TLB invalidation throttling and page-request draining. • Clarified Caching Mode (CM=1) behavior to indicate that the reserved Domain-ID of 0 is used only for context-cache and rest of the caching structures follow same tagging for cached entries for CM=0 and CM =1(including for cached faulting entries when CM=1).
November 2017	2.5	• Updated specification to include support for 5-level paging structures for first-level and second-level translation. Use of 5-level paging for first-level translation is controlled through an explicit enable per PASID-table entry. Use of 5-level paging for second-level translation is controlled through the programming of already existing Address Width (AW) field in the context/extended-context entry. New capability bit is added to report support for 5-level paging for first-level translation. Added FL-PML5E and SL-PML5E documentation. • Clarified the contents of address field in Fault Recording register when hardware supports multiple paging modes (E.g., 4-level and 5-level paging). • Fixed typo error in Section 4.1.3 from translation-requests to translated-requests. • Fixed error in Table-14 in Section 7.2.1.4 that had incorrectly listed translation-requests with PASID that fail ERE and SRE checks as returning Unsupported Request in Translation Completion Status and report Fault Reason of 19h and 1Ah. Instead, these conditions result in Recoverable Fault conditions for Translation Requests as the Translation Completion Data Entry returns R=W=U=S=0 (as documented in Table-16 in Section 7.2.2) • Clarified software must wait for the current Protected Memory Enable (PMEN) register control operation to be completed by hardware and reported in the status register before updating it again. • Added Fault code 30h in Appendix A to accommodate implementations logging a non-recoverable fault when a Page (PRS) request is blocked due to Present (P) or Page Request Enable (PRE) fields Clear in corresponding extended-context-entry. • Clarified that if a request-with-PASID with PR=1 (Privileged Request) is received by a remapping hardware implementation that reports SRS (Supervisory Requests Supported) as 0 (not supported), the translation completion entry is forced with PRIV=1 (same value as PR in request) and permissions forced to 0 (R=W=E=0). • Added new DMA_CTRL_PLATFORM_OPT_IN_FLAG in DMAR ACPI table for platform firmware to report compliance about platform initiated DMA restricted to RMRR ranges when transferring control to system software such as on *ExitBootServices()*. System software may program DMA remapping hardware to block DMA outside of RMRR, except for memory explicitly registered by device drivers with system software.
June 2018	3.0	• Removed all text related to Extended-Mode. • Added support for scalable-mode translation for DMA Remapping, that enables PASID-granular first-level, second-level, nested and pass-through translation functions. • Widen invalidation queue descriptors and page request queue descriptors from 128 bits to 256 bits and redefined page-request and page-response descriptors. • Listed all fault conditions in a unified table and described DMA Remapping hardware behavior under each condition. Assigned new code for each fault condition in scalable-mode operation. • Added support for Accessed/Dirty (A/D) bits in second-level translation. • Added support for submitting commands and receiving response from virtual DMA Remapping hardware. • Added a table on snooping behavior and memory type of hardware access to various remapping structures as appendix. • Move Page Request Overflow (PRO) fault reporting from Fault Status register (FSTS_REG) to Page Request Status register (PRS_REG).

Date	Revision	Description
June 2019	3.1	• Updated handling of requests to interrupt address range. • Clarified how hardware sets value of N field in Translation Completion in Scalable-mode (Table 10 of section 4.2.3). • Added support for RID-PASID capability (RPS field in ECAP_REG). • Clarified that software must use 16-byte aligned atomic operation to update certain fields in legacy-mode context-entry and scalable-mode PASID-table-entry with Present (P) bit Set • Clarified that Caching Mode (CM) bit in Capability Register does not apply to first-level mappings • Expanded Table 21 (section 6.5.2.10) to include Invalidation Queue descriptor handling when Translation Table Mode (TTM) field in Root Table Address Register is programmed with reserved values. • Added required device-TLB invalidations in guidance to software for Invalidations (section 6.5.3.3). • Updated DMA remapping related faults and handling for various requests when they encounter such faults • Removed requirement that remapping hardware process page-requests and Device-TLB invalidation completions and in the order they arrived at ingress of remapping hardware (section 7.8). • Specified atomicity requirement on remapping hardware access to various translation structures • Clarified when some of the fields of PASID-table entry are reserved and ignored. • Renamed 5LP capability to FL5LP, renamed SMPWC capability to SMPWCS and Clarified conditions under which hardware implementation must report Nested Translation Support (NEST) as Clear in Extended Capability Register (ECAP_REG) in section 10.4.3 • Updated how hardware treats upper bits of Fault Info (FI) field in Fault Recording Register (FRCD_REG) in section 10.4.14. • Clarified hardware handling of DMA requests accessing protected memory region in section 10.4.16. • Removed RsvdZ check on Descriptor Width (DW) field in Invalidation Queue Address Register (IQA_REG) in section 10.4.23. • Added Invalidation Queue Error Record Register (IQERCD_REG) as section 10.4.29. • Clarified that Page Request Overflow (PRO) trigger a Page Request Event in section 10.4.35 (and not Fault Event). • Updated Error Code definitions in Virtual Command Response Register (VCRSP_REG) in section 10.4.45. • Removed redundant table about Interrupt remapping related fault from chapter 7. • Added software requirement to block DMA from accessing Posted-Interrupt descriptors. • Added Interrupt remapping fault reason for VT-d hardware to detect and report inappropriate requests to interrupt range.
October 2020	3.2	• Added definition for SoC Integrated Address Translation Cache (SATC) Reporting Structure Type and provide an example of DMAR Structure layout in memory (Section 8.8). Added requirements for device-TLB implementation and validation for SoC integrated devices (Section 4.4). Provide guidance to software on enabling and disable Address Translation Service (Section 4.5). • Remove Transient Mapping (TM) field from second-level page-tables and treat the field as Reserved(0). • Added Enhanced SRTP and SIRTP capabilities (Section 11.4.2) • Added new fault conditions under which hardware will block DMA operations (Table 30). • Improved guidance to software on required/recommended invalidation after page-table modifications (Section 6.5.3.3). • Enhance Translation Disable command to invalidate all DMA remapping translation caches (Section 11.4.4.1). • Update PMR definition to call out impact of Compute Express Link (CXL) and clarify that software must not program PMR region to overlap with MMIO (Section 11.4.8). • Added recommendation for software to enable ACS on PCI Express Root Ports (Section 3.12.3). • Clarified that Trigger Mode and Level are fixed at value of 0 for remapping hardware generated interrupts. • Clarified that Invalidation Descriptors with invalid values in Granularity field are treated as invalid descriptors. • Improved readability of sections on remapping hardware snoop behavior (Table 6), handling of requests to interrupt range (Section 3.15), non-snoop access (N) field in translation completion (Table 10).

Date	Revision	Description
April 2021	3.3	• Added Abort DMA Mode (Section 3.4.4). • Deprecated Register Based Invalidation (Section 6.5.1). • Clarified hardware handling of drains for IOTLB Invalidations (Section 6.5.2.3). • Updated recommendation on usage of Protected Memory Registers (Section 11.4.8). • Clarified that IOTLB may cache pass-through translations (Section 6.2.4). • Removed EIMD field from Interrupt Data Registers and treat the field as reserved (Section 11.4.7.3, Section 11.4.9.6, Section 11.4.11.6). • Updated Virtual Command Register, Virtual Command Opcode B Register, Virtual Command Response Register, and Virtual Command Capability Register. (Section 11.4.16.1,Section 11.4.16.2, Section 11.4.16.3,and Section 11.4.16.4)
December 2021	3.4	• Renamed "First-level" to "First-stage". • Renamed "Second-level" to "Second-stage". • Updated architecture so that remapping hardware always uses memory-type of write-back (WB) when accessing First-stage and Second-stage translation structures (Section 3.11, Section 9.6, Section 9.7). • Remove 'Private Data' field from Page Request/Response (Section 7.4.1.1, Section 7.6.1) • Added definition for SoC Integrated Device Property (SIDP) Reporting Structure (Section 8.9) and enhanced Device Scope Entry (Section 8.3.1) to support SIDP • Added address space limit that software must honor when using first-stage translation structures for translating IO Virtual Address (Section 3.6). • Enhanced software guidance on required IOTLB invalidation when changing PASID-tables (Table 28). • Added that Page Snoop (PGSNP) field in PASID-table entry is treated as Reserved(0) for implementations not supporting Snoop Control (Section 9.6). • Enhanced description of RW1C and RW1CS register attributes (Section 11.3).
May 2022	4.0	• Added VT-d Performance Monitoring Extensions (Chapter 10 and Section 11.4.13) • Added Enhanced Command Interface (Section 11.4.14) • Updated DRHD Structure to support increased size of register set. (Section 8.3) • Removal of Advanced Fault Logging • Updated Guidance to Software for Invalidations (Table 28)
March 2023	4.1	• Updated System Topology Diagram (Figure 1-1) • Updated software guidance on usage of Abort DMA Mode when RMRRs are present. (Section 3.4.4) • Removed architectural support of requests-with-pasid with a value of 1 for Execute-Requested (ER). • Updated the invalidation granularity details for IOTLB Invalidate Descriptor and P_IOTLB Invalidate Descriptor (Section 6.5.2.3 and Section 6.5.2.6) • Updated definition of DMA Remapping Fault codes SPT4.4, SFS.5, and SGN.8 • Updated description of AW field in Scalable-Mode PASID Table Entry to reflect when field is treated as Reserved(0). (Section 9.6) • Added new occupancy performance monitoring events (Table 49) • Updated descriptions of SAGAW and MGAW fields in regards to when SSTS is reported as Clear. (Section 11.4.3) • Updated conditions of when FRCD.PRIV has a valid value (Section 11.4.7.6) • Added software guidance to IQA_REG (Section 11.4.9.3) • Updated Invalidation Queue Error Info (IQEI) to include abort-dma mode. (Section 11.4.9.9) • Updated bit width of PERFCNTR_PASIDFLTR_REG.PFM (Section 11.4.13.18) • ECCAP_REG split into 128-bit registers. (Section 11.4.14.5)

Date	Revision	Description
August 2024	5.0	• Added support for Second-Stage I/O Read/Write Permissions. (Section 9.8) • Added support for Host Permission Tables. (Section 4.2.4) • Added support for Translated Requests with PASID. (Section 4.1.3) • Added support for RDT Configuration. (Section 11.4.14.5.4) • Updated Interrupt Remapping hardware operation for implementations requiring Extended Interrupt Mode (Section 5.1.4 and Section 5.1.6) • Updated the NMI mode in the Delivery Mode field of the Interrupt Remapping Table Entry (Section 9.9) • Updated Fault Conditions table to include Translated-req-with-PASID. (Table 30) • Updated definitions of faults LCT.4.3,SCT.4.2,SPT.2, SSS.2, and SGN.7. (Table 30) • Updated performance monitoring event list and error checks. (Section 10.5 and Section 10.7) • Updated one-shot mask in GCMD register description (Section 11.4.4.1) • Renamed "Global" field to "Global Invalidate" in *p_dev_tlb_inv_dsc* (Section 6.5.2.6) • Updated Address Type Filter bitmap (Section 11.4.13.19) • Separated 128-bit registers ECCAP_0 and ECCAP_1 into separate 64-bit registers labeled ECCAP_0, ECCAP_1,ECCAP_2, and ECCAP_3. (Section 11.4.14.5)
November 2025	5.10	• Updated PASID Table programming considerations in Section 6.2.3.1. • Updated Optional Invalidation Guidance in Section 6.5.3.5 • Removed S.3 Condition from Table 30. • Changed bit 66 of PGR descriptor from RsvdZ to Ignore. (Section 7.6.1) • Added capability bit to advertise support for TDX Connect. (Section 11.4.3) • Added reference to TDX Connect Arch Spec in Section 1.3

1 Introduction

This document describes the Intel® Virtualization Technology (Intel® VT) for Directed I/O (Intel® VT-d); specifically, it describes the components supporting I/O virtualization as it applies to platforms that use Intel® processors and core logic chipsets complying with Intel® platform specifications.

Figure 1-1 illustrates the general platform topology.

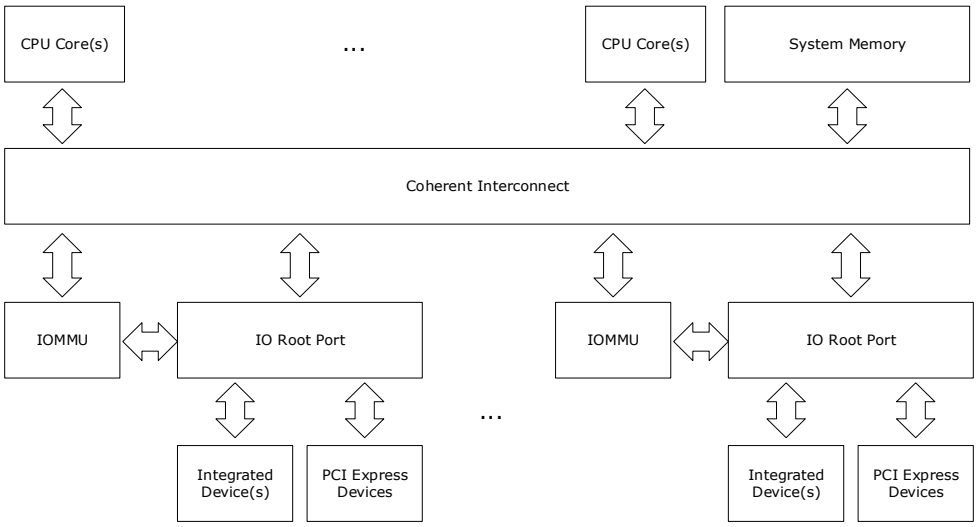


Figure 1-1. General Platform Topology

The document includes the following topics:

- An overview of I/O subsystem hardware functions for virtualization support.
- A brief overview of expected usages of the generalized hardware functions.
- The theory of operation of hardware, including the programming interface.

The following topics are not covered (or are covered in a limited context):

- Intel® Virtualization Technology for Intel® 64 Architecture. For more information, refer to the "*Intel® 64 Architecture Software Developer's Manual, Volume 3B: System Programming Guide*".

1.1 Audience

This document is aimed at hardware designers developing Intel® platforms or core-logic providing hardware support for virtualization. The document is also expected to be used by Operating System (OS) and Virtual Machine Monitor (VMM) developers utilizing the I/O virtualization hardware functions.

1.2 Glossary

The document uses the terms listed in the following table.

Table 1. Glossary

Term	Definition
Context	A hardware representation of state that identifies a device and the domain to which the device is assigned.
Context-cache	Remapping hardware cache that stores device to domain mappings
Device-TLB	A translation cache at the endpoint device (as opposed to in the platform).
DMA	Direct Memory Access: Address routed in-bound requests from I/O devices
DMA Remapping	The act of translating the address in a DMA request to a host physical address (HPA).
Domain	A collection of physical, logical, or virtual resources that are allocated to work together. Used as a generic term for virtual machines, partitions, etc.
DMA Address	Address in a DMA request: Depending on the software usage and hardware capabilities, DMA address can be Guest Physical Address (GPA), Guest Virtual Address (GVA), Virtual Address (VA), or I/O Virtual Address (IOVA).
First-Stage Paging	Paging structures used in scalable-mode for first-stage of DMA address translation.
First-Stage Caches	Translation caches used by remapping hardware units to cache intermediate (non-leaf) entries of the first-stage paging structures. These may include PML5 cache, PML4 cache, PDP cache, and PDE cache.
GAW	Guest Address Width: Physical addressability limit within a partition (virtual machine)
GPA	Guest Physical Address: the view of physical memory from software running in a partition (virtual machine).
Guest	Software running within a virtual machine environment (partition).
GVA	Guest Virtual Address: Processor virtual address used by software running in a partition (virtual machine).
HAW	Host Address Width: the DMA physical addressability limit for a platform.
HPA	Host Physical Address: Physical address used by hardware to access memory and memory-mapped resources.
HPT	Host Permission Table; Paging structures used for access rights verification of translated requests.
IEC	Interrupt Entry Cache: A translation cache in remapping hardware unit that caches frequently used interrupt-remapping table entries.
IOTLB	I/O Translation Lookaside Buffer: an address translation cache in remapping hardware unit that caches effective translations from DVA (GPA) to HPA.
I/OxAPIC	I/O Advanced Programmable Interrupt Controller
IOVA	I/O Virtual Address: Virtual address created by software for use in I/O requests.
Interrupt Remapping	The act of translating an interrupt request before it is delivered to the CPU complex.
MGAW	Maximum Guest Address Width: the maximum guest physical addressability supported by a remapping hardware implementation.
MSI	Message Signaled Interrupts.
Second-Stage Caches	Translation caches used by remapping hardware units to cache intermediate (non-leaf) entries of the second-stage (SS) paging structures. Depending on the Guest Address Width supported by hardware, these may include SS-PML5 cache, SS-PML4 cache, SS-PDP cache, and SS-PDE cache.

Table 1. Glossary

Term	Definition
PASID	Process Address Space Identifier that identifies the address space targeted by DMA requests. For requests with PASID, the PASID value is provided in the PASID TLP prefix of the request. For requests without PASID, the PASID value is programmed in the scalable-mode context-entry used to process the request.
PASID-cache	Remapping hardware cache that caches frequently accessed PASID-table entries used to translate DMA requests.
Second-Stage Paging	Paging structures used for second-stage of DMA address translation.
Source ID	A 16-bit identification number to identify the source of a DMA or interrupt request. For PCI family devices this is the 'Requester ID' which consists of PCI Bus number, Device number, and Function number.
Root-Complex	Refers to one or more hardware components that connect processor complexes to the I/O and memory subsystems. The chipset may include a variety of integrated devices.
VA	Virtual Address: Virtual address used by software on a host processor.
VMM	Virtual Machine Monitor: a software layer that controls virtualization. Also referred to as hypervisor in this document.
x2APIC	The extension of xAPIC architecture to support 32-bit APIC addressability of processors and associated enhancements.

1.3 References

Table 2. References

Description
Intel® 64 Architecture Software Developer's Manuals and Intel® Resource Director Technology (Intel® RDT) Architecture Specification http://www.intel.com/sdm
PCI Express* Base Specification Revision 5.0, Version 1.0 http://www.pcisig.com/specifications/pciexpress
Intel® Scalable I/O Virtualization Architecture Specification, Version 1.1 https://software.intel.com/en-us/articles/intel-sdm#specification
ACPI Specification http://www.acpi.info/
PCI Express* to PCI/PCI-X Bridge Specification, Revision 1.0 http://www.pcisig.com/specifications/pciexpress/bridge
Compute Express Link 2.0 Specification https://www.computeexpresslink.org/download-the-specification
Flexible Return and Event Delivery Specification https://www.intel.com/content/www/us/en/content-details/819481/flexible-return-and-event-delivery-fred-specification.html
Intel®TDX Connect Architecture Specification https://www.intel.com/content/www/us/en/content-details/862706/intel-tdx-connect-architecture-specification.html
Intel®TDX Connect Architecture ABI Reference Specification https://www.intel.com/content/www/us/en/content-details/795381/intel-tdx-module-architecture-application-binary-interface-abi-reference-specification.html

2 Overview

This chapter provides a brief overview of Intel® VT, the virtualization software ecosystem it enables, and hardware support offered for processor and I/O virtualization.

2.1 Intel® Virtualization Technology Overview

Intel® VT consists of technology components that support virtualization of platforms based on Intel® processors, thereby enabling the running of multiple operating systems and applications in independent partitions. Each partition behaves like a virtual machine (VM) and provides isolation and protection across partitions. This hardware-based virtualization solution, along with virtualization software, enables multiple usages such as server consolidation, activity partitioning, workload isolation, embedded management, legacy software migration, and disaster recovery.

2.2 VMM and Virtual Machines

Intel® VT supports virtual machine architectures comprised of two principal classes of software:

- **Virtual-Machine Monitor (VMM)**: A VMM acts as a host and has full control of the processor(s) and other platform hardware. VMM presents guest software (see below) with an abstraction of a virtual processor and allows it to execute directly on a logical processor. A VMM is able to retain selective control of processor resources, physical memory, interrupt management, and I/O.
- **Guest Software**: Each virtual machine is a guest software environment that supports a stack consisting of an operating system (OS) and application software. Each operates independently of other virtual machines and uses the same interface to processor(s), memory, storage, graphics, and I/O provided by a physical platform. The software stack acts as if it were running on a platform with no VMM. Software executing in a virtual machine must operate with reduced privilege so that the VMM can retain control of platform resources.

The VMM is a key component of the platform infrastructure in virtualization usages. Intel® VT can improve the reliability and supportability of virtualization infrastructure software with programming interfaces to virtualize processor hardware. It also provides a foundation for additional virtualization support for other hardware components in the platform.

2.3 Hardware Support for Processor Virtualization

Hardware support for processor virtualization enables simple, robust and reliable VMM software. VMM software relies on hardware support on operational details for the handling of events, exceptions, and resources allocated to virtual machines.

Intel® VT provides hardware support for processor virtualization. For Intel® 64 processors, this support consists of a set of virtual-machine extensions (VMX) that support virtualization of processor hardware for multiple software environments by using virtual machines.

2.4 I/O Virtualization

A VMM must support virtualization of I/O requests from guest software. I/O virtualization may be supported by a VMM through any of the following models:

- *Emulation*: A VMM may expose a virtual device to guest software by emulating an existing (legacy) I/O device. VMM emulates the functionality of the I/O device in software over whatever physical devices are available on the physical platform. I/O virtualization through emulation provides good compatibility (by allowing existing device drivers to run within a guest), but pose limitations with performance and functionality.
- *New Software Interfaces*: This model is similar to I/O emulation, but instead of emulating legacy devices, VMM software exposes a synthetic device interface to guest software. The synthetic device interface is defined to be virtualization-friendly to enable efficient virtualization compared to the overhead associated with I/O emulation. This model provides improved performance over emulation, but has reduced compatibility (due to the need for specialized guest software or drivers utilizing the new software interfaces).
- *Assignment*: A VMM may directly assign the physical I/O devices to VMs. In this model, the driver for an assigned I/O device runs in the VM to which it is assigned and is allowed to interact directly with the device hardware with minimal or no VMM involvement. Robust I/O assignment requires additional hardware support to ensure the assigned device accesses are isolated and restricted to resources owned by the assigned partition. The I/O assignment model may also be used to create one or more I/O container partitions that support emulation or software interfaces for virtualizing I/O requests from other guests. The I/O-container-based approach removes the need for running the physical device drivers as part of VMM privileged software.
- *I/O Device Sharing*: In this model, which is an extension to the I/O assignment model, an I/O device supports multiple functional interfaces, each of which may be independently assigned to a VM. The device hardware itself is capable of accepting multiple I/O requests through any of these functional interfaces and processing them utilizing the device's hardware resources.

Depending on the usage requirements, a VMM may support any of the above models for I/O virtualization. For example, I/O emulation may be best suited for virtualizing legacy devices. I/O assignment may provide the best performance when hosting I/O-intensive workloads in a guest. Using new software interfaces makes a trade-off between compatibility and performance, and device I/O sharing provides more virtual devices than the number of physical devices in the platform.

2.5 Intel® Virtualization Technology For Directed I/O Overview

A general requirement for all of above I/O virtualization models is the ability to isolate and restrict device accesses to the resources owned by the partition managing the device. Intel® VT for Directed I/O provides VMM software with the following capabilities:

- *I/O device assignment*: for flexibly assigning I/O devices to VMs and extending the protection and isolation properties of VMs for I/O operations.
- *DMA remapping*: for supporting address translations for Direct Memory Accesses (DMA) from devices.
- *Interrupt remapping*: for supporting isolation and routing of interrupts from devices and external interrupt controllers to appropriate VMs.
- *Interrupt posting*: for supporting direct delivery of virtual interrupts from devices and external interrupt controllers to virtual processors.
- *Reliability*: for recording and reporting of DMA and interrupt errors to system software that may otherwise corrupt memory or impact VM isolation.

2.5.1 Hardware Support for DMA Remapping

To generalize I/O virtualization and make it applicable to different processor architectures and operating systems, this document refers to *domains* as abstract isolated environments in the platform to which a subset of host physical memory is allocated.

DMA remapping provides hardware support for isolation of device accesses to memory, and enables each device in the system to be assigned to a specific domain through a distinct set of paging structures. When the device attempts to access system memory, the DMA-remapping hardware intercepts the access and utilizes the page tables to determine whether the access can be permitted; it also determines the actual location to access. Frequently used paging structures can be cached in hardware. DMA remapping can be configured independently for each device, or collectively across multiple devices.

2.5.1.1 OS Usages of DMA Remapping

There are several ways in which operating systems can use DMA remapping:

- *OS Protection*: An OS may define a domain containing its critical code and data structures, and restrict access to this domain from all I/O devices in the system. This allows the OS to limit erroneous or unintended corruption of its data and code through incorrect programming of devices by device drivers, thereby improving OS robustness and reliability.
- *Feature Support*: An OS may use domains to better manage DMA from legacy devices to high memory (For example, 32-bit PCI devices accessing memory above 4GB). This is achieved by programming the I/O page-tables to remap DMA from these devices to high memory. Without such support, software must resort to data copying through OS "bounce buffers".
- *DMA Isolation*: An OS may manage I/O by creating multiple domains and assigning one or more I/O devices to each domain. Each device-driver explicitly registers its I/O buffers with the OS, and the OS assigns these I/O buffers to specific domains, using hardware to enforce DMA domain protection. See Figure 2-1.
- *Shared Virtual Memory*: For devices supporting appropriate PCI Express[1] capabilities, OS may use the DMA remapping hardware capabilities to share virtual address space of application processes with I/O devices. Shared virtual memory along with support for I/O page-faults enable application programs to freely pass arbitrary data-structures to devices such as graphics processors or accelerators, without the overheads of pinning and marshalling of data.

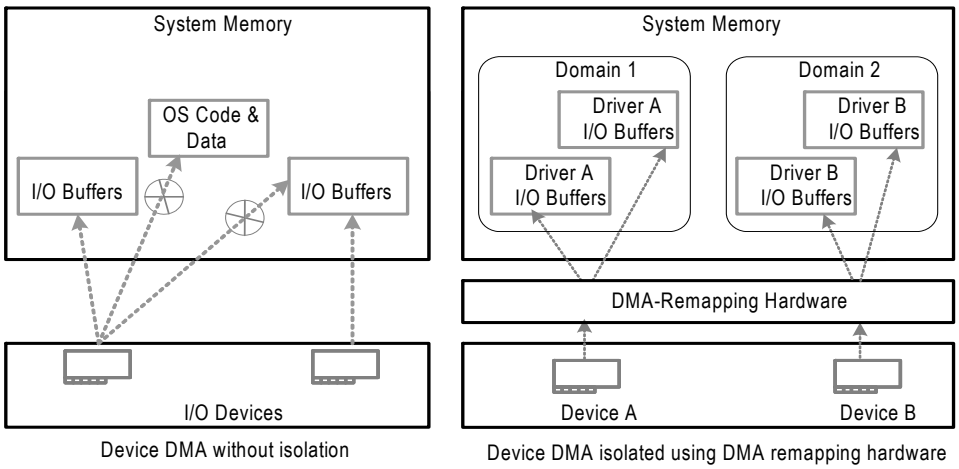


Figure 2-1. Example OS Usage of DMA Remapping

1. Refer to Process Address Space ID (PASID) capability in PCI Express* base specification.

2.5.1.2 VMM Usages of DMA Remapping

The limitations of software-only methods for I/O virtualization can be improved through direct assignment of I/O devices to partitions. With this approach, the driver for an assigned I/O device runs only in the partition to which it is assigned and is allowed to interact directly with the device hardware with minimal or no VMM involvement. The hardware support for DMA remapping enables this direct device assignment without device-specific knowledge in the VMM. See Figure 2-2.

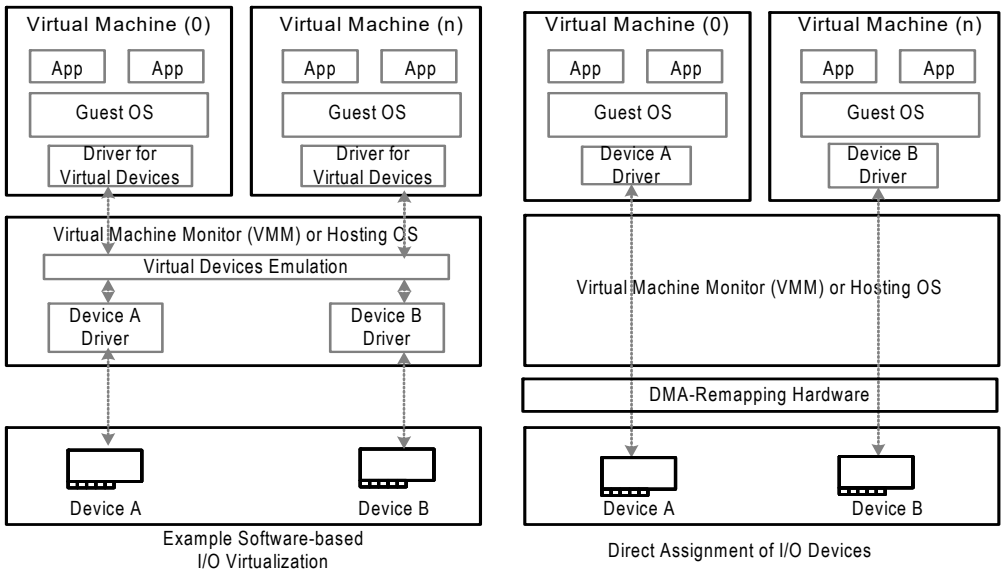


Figure 2-2. Example Virtualization Usage of DMA Remapping

In this model, the VMM restricts itself to enabling direct assignment of devices to their partitions. Rather than invoking the VMM for all I/O requests from a partition, the VMM is invoked only when guest software accesses protected resources (such as configuration accesses, interrupt management, etc.) that impact system functionality and isolation.

To support direct assignment of I/O devices, a VMM must enforce isolation of DMA requests. I/O devices can be assigned to domains, and the remapping hardware can be used to restrict DMA from an I/O device to the physical memory presently owned by its domain. For domains that may be relocated in physical memory, the remapping hardware can be programmed to perform the necessary translation.

I/O device assignment allows other I/O sharing usages — for example, assigning an I/O device to an I/O partition that provides I/O services to other user partitions. Remapping hardware enables virtualization software to choose the right combination of device assignment and software-based methods for I/O virtualization.

DMA-remapping also enables hardware based I/O virtualization technologies such as Single Root I/O Virtualization (SR-IOV) and Intel® Scalable I/O Virtualization (Intel® Scalable IOV)[1]. With SR-IOV capability, a device Physical Function may be configured to support multiple Virtual Functions (VFs)

1. Refer to PCI Express* specification for SR-IOV architecture details. Refer to Intel® Scalable I/O Virtualization architecture specification for Intel® Scalable IOV architecture details.

that can be assigned to different partitions. Similarly with Intel® Scalable IOV capability, a device Physical Function may be configured to support multiple light-weight Assignable Device Interfaces (ADIs) that can similarly be assigned to different partitions as virtual devices.

2.5.1.3 DMA Remapping Usages by Guests

A guest OS running in a VM may benefit from the availability of remapping hardware to support the usages described in Section 2.5.1.1. To support such usages, the VMM may virtualize the remapping hardware to its guests. For example, the VMM may intercept guest accesses to the virtual remapping hardware registers, and manage a shadow copy of the guest remapping structures that is provided to the physical remapping hardware. On updates to the guest I/O page tables, the guest software performs appropriate virtual invalidation operations. The virtual invalidation requests may be intercepted by the VMM, to update the respective shadow page tables and perform invalidations of remapping hardware. Due to the non-restartability of faulting DMA transactions (unlike CPU memory management virtualization), a VMM cannot perform lazy updates to its shadow remapping structures. To keep the shadow structures consistent with the guest structures, the VMM may expose virtual remapping hardware with eager pre-fetching behavior (including caching of not-present entries) or use processor memory management mechanisms to write-protect the guest remapping structures.

On hardware implementations supporting two stages of address translations (first-stage translation to remap a virtual address to intermediate (guest) physical address, and second-stage translations to remap a intermediate physical address to machine (host) physical address), a VMM may virtualize guest OS use of first-stage translations without shadowing page-tables, but by configuring hardware to perform nested translation of first and second stages.

2.5.1.4 Interaction with Processor Virtualization

Figure 2-3 depicts how system software interacts with hardware support for both processor-level virtualization and Intel® VT for Directed I/O.

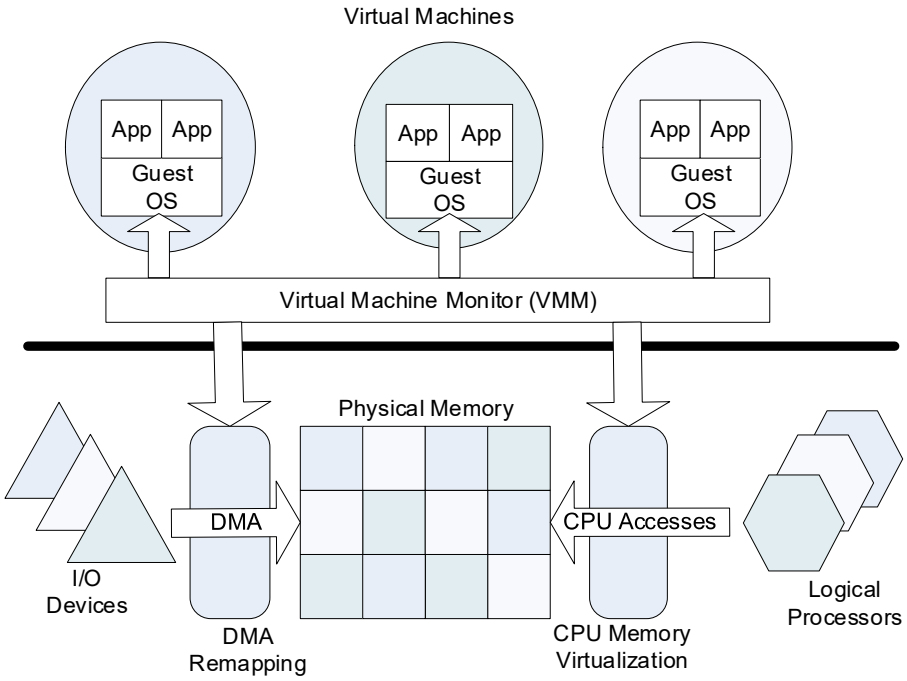


Figure 2-3. Interaction Between I/O and Processor Virtualization

The VMM manages processor requests to access physical memory via the processor's memory management hardware. DMA requests to access physical memory use remapping hardware. Both processor memory management and DMA memory management are under the control of the VMM.

2.5.2 Hardware Support for Interrupt Remapping

Interrupt remapping provides hardware support for remapping and routing of interrupt requests from I/O devices (generated directly or through I/O interrupt controllers). The indirection achieved through remapping enables isolation of interrupts across partitions.

The following usages are envisioned for the interrupt-remapping hardware.

2.5.2.1 Interrupt Isolation

On Intel® architecture platforms, interrupt requests are identified by the Root-Complex as DWORD sized write transactions targeting an architectural address range (FEEx_xxxxh). The interrupt requests are self-describing (i.e., attributes of the interrupt request are encoded in the request address and data), allowing any DMA initiator to generate interrupt messages with arbitrary attributes.

The interrupt-remapping hardware may be utilized by a Virtual Machine Monitor (VMM) to improve the isolation of external interrupt requests across domains. For example, the VMM may utilize the interrupt-remapping hardware to distinguish interrupt requests from specific devices and route them to the appropriate VMs to which the respective devices are assigned. The VMM may also utilize the interrupt-remapping hardware to control the attributes of these interrupt requests (such as destination CPU, interrupt vector, delivery mode etc.).

Another example usage is for the VMM to use the interrupt-remapping hardware to disambiguate external interrupts from the VMM owned inter-processor interrupts (IPIs). Software may enforce this by ensuring none of the remapped external interrupts have attributes (such as vector number) that matches the attributes of the VMM IPIs.

2.5.2.2 Interrupt Migration

The interrupt-remapping architecture may be used to support dynamic re-direction of interrupts when the target for an interrupt request is migrated from one logical processor to another logical processor. Without interrupt-remapping hardware support, re-balancing of interrupts require software to re-program the interrupt sources. However re-programming of these resources are non-atomic (requires multiple registers to be re-programmed), often complex (may require temporary masking of interrupt source), and dependent on interrupt source characteristics (e.g. no masking capability for some interrupt sources; edge interrupts may be lost when masked on some sources, etc.)

Interrupt-remapping enables software to efficiently re-direct interrupts without re-programming the interrupt configuration at the sources. Interrupt migration may be used by OS software for balancing load across processors (such as when running I/O intensive workloads), or by the VMM when it migrates virtual CPUs of a partition with assigned devices across physical processors to improve CPU utilization.

2.5.2.3 x2APIC Support

Intel® 64 x2APIC architecture extends the APIC addressability to 32-bits (from 8-bits). Refer to *Intel® 64 Architecture Software Developer's Manual, Volume 3B: System Programming Guide* for details.

Interrupt remapping enables x2APICs to support the expanded APIC addressability for external interrupts without requiring hardware changes to interrupt sources (such as I/OxAPICs and MSI/MSI-X devices).

2.5.3 Hardware Support for Interrupt Posting

Interrupt posting includes hardware support for optimized processing of interrupt requests from I/O devices (Physical Functions, SR-IOV Virtual Functions, or Intel® Scalable IOV Assignable Device Interfaces (ADIs)) that are directly assigned to a virtual machine. The following usages are envisioned for the interrupt-posting hardware.

2.5.3.1 Interrupt Vector Scalability

Devices supporting I/O virtualization capabilities such as SR-IOV and/or Intel® Scalable IOV, virtually increases the I/O fan-out of the platform, by allowing multiple Virtual Functions (VFs) or Assignable Device Interfaces (ADIs) to be enabled for a Physical Function (PF). Any of these PFs, VFs or ADIs can be assigned to a virtual machine. Interrupt requests from such assigned devices/resources are referred to as virtual interrupts as they target virtual processors of the assigned VM.

Each VF or ADI requires its own independent interrupt resources, resulting in more interrupt vectors needed than otherwise required without such I/O virtualization. Without interrupt-posting hardware support, all interrupt sources in the platform are mapped to the same physical interrupt vector space (8-bit vector space per logical CPU on Intel® 64 processors). For virtualization usages, partitioning the physical vector space across virtual processors is challenging in a dynamic environment when there is no static affinity between virtual process and logical processors.

Hardware support for interrupt posting addresses this vector scalability problem by allowing interrupt requests from device functions/resources assigned to virtual machines to operate in virtual vector space, thereby scaling naturally with the number of virtual machines or virtual processors.

2.5.3.2 Interrupt Virtualization Efficiency

Without hardware support for interrupt posting, interrupts from devices assigned to virtual machines are processed through the VMM software. Specifically, whenever an external interrupt destined for a virtual machine is received by the CPU, control is transferred to the VMM, requiring the VMM to process and inject corresponding virtual interrupt to the virtual machine. The control transfers associated with such VMM processing of external interrupts incurs both hardware and software overheads.

With hardware support for interrupt posting, interrupts from devices (PFs, VFs, or ADIs) assigned to virtual machines are posted (recorded) in memory descriptors specified by the VMM, and processed based on the running state of the virtual processor targeted by the interrupt.

For example, if the target virtual processor is running on any logical processor, hardware can directly deliver external interrupts to the virtual processor without any VMM intervention. Interrupts received while the target virtual processor is preempted (waiting for its turn to run) can be accumulated in memory by hardware for delivery when the virtual processor is later scheduled. This avoids disrupting execution of currently running virtual processors on external interrupts for non-running virtual machines. If the target virtual processor is halted (idle) at the time of interrupt arrival or if the interrupt is qualified as requiring real-time processing, hardware can transfer control to VMM, enabling VMM to schedule the virtual processor and have hardware directly deliver pending interrupts to that virtual processor.

This target virtual processor state based processing of interrupts reduces overall interrupt latency to virtual machines and reduces overheads otherwise incurred by the VMM for virtualizing interrupts.

2.5.3.3 Virtual Interrupt Migration

To optimize overall platform utilization, VMM software may need to dynamically evaluate the optimal logical processor to schedule a virtual processor, and in that process, migrate virtual processors across CPUs. For virtual machines with assigned devices, migrating a virtual processor across logical processors either incurs the overhead of forwarding interrupts in software (e.g., via VMM generated IPIs), or complexity to independently migrate each interrupt targeting the virtual processor to the

new logical processor. Hardware support for interrupt posting enables VMM software to atomically co-migrate all interrupts targeting a virtual processor when the virtual processor is scheduled to another logical processor.

3 DMA Remapping

This chapter describes the hardware architecture for DMA remapping. The architecture envisions remapping hardware to be implemented in the Root-Complex integrated into the Processor complex or in core logic chipset components.

3.1 Types of DMA Requests

Remapping hardware treats inbound memory requests from root-complex integrated devices and PCI Express* attached discrete devices into two categories:

- *Requests without address-space-identifier*: These are the normal memory requests from endpoint devices. These requests typically specify the type of access (read/write/atomics), targeted DMA address/size, and source-id of the device originating the request (e.g., Bus/Dev/Function).
- *Requests with address-space-identifier*: These are memory requests with additional information identifying the targeted address space from endpoint devices. Beyond attributes in normal requests, these requests specify the targeted Process Address Space Identifier (PASID), and Privileged-mode-Requested (PR) flag (to distinguish user versus supervisor access). For details, refer to the PASID Extended Capability Structure in the PCI Express specification.

For simplicity, this document refers to these categories as ***Requests-without-PASID***, and ***Requests-with-PASID***. Tagging requests of a DMA stream with a unique PASID enables scalable and fine-grained sharing of I/O devices, and operation of devices with a host application's virtual memory. Root-complexes without IOMMU or with IOMMU where DMA remapping is not enabled must ignore the PASID TLP Prefix. Later sections describe these usages for Requests-with-PASID. Versions of this specification prior to revision 2.0 supported only remapping of requests-without-PASID.

3.2 Domains and Address Translation

A domain is abstractly defined as an isolated environment in the platform, to which a subset of the host physical memory is allocated. I/O devices that are allowed to access physical memory directly are allocated to a domain and are referred to as the domain's assigned devices. For virtualization usages, software may treat each virtual machine as a domain.

The isolation property of a domain is achieved by blocking access to its physical memory from resources not assigned to it. Multiple isolated domains are supported in a system by ensuring that all I/O devices are assigned to some domain (possibly a null domain), and that they can only access the physical resources allocated to their domain. The DMA remapping architecture facilitates flexible assignment of I/O devices to an arbitrary number of domains. Each domain has a view of physical address space that may be different than the host physical address space. Remapping hardware treats the address in inbound requests as DMA Address. Depending on the software usage model, the DMA address space of a device (be it a Physical Function, SR-IOV Virtual Function, or Intel® Scalable IOV Assignable Device Interface (ADI)) may be the Guest-Physical Address (GPA) space of a virtual machine to which it is assigned, Virtual Address (VA) space of host application on whose behalf it is performing DMA requests, Guest Virtual Address (GVA) space of a client application executing within a virtual machine, I/O virtual address (IOVA) space managed by host software, or Guest I/O virtual address (GIOVA) space managed by guest software. In all cases, DMA remapping transforms the address in a DMA request issued by an I/O device to its corresponding Host-Physical Address (HPA).

Figure 3-1 illustrates DMA address translation. I/O devices 1 and 2 are assigned to domains 1 and 2, respectively. The software responsible for creating and managing the domains allocates system physical memory for both domains and sets up the DMA address translation function. DMA address in requests initiated by devices 1 & 2 are translated to appropriate HPAs by the remapping hardware.

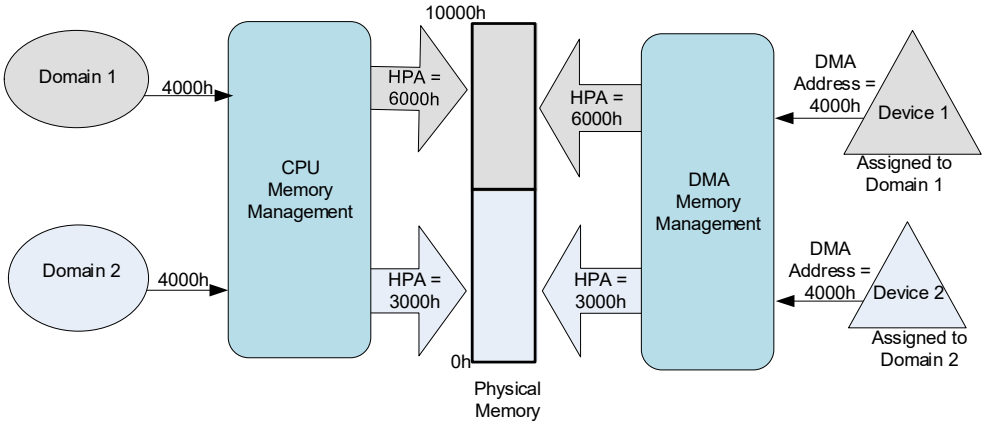


Figure 3-1. DMA Address Translation

The host platform may support one or more remapping hardware units. Each hardware unit supports remapping DMA requests originating within its hardware scope. For example, a desktop platform may expose a single remapping hardware unit in its processor complex that translates all DMA transactions. A server platform with one or more core components may support independent translation hardware units in each component, each translating DMA requests originating within its I/O hierarchy (such as a PCI Express root port). The architecture supports configurations in which these hardware units may either share the same translation data structures (in system memory) or use independent structures, depending on software programming.

The remapping hardware translates the address in a request to host physical address (HPA) before further hardware processing (such as address decoding, snooping of processor caches, and/or forwarding to the memory controllers).

3.3 Remapping Hardware - Software View

The remapping architecture allows hardware implementations supporting a single PCI segment group to expose (to software) the remapping function either as a single hardware unit covering the entire PCI segment group, or as multiple hardware units, each supporting a mutually exclusive subset of devices in the PCI segment group hierarchy. For example, an implementation may expose a remapping hardware unit that supports one or more integrated devices on the root bus, and additional remapping hardware units for devices behind one or a set of PCI Express root ports. The platform firmware (BIOS) reports each remapping hardware unit in the platform to software. Chapter 8 describes the reporting structure through ACPI constructs.

For hardware implementations supporting multiple PCI segment groups, the remapping architecture requires hardware to expose independent remapping hardware units (at least one per PCI segment group) for processing requests originating within the I/O hierarchy of each segment group.

3.4 Mapping Devices to Domains

The following sub-sections describe the DMA remapping architecture and data structures used to map I/O devices to domains.

3.4.1 Source Identifier

Each inbound request appearing at the address-translation hardware is required to identify the device originating the request. The attribute identifying the originator of an I/O transaction is referred to as the "source-id" in this document. The remapping hardware may determine the source-id of a transaction in implementation-specific ways. For example, some I/O bus protocols may provide the originating device identity as part of each I/O transaction. In other cases (for Root-Complex integrated devices, for example), the source-id may be derived based on the Root-Complex internal implementation.

For PCI Express devices, the source-id is the requester identifier in the PCI Express transaction layer header. The requester identifier of a device, which is composed of its PCI Bus/Device/Function number, is assigned by configuration software and uniquely identifies the hardware function that initiated the request. Figure 3-2 illustrates the Requester-id[1] as defined by the PCI Express Specification.

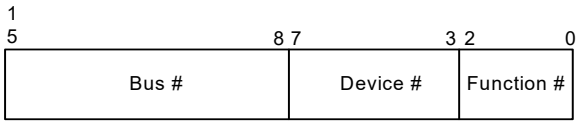


Figure 3-2. Requester Identifier Format

The following sections describe the data structures for mapping I/O devices to domains.

1. For PCI Express devices supporting Alternative Routing-ID Interpretation (ARI), bits traditionally used for the Device Number field in the Requester-id are used instead to expand the Function Number field.

3.4.2 Legacy Mode Address Translation

Figure 3-3 illustrates device to domain mapping in legacy mode.

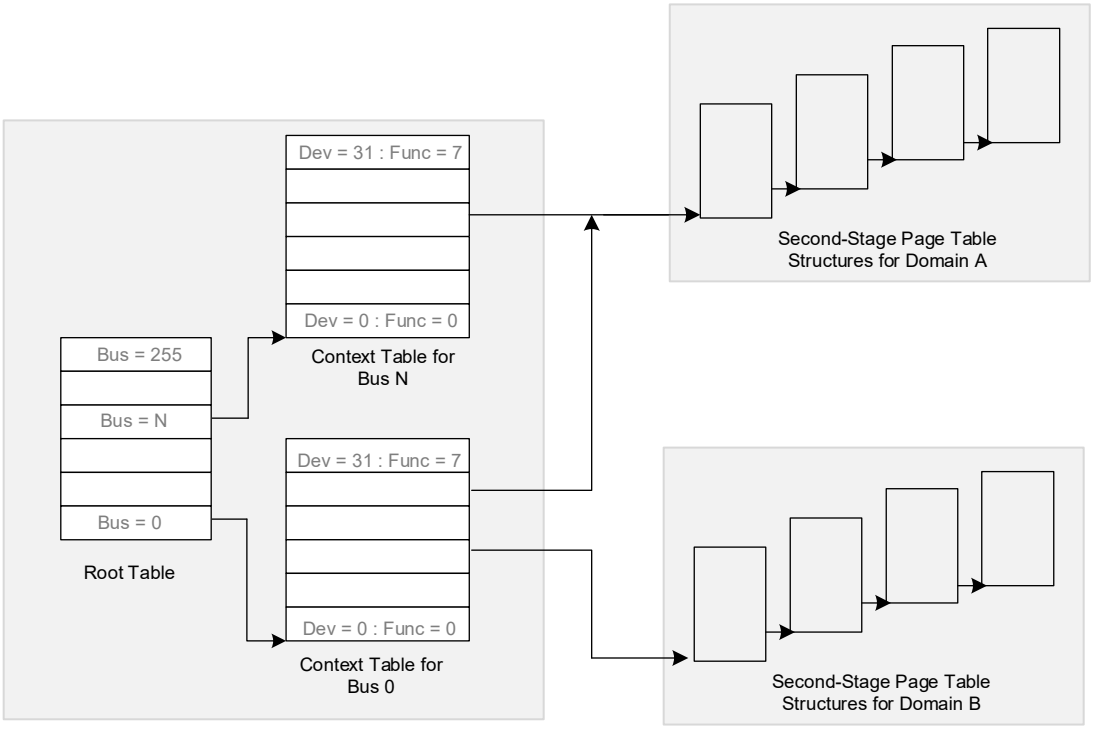


Figure 3-3. Device to Domain Mapping Structures in Legacy Mode

The root-table functions as the top level structure to map devices to their respective domains. The location of the root-table in system memory is programmed through the Root Table Address Register described in Section 11.4.5. Root Table Address Register (RTADDR_REG) points to a root-table when Translation Table Mode field in RTADDR_REG register is programmed to legacy mode (RTADDR_REG.TTM is 00b). The root-table is 4-KByte in size and contains 256 root-entries to cover the PCI bus number space (0-255). The bus number (upper 8-bits) encoded in a request's source-id field is used to index into the root-entry structure.The root-table-entry contains the context-table pointer which references the context-table for all the devices on the bus identified by the root-entry.

A context-entry maps a specific I/O device on a bus to the domain to which it is assigned, and, in turn, to the address translation structures for the domain. Each context-table contains 256 entries, with each entry corresponding to a PCI device function on the bus. For a PCI device, the device and function numbers (lower 8-bits) of source-id are used to index into the context-table.

Multiple devices may be assigned to the same domain by programming the context-entries for the devices to reference the same translation structures, and programming them with the same domain identifier. Root-entry format is described in Section 9.1 and context-entry format is described in Section 9.3.

3.4.3 Scalable Mode Address Translation

For implementations supporting Scalable Mode Translation (SMTS=1 in Extended Capability Register), the Root Table Address Register (RTADDR_REG) points to a scalable-mode root-table when the Translation Table Mode field in the RTADDR_REG register is programmed to scalable mode (RTADDR_REG.TTM is 01b). The scalable-mode root-table is similar to the root-table (4KB in size and containing 256 scalable-mode root-entries to cover the 0-255 PCI bus number space), but has a different format to reference scalable-mode context-entries. Each scalable-mode root-entry references a lower scalable-mode context-table and a upper scalable-mode context-table.

The lower scalable-mode context-table is 4-KByte in size and contains 128 scalable-mode context-entries corresponding to PCI functions in device range 0-15 on the bus. The upper scalable-mode context-table is also 4-KByte in size and contains 128 scalable-mode context-entries corresponding to PCI functions in device range 16-31 on the bus. Scalable-mode context-entries support both requests-without-PASID and requests-with-PASID. However unlike legacy mode, in scalable-mode, requests-without-PASID obtain a PASID value from the RID_PASID field of the scalable-mode context-entry and are processed similarly to requests-with-PASID. Implementations not supporting RID_PASID capability (ECAP_REG.RPS is 0b), use a PASID value of 0 to perform address translation for requests without PASID.

The scalable-mode context-entry contains a pointer to a scalable-mode PASID directory. The upper 14 bits (bits 19:6) of the request's PASID value are used to index into the scalable-mode PASID directory. Each present scalable-mode PASID directory entry contains a pointer to a scalable-mode PASID-table. The lower 6 bits (bits 5:0) of the request's PASID value are used to index into the scalable-mode PASID-table. The PASID-table entries contain pointers to both first-stage and second-stage translation structures, along with the PASID Granular Translation Type (PGTT) field which specifies whether the request undergoes a first-stage, second-stage, nested, or pass-through translation process.

Figure 3-4 illustrates device to domain mapping with scalable-mode context-table.

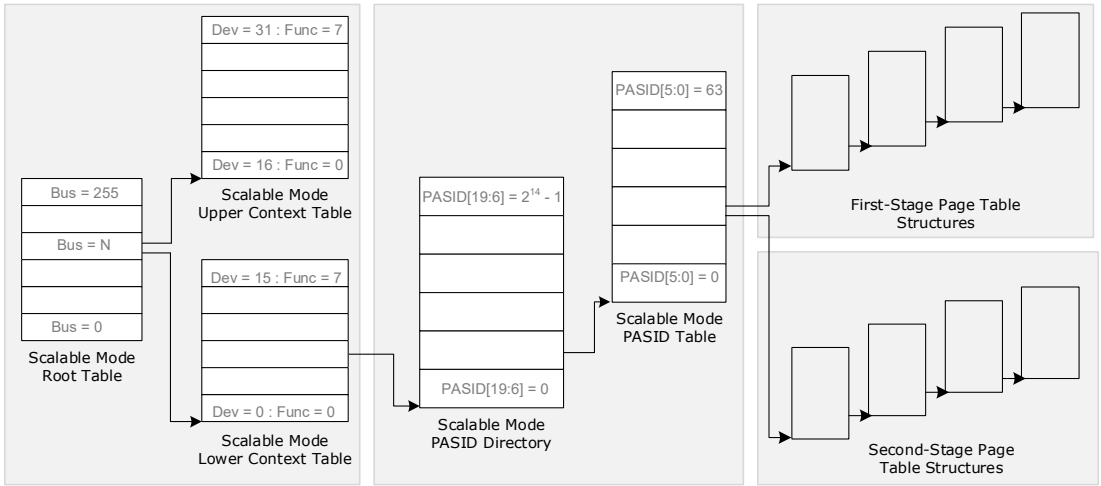


Figure 3-4. Device to Domain Mapping Structures in Scalable Mode

The scalable-mode root-entry format is described in Section 9.2, the scalable-mode context-entry format is described in Section 9.4, the scalable-mode PASID-directory-entry format is described in Section 9.5, and the scalable-mode PASID-table entry format is described in Section 9.6.

Note:	Prior version of this specification supported a limited form of address translation for requests-with-PASID, that was referred to as Extended Mode address translation (enumerated through ECAP_REG bit 24). This mode is no longer supported and replaced with scalable mode address translation. ECAP_REG bit 24 must be reported as 0 in all future implementations to ensure software backward compatibility.

3.4.4 Abort DMA Mode

For implementations supporting Abort DMA Mode (ADMS=1 in Extended Capability Register), when the Translation Table Mode (TTM) field in the Root Table Address Register (RTADDR_REG) is programmed to abort-dma mode(RTADDR_REG.TTM is 11b) the hardware behaves as if a root-table is not present. In this mode, hardware will abort all DMA operations without the need to set up a root-table with each entry marked as not-present. This mode is most useful in the process of enabling DMA remapping based memory protection. In absence of abort-dma mode, system software must find a region in memory that is protected from DMA accesses to set up translation tables and then enable DMA remapping hardware. The abort-dma mode removes the need for a DMA protected region in order to enable DMA remapping.

System software is recommended to program remapping hardware in abort-dma mode using the SRTP command, enable DMA remapping and then set up necessary translation tables and transition remapping hardware into the desired operating mode (legacy or scalable) using another SRTP command.

System software should not program remapping hardware in abort-dma mode if devices are active that require access to the reserved memory region as reported through RMRR (i.e., Bus Master Enable = 1 for such devices).

3.5 Hierarchical Translation Structures

DMA remapping uses hierarchical translation structures for both first-stage translation and second-stage translation.

For first-stage only translation and second-stage only translation, the DMA-address in the request is used as the input address. For nested translation, any address generated by first-stage translation (both addresses to access first-stage translation structures and the output address from first-stage translation) is used as the input address for nesting with second-stage translation. Section 3.6, Section 3.7, and Section 3.8 provides more details on first-stage, second-stage, and nested translation respectively.

Every paging structure in the hierarchy is 4-KByte in size, with 512 8-Byte entries. Remapping hardware uses the upper portion of the input address to identify a series of paging-structure entries. The last of these entries identifies the physical address of the region to which the input address translates (called the page frame). The lower portion of the input address (called the page offset) identifies the specific offset within that region to which the input address translates. Each paging-structure entry contains a physical address, which is either the address of another paging structure or the address of a page frame. First-stage translation supports a 4-level structure or a 5-level structure. Second-stage translation supports an N-level structure, where the value of N depends on the Guest Address Width (GAW) supported by an implementation as enumerated in the Capability Register.

The paging structures support a base page-size of 4-KByte. The page-size field in paging entries enable larger page allocations. When a paging entry with the page-size field Set is encountered by hardware on a page-table walk, the translated address is formed immediately by combining the page-base-address in the paging-entry with the unused input address bits. The remapping architecture defines support for 2-MByte and 1-GByte large-page sizes. Implementations report support for each large page size through the Capability Register.

Figure 3-5 illustrates the paging structure for translating a 48-bit address to a 4-KByte page.

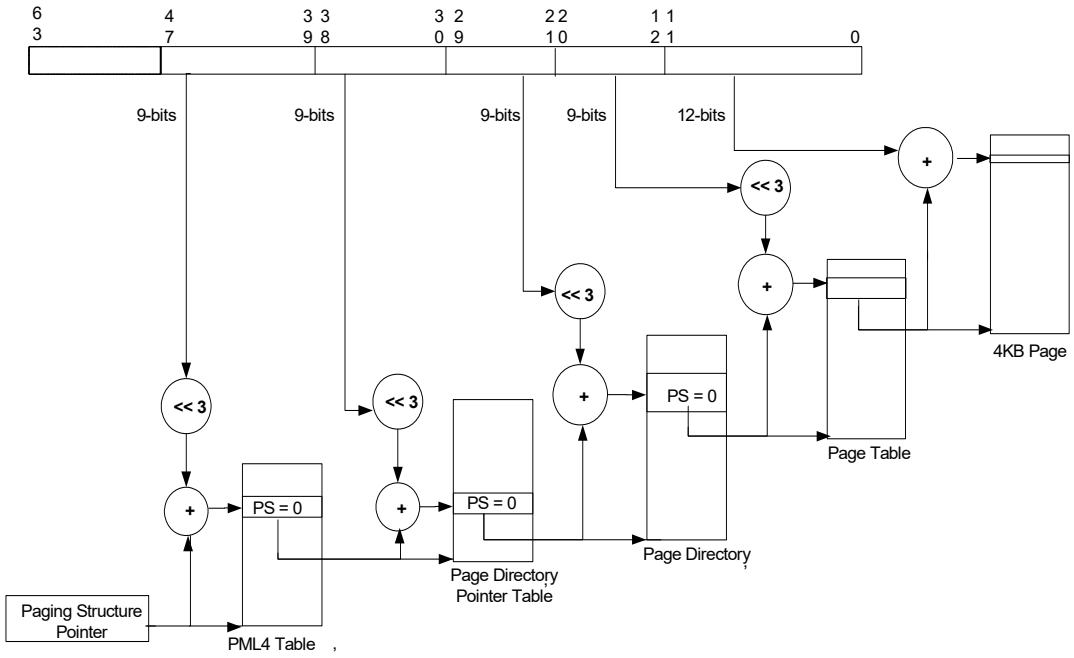


Figure 3-5. Address Translation to a 4-KByte Page

Figure 3-6 illustrates the paging structure for translating a 48-bit address to a 2-MByte large page.

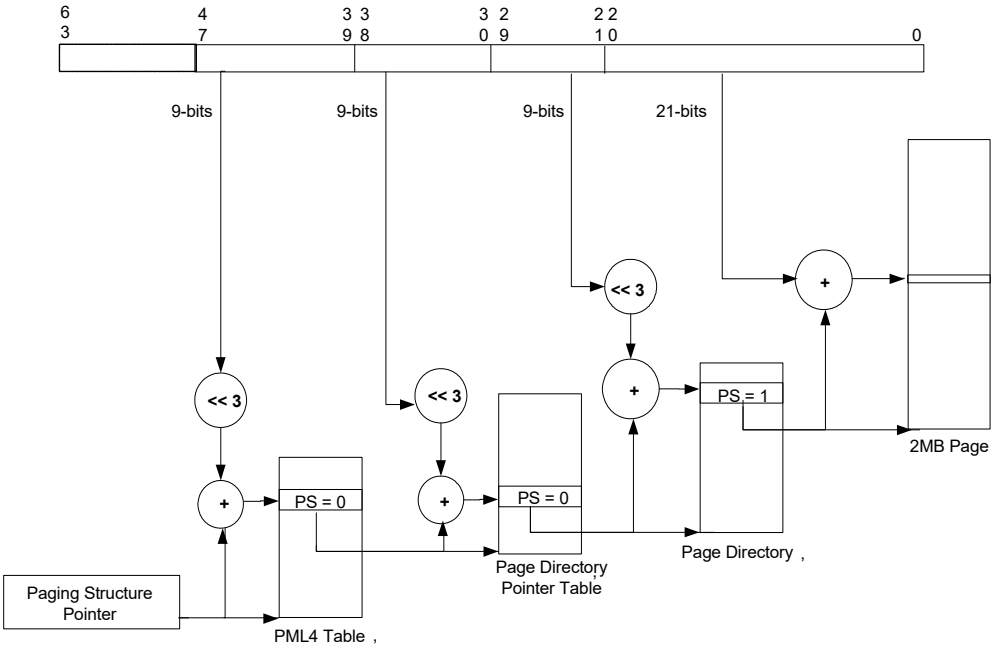


Figure 3-6. Address Translation to a 2-MByte Large Page

Figure 3-7 illustrates the paging structure for translating a 48-bit address to a 1-GByte large page.

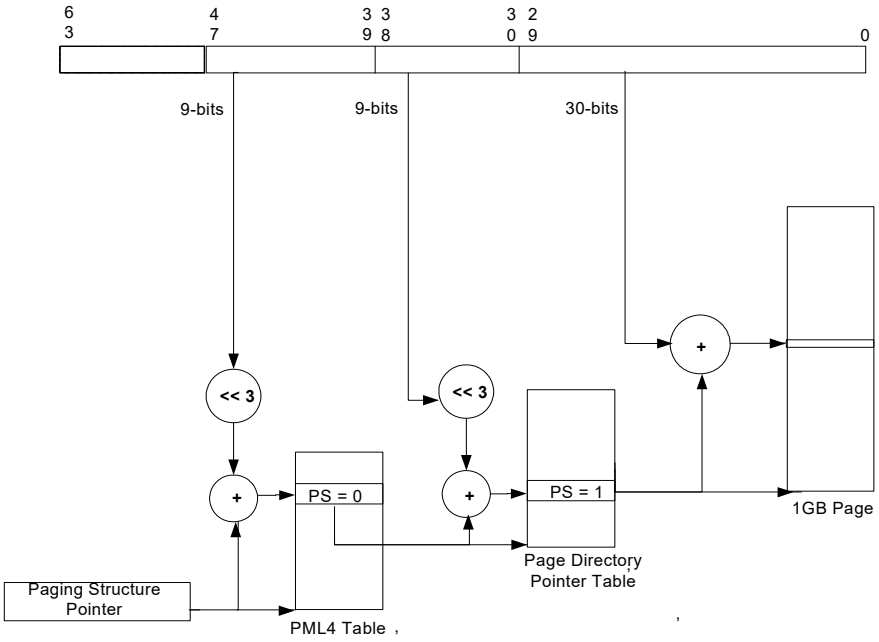


Figure 3-7. Address Translation to a 1-GByte Large Page

3.6 First-Stage Translation

Scalable-mode PASID-table entries can be configured to translate requests (with or without PASID) using first-stage translation. Requests-without-PASID use the PASID value configured in the RID_PASID field in the scalable-mode context-entry to process the request.

First-stage translation restricts the input-address to a canonical address (i.e., address bits 63:N have the same value as address bit [N-1], where N is 48 bits with 4-level paging and 57 bits with 5-level paging). Requests subject to first-stage translation by remapping hardware are subject to canonical address checking as a pre-condition for first-stage translation, and a violation is treated as a translation-fault. Chapter 7 provides details of translation-fault conditions and how they are reported to software.

Software using first-stage translation structures to translate an IO Virtual Address (IOVA) must use canonical addresses. Additionally, software must limit addresses to less than the minimum of MGAW and the lower canonical address width implied by FSPM (i.e., 47-bit when FSPM is 4-level and 56-bit when FSPM is 5-level).

First-stage translation supports the same paging structures as Intel® 64 processors when operating in 64-bit mode. Table 3 gives the different names of the first-stage translation structures, that are given based on their use in the translation process. It also provides, for each structure, the source of the physical-address used to locate it, the bits in the input-address used to select an entry from the structure, and details of whether and how such an entry can map a page. Section 9.7 describes the format of each of these paging structures in detail. For implementations supporting 5-level paging for first-stage translation, 4-level versus 5-level paging is selected based on the programming of the First Stage Paging Mode (FSPM) field in the corresponding scalable-mode PASID-table entry (see Section 9.5 and Section 9.6).

Table 3. First-Stage Paging Structures

Paging Structure	Entry Name	Physical Address of Structure	Bits Selecting Entry	Page Mapping
PML5 table	PML5E	Scalable-mode PASID-table entry (for 5-level paging)	56:48	N/A
PML4 table	PML4E	PML5E (for 5-level paging); scalable-mode PASID-table entry (for 4-level paging)	47:39	N/A
Page-directory-pointer table	PDPE	PML4E	38:30	1-GByte page (if Page-Size (PS) field is Set)
Page directory	PDE	PDPE	29:21	2-MByte page (if Page-Size (PS) field is Set)
Page table	PTE	PDE	20:12	4-KByte page

First-stage translation may map input addresses to 4-KByte pages, 2-MByte pages, or 1-GByte pages. Support for 4-KByte pages and 2-MByte pages are mandatory for first-stage translation. Implementations supporting 1-GByte pages report it through the FS1GP field in the Capability Register (see Section 11.4.2). Figure 3-5 illustrates the translation process when it produces a 4-KByte page; Figure 3-6 covers the case of a 2-MByte page; Figure 3-7 covers the case of a 1-GByte page.

The following describe the first-stage translation in more detail and how the page size is determined:

- When 5-level page-tables are used for first-stage translation, a 4-KByte naturally aligned PML5 table is located at the physical address specified in the FSPTPTR field in the scalable-mode PASID-table entry (see Section 9.5 and Section 9.6). A PML5 table comprises 512 64-bit entries (PML5Es). A PML5E is selected using the physical address defined as follows:
  - Bits 2:0 are all 0.
  - Bits 11:3 are bits 56:48 of the input address.
  - Bits 12 and higher are from the FSPTPTR field in the scalable-mode PASID-table entry.

  Because a PML5E is identified using bits 63:48 of the input address, it controls access to a 256-TByte region of the input-address space.
- When 5-level page-tables are used for first-stage translation, a 4-KByte naturally aligned PML4 table is located at the physical address specified in address (ADDR) field in the PML5E (see Table 34). If 4-level page-tables are used for first-stage translation, the 4-KByte naturally aligned PML4 table is located at the physical address specified in the FSPTPTR field in the PASID-table entry (see Section 9.5). A PML4 table comprises 512 64-bit entries (PML4Es). A PML4E is selected using the physical address defined as follows:
  - Bits 2:0 are all 0.
  - Bits 11:3 are bits 47:39 of the input address.
  - Bits 12 and higher are from the ADDR field in the PML5E (with 5-level page-tables) or from the FSPTPTR field in the scalable-mode PASID-table entry (with 4-level page-tables).

  Because a PML4E is identified using bits 63:39 of the input address, it controls access to a 512-GByte region of the input-address space.
- A 4-KByte naturally aligned page-directory-pointer table is located at the physical address specified in address (ADDR) field in the PML4E (see Table 35). A page-directory-pointer table comprises 512 64-bit entries (PDPEs). A PDPE is selected using the physical address defined as follows:
  - Bits 2:0 are all 0.
  - Bits 11:3 are bits 38:30 of the input address.

— Bits 12 and higher are from the ADDR field in the PML4E.

Because a PDPE is identified using bits 63:30 of the input address, it controls access to a 1-GByte region of the input-address space. Use of the PDPE depends on its page-size (PS) field, defined as follows in the next two bullets.

- If the PDPE's PS field is 1, the PDPE maps a 1-GByte page (see Table 36). The final physical address is computed as follows:
  - — Bits 29:0 are from the input address.
  - — Bits 30 and higher are from the ADDR field in the PDPE.
- If the PDPE's PS field is 0, a 4-KByte naturally aligned page directory is located at the physical address specified in the address (ADDR) field in the PDPE (see Table 37). A page directory comprises 512 64-bit entries (PDEs). A PDE is selected using the physical address defined as follows:
  - — Bits 2:0 are all 0.
  - — Bits 11:3 are bits 29:21 of the input address.
  - — Bits 12 and higher are from the ADDR field in the PDPE.

  Because a PDE is identified using bits 63:21 of the input address, it controls access to a 2-MByte region of the input-address space. Use of the PDPE depends on its page-size (PS) field, defined as follows in the next two bullets.
- If the PDE's PS field is 1, the PDE maps a 2-MByte page (see Table 38). The final physical address is computed as follows:
  - — Bits 20:0 are from the input address.
  - — Bits 21 and higher are from the ADDR field in the PDE.
- If the PDE's PS field is 0, a 4-KByte naturally aligned page table is located at the physical address specified in the address (ADDR) field in the PDE (see Table 39). A page table comprises 512 64-bit entries (PTEs). A PTE is selected using the physical address defined as follows:
  - — Bits 2:0 are all 0.
  - — Bits 11:3 are bits 20:12 of the input address.
  - — Bits 12 and higher are from the ADDR field in the PDE.

  Because a PTE referenced by a PDE is identified using bits 63:12 of the input address, every such PTE maps a 4-KByte page (Table 40). The final page address is translated as follows:
  - — Bits 11:0 are from the input address.
  - — Bits 12 and higher are from the ADDR field in the PTE.

If a paging-structure entry's Present (P) field (bit 0) is 0 or if the entry sets any reserved field, the entry is used neither to reference another paging-structure entry nor to map a page. A reference using a input address whose translation would use such a paging-structure entry causes a translation fault (see Chapter 7).

The following bits are reserved with first-stage translation:

- If the P field of a paging-structure entry is 1, bits 51:HAW (Host Address Width) are reserved.
- If the P field of a PML5E is 1, the PS field is reserved.
- If the P field of a PML4E is 1, the PS field is reserved.
- If 1-GByte pages are not supported and the P field of a PDPE is 1, the PS field is reserved.
- If the P field and PS field of a PDPE are both 1, bits 29:13 are reserved.
- If the P field and PS field of a PDE are both 1, bits 20:13 are reserved.
- If the Extended-Accessed flag is not supported, the EA field in the paging entries are ignored.

3.6.1 Access Rights

Requests can result in first-stage translation faults for either of two reasons: (1) there is no valid translation for the input address; or (2) there is a valid translation for the input address, but its access rights do not permit the access. Chapter 7 provides detailed hardware behavior on translation faults and reporting to software.

The accesses permitted for a request whose input address is successfully translated through first-stage translation is determined by the attributes of the request and the access rights specified by the paging-structure entries controlling the translation.

Devices report support for requests-with-PASID through the PCI Express PASID Capability structure. The PASID Capability allows software to query and control if the endpoint can issue requests-with-PASID with supervisor-privilege. Remapping hardware implementations report support for requests seeking supervisor privilege through the Extended Capability Register (see SRS fields in Section 11.4.3).

The following describes how first-stage translation determines access rights:

- For requests with supervisor privilege (value of 1 in Privilege-mode-Requested (PR) field) processed through a scalable-mode PASID-table entry with SRE (Supervisor Requests Enable) field Set:
  - — Read requests
    - Reads are allowed from any input address with a valid translation.
  - — Write requests and Atomics requests
    - If Write-Protect-Enable (WPE) field in the scalable-mode PASID-table entry used to translate the request is 0, writes are allowed to any input address with a valid translation.
    - If WPE is 1, writes are allowed to any input address with a valid translation for which the R/W field (bit 1) is 1 in every paging-structure entry controlling the translation.
- For requests with user privilege (value of 0 in Privilege-mode-Requested (PR) field):
  - — Read requests
    - Reads are allowed from any input address with a valid translation for which the U/S field is 1 in every paging-structure entry controlling the translation.
  - — Write requests and Atomics requests
    - Writes are allowed to any input address with a valid translation for which the R/W field and the U/S field are 1 in every paging-structure entry controlling the translation.

Remapping hardware may cache information from the paging-structure entries in translation caches. These caches may include information about access rights. Remapping hardware may enforce access rights based on these caches instead of on the paging structures in memory. This fact implies that, if software modifies a paging-structure entry to change access rights, the hardware might not use that change for a subsequent access to an affected input address. Refer to Chapter 6 for details on hardware translation caching and how software can enforce consistency with translation caches when modifying paging structures in memory.

3.6.2 Accessed, Extended Accessed, and Dirty Flags

For any paging-structure entry that is used during first-stage translation, bit 5 is the Accessed (A) flag. For first-stage paging-structure entries referenced through a scalable-mode PASID-table entry with EAFE=1, bit 10 is the Extended-Accessed flag. For paging-structure entries that map a page (as opposed to referencing another paging structure), bit 6 is the Dirty (D) flag. These flags are provided for use by memory-management software to manage the transfer of pages and paging structures into and out of physical memory.

- Whenever the remapping hardware uses a first-stage paging-structure entry as part of input-address translation, it atomically sets the A field in that entry (if it is not already set).

- If the Extended-Accessed-Flag-Enable (EAFE) is 1 in a scalable-mode PASID-table entry that references a first-stage paging-structure entry used by the remapping hardware, it atomically sets the EA field in that entry. Whenever EA field is atomically set, the A field is also set in the same atomic operation. For software usages where the first-stage paging structures are shared across heterogeneous agents (e.g., CPUs and accelerator devices such as GPUs), the EA flag may be used by software to identify pages accessed by non-CPU agent(s) (as opposed to the A flag which indicates access by any agent sharing the paging structures).
- Whenever there is a write to a input address, the remapping hardware atomically sets the D field (if it is not already set) in the paging-structure entry that identifies the final translated address for the input address (either a PTE or a paging-structure entry in which the PS field is 1). The atomic operation that sets the D field also sets the A field (and the EA field, if EAFE=1 as described above).
- Hardware may speculatively set the Accessed (A) flag and the Extended-Accessed (EA) flag in paging-structure entries for requests that encounter an address translation fault and are aborted.

Memory-management software may clear these flags when a page or a paging structure is initially loaded into physical memory. These flags are "sticky", meaning that, once set, the remapping hardware does not clear them; only software can clear them.

Remapping hardware may cache information from the first-stage paging-structure entries in translation caches (see Chapter 6). These caches may include information about accessed, extended-accessed, and dirty flags. This fact implies that, if software modifies an accessed flag, extended-accessed flag, or a dirty flag from 1 to 0, the hardware might not set the corresponding bit in memory on a subsequent access using an affected input address until software issues a suitable invalidation. Refer to Chapter 6 for details on hardware translation caching and how software can enforce consistency with translation caches when modifying paging structures in memory.

3.7 Second-Stage Translation

Context entries and scalable-mode PASID-Table entries can be configured to support second-stage translation. With context entries, second-stage translation applies only to requests-without-PASID. With scalable-mode PASID-Table entries, second-stage translation can be applied to all requests (with or without PASID), and can be applied nested with first-stage translation for all requests (with or without PASID). This section describes the use of second-stage translation without nesting. Section 3.8 describes the nested use of second-stage translation.

Each context entry contains a pointer to the base of a second-stage translation structure. Section 9.3 describe the exact format of the context entry. Scalable-mode context-entries reference a scalable-mode PASID structure. Each scalable-mode PASID-table entry contains a pointer to the base of a second-stage translation structure. Second-stage translation restricts the input-address to an implementation-specific address-width reported through the Maximum Guest Address Width (MGAW) field in the Capability Register. The input address is subject to MGAW address checking, and any violations are treated as a translation fault. Chapter 7 provides details of fault conditions and its reporting to software.

Second-stage translation uses a hierarchical paging structure as described in Section . To allow page-table walks with 9-bit stride, the Adjusted Guest Address Width (AGAW) value for a domain is defined as its Guest Address Width (GAW) value adjusted, such that (AGAW-12) is a multiple of 9. The AGAW indicates the number of levels of page walk. Hardware implementations report the supported AGAWs through the Capability Register. Second-stage translation may map input addresses to 4-KByte pages, 2-MByte pages, or 1-GByte pages. Implementations report support in second-stage translation for 2-MByte and 1-GByte large-pages through the Capability Register. Figure 3-5 illustrates the translation process for a 4-level paging structure when it produces a 4-KByte page; Figure 3-6 illustrates mapping to a 2-MByte page; Figure 3-7 illustrates mapping to a 1-GByte page.

Table 4. Second-stage Paging Structures

Paging Structure	Entry Name	Physical Address of Structure	Bits Selecting Entry	Page Mapping
Second-stage PML5 table	SS-PML5E	Context-entry or scalable-mode PASID-table entry (with 5-level translation)	MGAW:48	N/A
Second-stage PML4 table	SS-PML4E	SS-PML5E (with 5-level translation); Context-entry or scalable-mode PASID-table entry (with 4-level translation)	47:39	N/A
Second-stage Page-directory-pointer table	SS-PDPE	SS-PML4E (with 4-level or 5-level translation); Context-entry or scalable-mode PASID-table entry (with 3-level translation)	38:30	1-GByte page (if Page Size (PS) field is Set)
Second-stage Page directory	SS-PDE	SS-PDPE	29:21	2-MByte page (if Page-Size (PS) field is Set)
Second-stage Page table	SS-PTE	SS-PDE	20:12	4-KByte page

Table 4 gives the different names of the second-stage translation structures, that are given based on their use in the translation process. It also provides, for each structure, the source of the physical-address used to locate it, the bits in the input-address used to select an entry from the structure, and details of whether and how such an entry can map a page. Section 9.8 describes format of each of these paging structures in detail.

The following describe the second-stage translation in more detail and how the page size is determined:

- When a 5-level paging structure is in use for second-stage translation, a 4-KByte naturally aligned second-stage-PML5 table is located at the physical address specified in the SSPTPTR field in the context-entry or scalable-mode PASID-table entry. A second-stage-PML5 table comprises 512 64-bit entries (SS-PML5Es). An SS-PML5E is selected using the physical address defined as follows:
  - Bits 2:0 are all 0.
  - Bits 11:3 are bits MGAW:48 of the input address.
  - Bits 12 and higher are from the SSPTPTR field in the context-entry or scalable-mode PASID-table entry.

  Because an SS-PML5E is identified using bits MGAW:48 of the input address, it controls access to a 256-TByte region of the input-address space.
- When a 5-level paging structure is in use for second-stage translation, a 4-KByte naturally aligned second-stage-PML4 table is located at the physical address specified in the address (ADDR) field in the SS-PML5E (see Table 41). When a 4-level paging structure is in use for second-stage translation, a 4-KByte naturally aligned second-stage-PML4 table is located at the physical address specified in the SSPTPTR field in the context-entry or scalable-mode PASID-table entry. A second-stage-PML4 table comprises 512 64-bit entries (SS-PML4Es). An SS-PML4E is selected using the physical address defined as follows:
  - Bits 2:0 are all 0.
  - Bits 11:3 are bits 47:39 (for 5-level translation) or MGAW:39 (for 4-level translation) of the input address.
  - Bits 12 and higher are from the ADDR field in SS-PML5E (for 5-level translation) or from the SSPTPTR field in the context-entry or scalable-mode PASID-table entry (for 4-level translation).

  Because an SS-PML4E is identified using bits MGAW:39 of the input address, it controls access to a 512-GByte region of the input-address space.

- When a 5-level or 4-level paging structure is in use for second-stage translation, a 4-KByte naturally aligned page-directory-pointer table is located at the physical address specified in the address (ADDR) field in the SS-PML4E (see Table 42). When a 3-level paging structure is in use, the 4-KByte naturally aligned page-directory-pointer table is located at the physical address specified in the SSPTPTR field in the context-entry or scalable-mode PASID-table entry. A page-directory-pointer table comprises 512 64-bit entries (SS-PDPEs). An SS-PDPE is selected using the physical address defined as follows:
  - Bits 2:0 are all 0.
  - Bits 11:3 are bits 38:30 (for 4-level or 5-level translation) or MGAW:30 (for 3-level translation) of the input address.
  - Bits 12 and higher are from the ADDR field in the SS-PML4E (or from the SSPTPTR field in the context-entry or scalable-mode PASID-table entry for 3-level paging structures).

  Because an SS-PDPE is identified using bits MGAW:30 of the input address, it controls access to a 1-GByte region of the input-address space. Use of the SS-PDPE depends on its page-size (PS) field, defined as follows in the next two bullets.
- If the SS-PDPE's PS field is 1, the SS-PDPE maps a 1-GByte page (see Table 43). The final physical address is computed as follows:
  - Bits 29:0 are from the input address.
  - Bits 30 and higher are from the ADDR field in the SS-PDPE.
- If the SS-PDPE's PS field is 0, a 4-KByte naturally aligned second-stage page directory is located at the physical address specified in the address (ADDR) field in the SS-PDPE (see Table 44). A second-stage page directory comprises 512 64-bit entries (SS-PDEs). A PDE is selected using the physical address defined as follows:
  - Bits 2:0 are all 0.
  - Bits 11:3 are bits 29:21 of the input address.
  - Bits 12 and higher are from the ADDR field in the SS-PDPE.

  Because an SS-PDE is identified using bits MGAW:21 of the input address, it controls access to a 2-MByte region of the input-address space. Use of the SS-PDPE depends on its page-size (PS) field, defined as follows in the next two bullets.
- If the SS-PDE's PS field is 1, the SS-PDE maps a 2-MByte page (see Table 45). The final physical address is computed as follows:
  - Bits 20:0 are from the input address.
  - Bits 21 and higher are from the ADDR field in the SS-PDE.
- If the SS-PDE's PS field is 0, a 4-KByte naturally aligned second-stage page-table is located at the physical address specified in the address (ADDR) field in the SS-PDE (see Table 46). Such a second-stage page-table comprises 512 64-bit entries (SS-PTEs). An SS-PTE is selected using the physical address defined as follows:
  - Bits 2:0 are all 0.
  - Bits 11:3 are bits 20:12 of the input address.
  - Bits 12 and higher are from the ADDR field in the SS-PDE.

  Because a SS-PTE referenced by a SS-PDE is identified using bits MGAW:12 of the input address, every such SS-PTE maps a 4-KByte page (Table 47). The final page address is translated as follows:
  - Bits 11:0 are from the input address.
  - Bits 12 and higher are from the ADDR field in the SS-PTE.

If a second-stage paging-structure entry's read and write permissions are both 0 or if the entry sets any reserved field, the entry is used neither to reference another paging-structure entry nor to map a page. A reference using an input address whose translation would use such a paging-structure entry causes a translation error (see Chapter 7).

The following bits are reserved with second-stage translation:

- If either the R or W field of a paging-structure entry is 1, bits 51: HAW are reserved.
- If either the R or W field of an SS-PML5E is 1, the PS field is reserved.
- If either the R or W field of an SS-PML4E is 1, the PS field is reserved.
- If 1-GByte pages are not supported and the R or W fields of an SS-PDPE is 1, the PS field is reserved.
- If the R or W fields of an SS-PDPE is 1, and the PS field in that SS-PDPE is 1, bits 29:12 are reserved.
- If 2-MByte pages are not supported and the R or W fields of an SS-PDE is 1, the PS field is reserved.
- If either the R or W field of an SS-PDE is 1, and the PS field in that SS-PDE is 1, bits 20:12 are reserved.
- If either the R or W field of a non-leaf paging-structure entry (i.e., SS-PML5E, SS-PML4E, SS-PDPE, or SS-PDE with PS=0) is 1, the SNP (Snoop) field is reserved.
- If either the R or W field of an SS-PTE is 1, and Snoop Control (SC) is reported as 0 in the Extended Capability Register, the SNP field is reserved.

3.7.1 Access Rights

Requests can result in second-stage translation faults for either of two reasons: (1) there is no valid translation for the input address; or (2) there is a valid translation for the input address, but its access rights do not permit the access. Chapter 7 provides detailed hardware behavior on translation faults and reporting to software.

The accesses permitted for a request whose input address is successfully translated through second-stage translation is determined by the attributes of the request and the access rights specified by the second-stage paging-structure entries controlling the translation.

Devices can issue requests for reads, writes, or atomics. The following describes how second-stage translation determines access rights for such requests:

- Read request:
  - Reads are allowed from any input address with a valid translation for which read permission is granted in every paging-structure entry controlling the translation.
- Write request:
  - Writes are allowed to any input address with a valid translation for which write permission is granted in every paging-structure entry controlling the translation.
- Atomics request:
  - Atomics requests are allowed from any input address with a valid translation for which read and write permissions are both granted in every paging-structure entry controlling the translation.

Remapping hardware may cache information from the second-stage paging-structure entries in translation caches. These caches may include information about access rights. Remapping hardware may enforce access rights based on these caches instead of on the paging structures in memory. This fact implies that, if software modifies a paging-structure entry to change access rights, the hardware might not use that change for a subsequent access to an affected input address. Refer to Chapter 6 for details on hardware translation caching and how software can enforce consistency with translation caches when modifying paging structures in memory.

3.7.2 Accessed and Dirty Flags

Accessed and dirty flags support in first-stage paging-structure entries are described in Section 3.6.2. DMA Remapping hardware implementations can also support accessed and dirty flags in second-stage paging-structure entries. Software should read the ECAP_REG.SSADS field to determine whether the implementation supports this feature.

When supported, software can enable accessed and dirty flags for second-stage translation using the 'Second Stage Accessed Dirty Enable' (SSADE) field in the PASID-table entry. When enabled, for any second-stage paging-structure entry that is used during address translation, bit 8 is the accessed flag. For a second-stage paging-structure entry that maps a page (as opposed to referencing another second-stage paging structure), bit 9 is the dirty flag.

When enabled, the hardware will set the accessed and dirty flags for second-stage translation as follows:

- Whenever hardware uses a second-stage paging-structure entry as part of address translation, it atomically sets the accessed flag in that entry (if it is not already set).
- Whenever there is a write to an input address, the hardware atomically sets the dirty flag (if it is not already set) in the second-stage paging-structure entry that identifies the final physical address for the input address (either an SS-PTE or a second-stage paging-structure entry in which bit 7 (PS) is 1).
- Hardware may speculatively set the Accessed (A) flag in paging-structure entries for requests that encounter an address translation fault and are aborted.

These flags are "sticky," meaning that, once set, hardware does not clear them; only software can clear them. Hardware may cache information from the second-stage paging-structure entries in TLBs and paging-structure caches. This fact implies that, if software changes an accessed flag or a dirty flag from 1 to 0, the hardware might not set the corresponding bit in memory on a subsequent access using an affected second-stage input address until software issues a suitable invalidation. Refer to Chapter 6 for details on hardware translation caching and how software can enforce consistency with translation caches when modifying paging structures in memory.

3.8 Nested Translation

When PASID Granular Translation Type (PGTT) field is set to 011b in scalable-mode PASID-table-entry, requests translated through first-stage translation are also subjected to nested second-stage translation. Scalable-mode PASID-table entries configured for nested translation contain both the pointer to the first-stage translation structures, and the pointer to the second-stage translation structures. Nested translation can be applied to any request (with or without PASID) as request-without-PASID obtain the PASID value from RID_PASID field in scalable-mode context-entry.

Figure 3-8 illustrates the nested translation for a request mapped to a 4-KByte page through 4-level first-stage translation, and interleaved through 4-KByte mappings in 4-level second-stage paging structures.

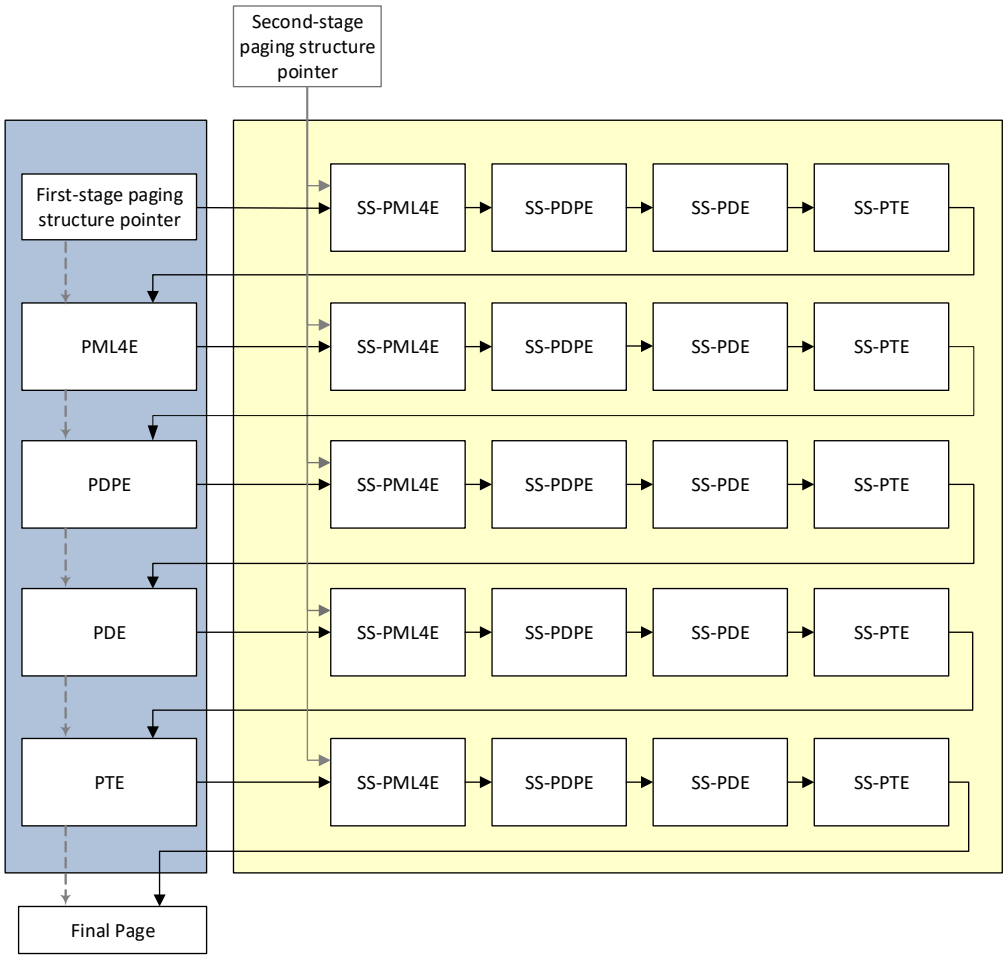


Figure 3-8. Nested Translation with 4-KByte Pages

With nesting, all memory accesses generated when processing a request through first-stage translation are subjected to second-stage translation. This includes access to first-stage paging structure entries (PML5E, PML4E, PDPE, PDE, PTE), and access to the output address from first-stage translation. Just like root-table and context-table in legacy mode address translation are in host physical address, scalable-mode root/context/PASID-directory/PASID-tables are in host physical address and not subjected to first or second stage translation.

With nested translation, a guest operating system running within a virtual machine may utilize first-stage translation as described in Section 2.5.1.3, while the virtual machine monitor may virtualize memory by enabling nested second-stage translations.

The first-stage translation follows the same process as described in Section 3.6 to map input addresses to 4-KByte, 2-MByte or 1-GByte pages. The second-stage translation is interleaved at each step, and follows the process described in Section 3.7 to map input addresses to 4-KByte, 2-MByte or 1-GByte pages.

3.8.1 Access Rights

Requests subjected to nested translation can result in fault at the first-stage translation or any of the second-stage translation stages. Translation faults at a level can result from either of two reasons: (1) there is no valid translation for the respective input address; or (2) there is a valid translation for the respective input address, but its access rights do not permit the access. Chapter 7 provides detailed hardware behavior on translation faults and reporting to software.

For requests subjected to nested translation, access rights are checked at both first and second stages of translation.

Access rights checking for first-stage translation follows the behavior described in Section 3.6.1.

Access rights for second-stage translations function as follows:

- Access to paging structures (First-stage paging structure pointer, PML5E, PML4E, PDPE, PDE, PTE) is treated as follows:
  - Accessed (A), Extended-Accessed (EA), Dirty (D) flag update of first-stage paging-structure entries
    - Atomic A/EA/D flag updates of first-stage paging-entries are allowed from any input address with a valid translation for which read and write permissions are both granted in every second-stage paging-entry controlling the translation to the respective first-stage paging-entry.
  - Reads of first-stage paging structures
    - When Second-Stage Accessed/Dirty flags are not enabled in PASID-table entry(SSADE=0) Reads of first-stage paging structures are allowed from any input address with a valid translation for which read permission is granted in every second-stage paging-entry controlling the translation to the respective first-stage paging-entry.on to the respective first-stage paging-entry.
- When Second-Stage Accessed/Dirty flags are enabled in PASID-table entry(SSADE=1), reads to first-stage paging structures are allowed from any input address with valid translation for which read and write permissions are both granted in every second-stage paging-entry. Access to the final page is treated as follows:
  - Read requests:
    - Reads are allowed from any input address with a valid translation for which read permission is granted in every second-stage paging-structure entry used to translated a final page guest-physical address into a host-physical address.
  - Write requests
    - Writes are allowed from any input address with a valid translation for which write permission is granted in every second-stage paging-entry used to translated a final page guest-physical address into a host-physical address.
  - Atomics requests
    - Atomics requests are allowed from any input address with a valid translation for which read and write permissions are both granted in every second-stage paging-entry used to translated a final page guest-physical address into a host-physical address.

With nested translations, remapping hardware may cache information from both first-stage and second-stage paging-structure entries in translation caches. These caches may include information about access rights. Remapping hardware may enforce access rights based on these caches instead of on the paging structures in memory. This fact implies that, if software modifies a paging-structure

entry to change access rights, the hardware might not use that change for a subsequent access to an affected input address. Refer to Chapter 6 for details on hardware translation caching and how software can enforce consistency with translation caches when modifying paging structures in memory.

3.9 Pass-through Translation

When DMA remapping hardware is programmed to provide pass-through translations, the hardware takes the input address and provides it back as the output address. Remapping hardware does not perform any access rights checking for pass-through translations. The input address to pass-through translation is subject to Host Address Width (HAW) address checking and any violations are treated as a translation fault. Chapter 7 provides details of translation-fault conditions and how they are reported to software.

3.10 Snoop Behavior

Snoop behavior for a memory access (to a translation structure entry or access to the mapped page) specifies if the access is coherent (snoops the processor caches) or not. The snoop behavior is independent of the memory typing described in Section 3.11. The snoop behavior for various accesses is specified as follows:

- Access to root, scalable-mode root, context, scalable-mode context, scalable-mode PASID-directory and scalable-mode PASID-table entries are snooped if the Coherency (C) field in Extended Capability Register (ECAP_REG: see Section 11.4.3) is reported as 1. These accesses are not required to be snooped if the field is reported as 0.
- Access to paging structures have snoop behavior as follows:
  - Remapping hardware is setup in legacy mode (RTADDR_REG.TTM=00b)
    - Access to paging structures is snooped if the C field in ECAP_REG is reported as 1. These accesses are not required to be snooped if the C field is reported as 0.
  - Remapping hardware is setup in scalable mode (RTADDR_REG.TTM=01b)
    - Remapping hardware encountering the need to atomically update A/EA/D bits in a paging-structure entry that is not snooped will result in a non-recoverable fault.
    - When the Scalable Mode Page-walk Coherency (SMPWC) field in ECAP_REG is reported as Clear, software must pre-set A/D bits in all first-stage paging structures to avoid a non-recoverable fault.
    - When software programs the Page-walk Snoop (PWSNP) field in a PASID-table entry as 0, software must pre-set certain bits as specified below to avoid non-recoverable fault:
      - Pre-set A/D bits in first-stage paging structures accessed through the PASID-table entry.
      - If the EAFE field is also set in the PASID-table entry, then pre-set the EA bit in first-stage paging structures accessed through the PASID-table entry.
      - If the SSADE field is also set in the PASID-table entry, then pre-set A/D bit in second-stage paging structures accessed through the PASID-table entry.
    - Access to paging structures are snooped if the Scalable Mode Page-walk Coherency (SMPWC) field in ECAP_REG is reported as 1 and Page-walk Snoop (PWSNP) field in the PASID-table entry is 1. Otherwise these accesses are not required to be snooped.
- Access to the final page by untranslated request has snoop behavior as follows:
  - If the no-snoop attribute in the request is Clear, the access to the final page is snooped.
  - If the remapping hardware is setup in scalable mode (RTADDR_REG.TTM=01b) and the Page Snoop (PGSNP) field in PASID-table entry is Set, access to the final page is snooped.

— When the remapping hardware reports Snoop Control (SC) field as 1 in the ECAP_REG, if the translation process used second-stage leaf paging structure entry with Snoop (SNP) field Set, the access to the final page is snooped.

— Otherwise access to the final page is not required to be snooped.

The table below summarizes the snoop behavior for access to translation structures by hardware and for access to the final page by untranslated request. A value of 1 implies memory access is snooped and a value of 0 implies memory access is not snooped. For snoop behavior on translation request see Table 10.

- ECAP.C is Page-walk Coherency field in ECAP_REG.
- ECAP.SC is Snoop Control field in ECAP_REG.
- ECAP.SMPWC is the Scalable Mode Page-walk Coherency field in ECAP_REG.
- Request.NS is the No-Snoop bit that comes with a DMA transaction.
- SS.Leaf.SNP is the Snoop field in leaf second stage paging structure.

Table 5. Snoop Behavior for Root/Context/PASID-structures

Mode	Root Tables/ SM Root Tables	Context Tables/ SM Context Tables	PASID Directory	PASID Table
legacy	ECAP.C	ECAP.C	NA	NA
scalable	ECAP.C	ECAP.C	ECAP.C	ECAP.C

Table 6. Snoop Behavior for FS/SS Paging Structures and Final Page

Address Translation Mode	TT (Legacy) or PGTT (Scalable)	First-stage Paging Structures	Second-stage Paging Structures	Page: Untranslated Request
legacy	TT=00b (second-stage with ATS blocked)	NA	ECAP.C	!(Request.NS) || SS.leaf.SNP
	TT=01b (second-stage with ATS allowed)	NA	ECAP.C	!(Request.NS) || SS.leaf.SNP
	TT=10b (pass-through)	NA	NA	!(Request.NS)
	TT=11b (reserved)	NA	NA	NA
scalable	PGTT=001b (first-stage)	PASID-table-entry.PWSNP	NA	!(Request.NS) || PASID-table-entry.PGSNP
	PGTT=010b (second-stage)	NA	PASID-table-entry.PWSNP	!(Request.NS) || PASID-table-entry.PGSNP || SS.leaf.SNP
	PGTT=011b (nested)	PASID-table-entry.PWSNP	PASID-table-entry.PWSNP	!(Request.NS) || PASID-table-entry.PGSNP || SS.leaf.SNP
	PGTT=100b (pass-through)	NA	NA	!(Request.NS) || PASID-table-entry.PGSNP
	PGTT=other (reserved)	NA	NA	NA

3.11 Memory Type

The memory type of a memory access (to a translation structure entry or access to the mapped page) refers to the type of caching used for that access. Refer to Intel® 64 processor specifications for definition and properties of each supported memory-type (UC, UC-, WC, WT, WB, WP). Support for memory typing in remapping hardware is reported through the Memory-Type-Support (MTS) field in the Extended Capability Register (see Section 11.4.3). This section describes how memory type is determined.

- Memory-type has no meaning (and hence is ignored) for memory accesses from devices operating outside the processor coherency domain.
- Memory-type is applicable to memory accesses through a coherent link from devices operating inside the processor coherency domain (such as Intel® processor graphics device). Memory-type is also applicable to DMA Remapping hardware accesses through a coherent link.

The following sub-sections describe details of computing memory-type from PAT, memory type from MTRR, and how to combine them to form the effective memory type.

3.11.1 Selecting Memory Type from Page Attribute Table

Memory-type selection from Page Attribute Table requires hardware to form a 3-bit index made up of the PAT, PCD and PWT bits from the respective paging-structure entries. The PAT bit is bit 7 in page-table entries that point to 4-KByte pages and bit 12 in paging-structure entries that point to larger pages. The PCD and PWT bits are bits 4 and 3, respectively, in paging-structure entries that point to pages of any size.

The PAT memory-type comes from entry *i* of the Page Attribute Table in the scalable-mode PASID-table entry controlling the request, where *i* is defined as follows:

- For access to the physical address that is the translation of an input address, *i* = 4*PAT+2*PCD+PWT, where the PAT, PCD, and PWT values come from the relevant PTE (if the translation uses a 4-KByte page), the relevant PDE (if the translation uses a 2-MByte page), or the relevant PDPE (if the translation uses a 1-GByte page).

3.11.2 Selecting Memory Type from Memory Type Range Registers

Remapping hardware implementations reporting Memory-Type-Support (MTS) field as Set in the Extended Capability Register support the Memory Type Range Registers (MTRRs). These include the MTRR Capability Register (see Section 11.4.12.1), MTRR Default Type Register (see Section 11.4.12.2), fixed-range MTRRs (see Section 11.4.12.3), and variable-range MTRRs (see Section 11.4.12.4).

Selection of memory-type from the MTRR registers function as follows:

- If the MTRRs are not enabled (Enable (E) field is 0 in the MTRR Default Type Register), then MTRR memory-type is uncacheable (UC).
- If the MTRRs are enabled (E=1 in MTRR Default Type Register), then the MTRR memory-type is determined as follows:
  - If the physical address falls within the first 1-MByte and fixed MTRRs are enabled, the MTRR memory-type is the memory-type stored for the appropriate fixed-range MTRR (see Section 11.4.12.3).
  - Otherwise, hardware attempts to match the physical address with a memory type set by the variable-range MTRRs ((see Section 11.4.12.4):
    - If one variable memory range matches, the MTRR memory-type is the memory type stored in the MTRR_PHYSBASEn_REG Register for that range.
    - If two or more variable memory ranges match and the memory-types are identical, then MTRR memory-type is that memory-type.
    - If two or more variable memory ranges match and one of the memory types is UC, then MTRR memory-type is UC.
    - If two or more variable memory ranges match and the memory types are WT and WB, then MTRR memory-type is WT.
    - For overlaps not defined by above rules, hardware behavior is undefined.
- If no fixed or variable memory range matches, then the MTRR memory-type is the default memory-type from the MTRR Default Type Register (see Section 11.4.12.2).

3.11.3 Selecting Effective Memory Type

The effective memory-type for an access is computed from the PAT memory-type and the MTRR memory-type as illustrated in Table 7 below.

Remapping hardware may cache information from the first-stage paging-structure entries in translation caches (see Chapter 6). These caches may include information about memory typing. Hardware may use memory-typing information from these caches instead of from the paging structures in memory. This fact implies that, if software modifies a paging-structure entry to change

Table 7. Effective Memory Types

MTRR Memory Type	PAT Memory Type	Effective Memory Type
UC	UC	UC
	UC-	UC
	WC	WC
	WT	UC
	WB	UC
	WP	UC
WC	UC	UC
	UC-	WC
	WC	WC
	WT	UC
	WB	WC
	WP	UC
WT	UC	UC
	UC-	UC
	WC	WC
	WT	WT
	WB	WT
	WP	WP
WB	UC	UC
	UC-	UC
	WC	WC
	WT	WT
	WB	WB
	WP	WP
WP	UC	UC
	UC-	WC
	WC	WC
	WT	WT
	WB	WP
	WP	WP

the memory-typing bits, hardware might not use that change for a subsequent translation using that entry or for access to an affected input-address. Refer to Chapter 6 for details on hardware translation caching and how software can enforce consistency with translation caches when modifying paging structures in memory.

3.11.4 Determining Memory Type

When processing requests from devices operating in the processor coherency domain, the memory type for any access is computed as follows.

3.11.4.1 Memory Type in Legacy Mode (RTADDR_REG.TTM = 00b)

- Access to root-table, context-table, second-stage translation structures, and the final page use memory-type of write-back (WB).

3.11.4.2 Memory Type in Scalable Mode (RTADDR_REG.TTM = 01b)

- Access to scalable-mode root-table, scalable-mode context-table, scalable-mode PASID-directory, scalable-mode PASID-table, first-stage translation structures, and second-stage translation structures use memory-type of write-back (WB).
- If the cache-disable (CD) field in the scalable-mode PASID-table entry used to process the request is 1, the final page uses memory-type of uncacheable (UC).
- If the cache-disable (CD) field is 0 in the scalable-mode PASID-table entry, the memory-type for accesses to the final page is computed as follows:
  - If the PASID Granular Translation Type (PGTT) field in the scalable-mode PASID-table entry has a value of 010b, the memory type is computed as follows:
    - If the extended memory-type enable (EMTE) field in the scalable-mode PASID-table entry used is 0, memory-type of write-back (WB) is used.
    - If the EMTE field in the scalable-mode PASID-table entry used is 1, the memory-type specified in the extended memory-type (EMT) field in the last (leaf) second-stage translation-structure entry is used.
  - If the PASID Granular Translation Type (PGTT) field in the scalable-mode PASID-table entry has a value of 001b or 011b, the memory type is computed as follows:
    - First, the first-stage memory-type specified by the Page Attribute Table (PAT) is computed as described in Section 3.11.1.
    - Second, the memory-type for the target physical address as specified by the Memory Type Range Registers (MTRRs) is computed as described in Section 3.11.2.
    - In the scalable-mode PASID-table entry used to process this request, if the PASID Granular Translation Type (PGTT) field is 001b (first-stage-only), the effective memory-type used is computed by combining the first-stage PAT memory-type with the MTRR memory-type computed above as described in Section 3.11.3.
    - In the scalable-mode PASID-table entry used to process this request, if the PASID Granular Translation Type (PGTT) field is 011b (nested) and the extended memory-type enable (EMTE) field is 0, the effective memory-type used is computed by combining the first-stage PAT memory-type with the MTRR memory-type computed above as described in Section 3.11.3.
    - In the scalable-mode PASID-table entry used to process this request, if the PASID Granular Translation Type (PGTT) field is 011b (nested) and the extended memory-type enable (EMTE) field is 1, the memory-type is computed as follows:
      - During the second-stage translation to access the final page, the ignore-PAT (IPAT) and the extended memory-type (EMT) field from the last (leaf) second-stage translation-structure entry used is fetched.
      - If the IPAT field is 1, the PAT memory-type computed from first-stage translation is ignored, and the memory-type specified by the EMT field is used as the memory-type for the access.
      - If the IPAT field is 0, the effective memory-type for the access is computed by combining the first-stage PAT memory-type above with the EMT field from the last (leaf) second-stage translation-structure entry. The effective memory-type computation follows the

same rules described in Table 7 in Section 3.11.3, except the memory-type specified by the EMT field is used instead of the MTRR memory-type.

— If the PASID Granular Translation Type (PGTT) field in the scalable-mode PASID-table entry has a value of 100b, memory-type of write-back (WB) is used.

The table below summarizes the memory type calculation for memory access to various translation structures.

Table 8. Memory Type Calculation for Remapping Structures and Final Page

Mode	Root Table or SM Root Table	Context Tables or SM Context Tables	PASID Directory, PASID Table	First-stage Tables	Second-stage Tables	Page
Legacy	WB	WB	NA	NA	WB	WB
Scalable	WB	WB	WB	WB	WB	see Table 9

Table 9. Memory Type Calculation for Final Page in Scalable Mode

CD	PGTT	EMTE	Page
1	**don't care**	**don't care**	UC
0	**001b (first-stage-only)**	**don't care**	Eff_mem_type(MTRR,FS.PAT)
0	**010b (second-stage-only)**	**0**	WB
0	**010b (second-stage-only)**	**1**	SS.leaf.EMT
0	**011b (nested)**	**0**	Eff_mem_type (MTRR,FS.PAT)

0	**011b (nested)**	**1**	if(SS.leaf.IPAT) { SS.leaf.EMT
			} else {
			Eff_mem_type (SS.leaf.EMT, FS.PAT) }

0	**100b (pass-through)**	**don't care**	WB

With nesting, remapping hardware may cache information from the first-stage and second-stage paging-structure entries in translation caches (see Chapter 6). These caches may include information about memory typing. Hardware may use memory-typing information from these caches instead of from the paging structures in memory. This fact implies that, if software modifies a paging-structure entry to change the memory-typing bits, hardware might not use that change for a subsequent translation using that entry or for access to an affected input-address. Refer to Chapter 6 for details on hardware translation caching and how software can enforce consistency with translation caches when modifying paging structures in memory.

3.12 Identifying Origination of DMA Requests

In order to support usages requiring isolation, the platform must be capable of uniquely identifying the requester (Source-Id) for each DMA request. The DMA sources in a platform and use of source-id in these requests may be categorized as below.

3.12.1 Devices Behind PCI Express* to PCI/PCI-X Bridges

The PCI Express-to-PCI/PCI-X bridges may generate a different requester-id and tag combination in some instances for transactions forwarded to the bridge's PCI Express interface. The action of replacing the original transaction's requester-id with one assigned by the bridge is generally referred to as taking 'ownership' of the transaction. If the bridge generates a new requester-id for a transaction forwarded from the secondary interface to the primary interface, the bridge assigns the

PCI Express requester-id using the secondary interface's bus number, and sets both the device number and function number fields to zero. Refer to the PCI Express-to-PCI/PCI-X bridge specifications for more details.

For remapping requests from devices behind PCI Express-to-PCI/PCI-X bridges, software must consider the possibility of requests arriving with the source-id in the original PCI-X transaction or the source-id provided by the bridge. Devices behind these bridges can only be collectively assigned to a single domain. When setting up remapping structures for these devices, software must program multiple context entries, each corresponding to the possible set of source-ids. Each of these context-entries must be programmed identically to ensure the DMA requests with any of these source-ids are processed identically.

3.12.2 Devices Behind Conventional PCI Bridges

For devices behind conventional PCI bridges, the source-id in the DMA requests is the requester-id of the bridge device. For remapping requests from devices behind conventional PCI bridges, software must program the context-entry corresponding to the bridge device. Devices behind these bridges can only be collectively assigned to a single domain.

3.12.3 Devices Behind PCI Express Root Port

It is recommended that software enable ACS capability on PCI Express Root Ports to prevent devices behind root ports from spoofing requester-id of other devices.

3.12.4 Root-Complex Integrated Devices

Transactions generated by all root-complex integrated devices must be uniquely identifiable through its source-id (PCI requester-id). This enables any root-complex integrated endpoint device (PCI or PCI Express) to be independently assigned to a domain.

3.12.5 PCI Express* Devices Using Phantom Functions

To increase the maximum possible number of outstanding requests requiring completion, PCI Express allows a device to use function numbers not assigned to implemented functions to logically extend the Tag identifier. Unclaimed function numbers are referred to as Phantom Function Numbers (PhFN). A device reports its support for phantom functions through the Device Capability configuration register, and requires software to explicitly enable use of phantom functions through the Device Control configuration register.

Since the function number is part of the requester-id used to locate the context-entry for processing a DMA request, when assigning PCI Express devices with phantom functions enabled, software must program multiple context entries, each corresponding to the PhFN enabled for use by the device function. Each of these context-entries must be programmed identically to ensure the DMA requests with any of these requester-ids are processed identically.

3.12.6 Single-Root I/O Virtualization Capable Devices

Single-Root I/O Virtualization (SR-IOV) architecture enables an endpoint device Physical Function (PF) to support multiple Virtual Function (VFs). To support independent assignment of PF and VFs to a domain, transactions generated by PF and VFs must each have uniquely identifiable source-id (PCI Express requester-id). Refer to the PCI Express base specification for SR-IOV endpoint architecture and requirements.

3.12.7 Intel® Scalable I/O Virtualization Capable Devices

Intel® Scalable I/O Virtualization (Intel® Scalable IOV) architecture enables fine-grained partitioning of an endpoint device Physical Function (PF) to support multiple Assignable Device Interface (ADI). Transactions generated by ADIs share the same source-id (PCI Express requester-id) as the hosting

PF. To support independent assignment of ADIs to a domain, transactions generated by ADIs must each have a uniquely identifiable PASID (Process Address Space Identifier) so that DMA-remapping hardware with Scalable Mode Translation Support can be used to apply a unique translation function per ADI. Refer to the PCI Express base specification for PASID capability and Intel® Scalable I/O Virtualization specification for Intel® Scalable IOV endpoint device requirements.

3.13 Handling Requests Crossing Page Boundaries

PCI Express memory requests are specified to disallow address/length combinations which cause a memory space access to cross a page (4KB) boundary. However, the PCI Express Specification defines checking for violations of this rule at the receivers as optional. If checked, violations are treated as malformed transaction layer packets and reported as PCI Express errors. Checking for violations from Root-Complex integrated devices is typically platform-dependent.

Platforms supporting DMA remapping are expected to check for violations of the rule in one of the following ways:

- The platform hardware checks for violations and explicitly blocks them. For PCI Express memory requests, this may be implemented by hardware that checks for the condition at the PCI Express receivers and handles violations as PCI Express errors. DMA requests from other devices (such as Root-Complex integrated devices) that violate the rule (and hence are blocked by hardware) may be handled in platform-specific ways. In this model, the remapping hardware units never receive DMA requests that cross page boundaries.
- If the platform hardware cannot check for violations, the remapping hardware units must perform these checks and re-map the requests as if they were multiple independent DMA requests.

3.14 Handling of Zero-Length Reads

A memory read request of one double-word with no bytes enabled (“zero-length read”) is typically used by devices as a type of flush request. For a requester, the semantics of the flush request allow a device to ensure that previously issued posted writes in the same traffic class have been completed at its destination.

Zero-length read requests are handled as follows by remapping hardware:

- Implementations reporting ZLR field as Clear in the Capability Register process zero-length read requests like any other read requests. Specifically, zero-length read requests are address-translated based on the programming of the remapping structures. Zero-length reads translated to memory are completed in the coherency domain with all byte enables off. Unsuccessful translations result in translation faults. For example, zero-length read requests to write-only pages in second-stage translation are blocked due to read permission violation.
- Implementations reporting ZLR field as Set in the Capability Register handles zero-length read requests same as above, except if it is to a write-only page. Zero-length read requests to write-only pages that do not encounter any faulting conditions other than read permission violation are successfully remapped and completed. Zero-length reads translated to memory complete in the coherency domain with all byte enables off. Data returned in the read completion is obfuscated.

DMA remapping hardware implementations are recommended to report ZLR field as Set and support the associated hardware behavior.

3.15 Handling Requests to Interrupt Address Range

On Intel® architecture platforms, physical address range FEEx_xxxxh is designated as the interrupt address range. Single-DWORD length write requests without PASID to this range are treated as interrupt requests and are not subjected to DMA remapping (even if translation structures specify a mapping for this range). See Chapter 5 for a description of how such requests are handled.

Zero-length-read requests without PASID to this range bypass remapping hardware and complete successfully. Data returned in the read completion is obfuscated.

The following types of requests to this range are illegal requests. They are blocked and reported as DMA Remapping faults.

- Read requests without PASID that are not ZLR.
- Atomics requests without PASID.
- Non-DWORD length write requests without PASID.

Requests-with-PASID with input address in range FEEx_xxxxh are translated normally like any other request-with-PASID through DMA-remapping hardware. However, if such a request is processed using pass-through translation, it will be blocked as described in the paragraph below.

Software must not program paging-structure entries to remap any address to the interrupt address range. Untranslated requests and translation requests that result in an address in the interrupt range will be blocked with condition code LGN.4 or SGN.8. Translated requests with an address in the interrupt address range are treated as Unsupported Request (UR).

3.16 Handling Requests to Reserved System Memory

Reserved system memory regions are typically allocated by BIOS at boot time and reported to OS as reserved address ranges in the system memory map. Requests to these reserved regions may either occur as a result of operations performed by the system software driver (for example in the case of DMA from unified memory access (UMA) graphics controllers to graphics reserved memory), or may be initiated by non system software (for example in case of DMA performed by a USB controller under BIOS SMM control for legacy keyboard emulation). For proper functioning of these legacy reserved memory usages, when system software enables DMA remapping, the second-stage translation structures for the respective devices are expected to be set up to provide identity mapping for the specified reserved memory regions with read and write permissions.

Platform implementations supporting reserved memory must carefully consider the system software and security implications of its usages. These usages are beyond the scope of this specification. Platform hardware may use implementation-specific methods to distinguish accesses to system reserved memory. These methods must not depend on simple address-based decoding since DMA virtual addresses can indeed overlap with the host physical addresses of reserved system memory.

For platforms that cannot distinguish between device accesses to OS-visible system memory and device accesses to reserved system memory, the architecture defines a standard reporting method to inform system software about the reserved system memory address ranges and the specific devices that require device access to these ranges for proper operation. Refer to Section 8.4 for details on the reporting of reserved memory regions.

For legacy compatibility, system software is expected to setup identity mapping in second-stage translation (with read and write privileges) for these reserved address ranges, for the specified devices. For these devices, the system software is also responsible for ensuring that any input addresses used for device accesses to OS-visible memory do not overlap with the reserved system memory address ranges.

3.17 Root-Complex Peer to Peer Considerations

When DMA remapping is enabled, peer-to-peer requests through the Root-Complex must be handled as follows:

- The input address in the request is translated (through first-stage, second-stage or nested translation) to a host physical address (HPA). The address decoding for peer addresses must be done only on the translated HPA. Hardware implementations are free to further limit peer-to-peer accesses to specific host physical address regions (or to completely disallow peer-forwarding of translated requests).

- Since address translation changes the contents (address field) of the PCI Express Transaction Layer Packet (TLP), for PCI Express peer-to-peer requests with ECRC, the Root-Complex hardware must use the new ECRC (re-computed with the translated address) if it decides to forward the TLP as a peer request.
- Root-ports, and multi-function root-complex integrated endpoints, may support additional peer-to-peer control features by supporting PCI Express Access Control Services (ACS) capability. Refer to ACS capability in PCI Express specifications for details.

4 Support For Device-TLBs

The DMA remapping architecture described in Chapter 3 supports address translation of DMA requests received by the Root-Complex. Hardware may accelerate the address-translation process by caching data from the translation structures. Chapter 6 describes details of these translation caches supported by remapping hardware. Translation caches at the remapping hardware is a finite resource that supports requests from multiple endpoint devices. As a result, efficiency of these translation caches in the platform can depend on number of simultaneously active DMA streams in the platform, and address locality of DMA accesses.

One approach to scaling translation caches is to enable endpoint devices to participate in the remapping process with translation caches implemented at the devices. These translation caches on the device are referred to as Device-TLBs (Device Translation Lookaside Buffers). Device-TLBs alleviate pressure for translation caches in the Root-Complex, and provide opportunities for devices to improve performance by pre-fetching address translations before issuing DMA requests. Device-TLBs can be useful for devices with strict access latency requirements (such as isochronous devices), and for devices that have large DMA working sets or multiple active DMA streams. Remapping hardware units report support for Device-TLBs through the Extended Capability Register (see Section 11.4.3). Additionally, Device-TLBs may be utilized by devices to support recoverable I/O page faults. This chapter describes the basic operation of Device-TLBs. Chapter 7 covers use of Device-TLBs to support recoverable I/O page faults.

4.1 Device-TLB Operation

Device-TLB support in endpoint devices requires standardized mechanisms to:

- Request and receive translations from the Root-Complex
- Indicate if a memory request (with or without PASID) has a translated or un-translated address
- Invalidate translations cached at Device-TLBs.

Figure 4-1 illustrates the basic interaction between the Device-TLB in an endpoint and remapping hardware in the Root-Complex, as defined by Address Translation Services (ATS) in PCI Express Base Specification, Revision 4.0 or later.

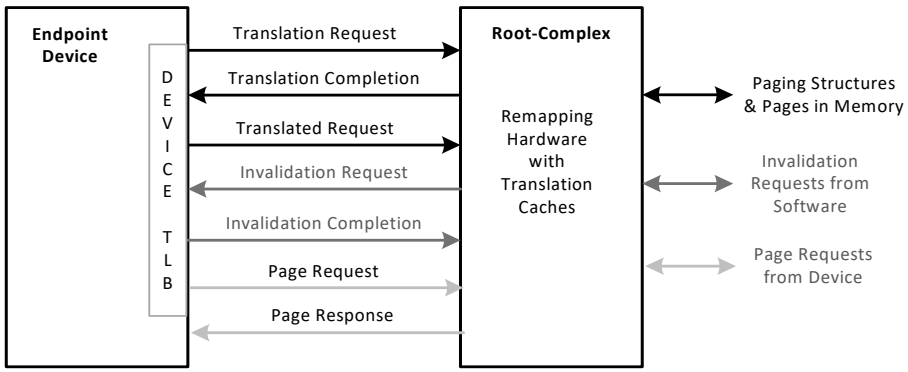


Figure 4-1. Device-TLB Operation

ATS defines the 'Address Type' (AT) field in the PCI Express transaction header for memory requests. The AT field indicates if transaction is a memory request with 'Untranslated' address (AT=00b), 'Translation Request' (AT=01b), or memory request with 'Translated' address (AT=10b). ATS also defines Device-TLB invalidation messages. Following sections describe details of these transactions.

4.1.1 Translation Request

Translation-requests-without-PASID specify the following attributes that are used by remapping hardware to process the request:

- Address Type (AT):
  - AT field has value of 01b to identify it as a translation-request.
- Address:
  - Address field indicates the starting input address for which the translation is requested.
- Length:
  - Length field indicates how many sequential translations may be returned in response to this request. Each translation is 8 bytes in length. The length field must always indicate an even number of DWORDs with a minimum value of 2 (DWs). If the length field has a value greater than 2, then the additional translations (if returned in the translation response) are for sequentially increasing equal-sized pages starting at the requested input address. Refer to ATS specification within PCI Express Base Specification Revision 4.0 or later for more details.
- No Write (NW) flag:
  - The NW flag, when Set, indicates if the endpoint is requesting read-only access for this translation.

Translation-requests-with-PASID specify the same attributes as above, and also specify the additional attributes (PASID value and Privileged-mode-Requested (PR) flag) in the PASID TLP Prefix.

4.1.2 Translation Completion

If the remapping hardware was not able to successfully process the translation-request (with or without PASID), a translation-completion without data is returned.

- A status code of UR (Unsupported Request) is returned in the completion if the remapping hardware is configured to not support translation requests from this endpoint.
- A status code of CA (Completer Abort) is returned if the remapping hardware encountered errors when processing the translation-request.

If the remapping hardware was able to successfully process a translation-request, a translation-completion with data is returned.

For successful translation-requests-without-PASID, each translation returned in the translation-completion data specifies the following attributes:

- Size (S):
  - This field is Clear if the translation applies to a 4-KByte range of memory. If this field is Set, then the translation applies to a range of memory that is larger than 4-KByte. Refer to ATS specification within PCI Express Base Specification Revision 4.0 or later for more details.
- Non-Snooped access flag (N):
  - When Set, the Non-Snooped access field indicates that the translated-requests that use this translation must clear the No Snoop Attribute in the request.
- Untranslated access only flag (U):
  - When Set, the input address range for the translation can only be accessed by the endpoint using untranslated-request.

- Read permission (R):
  - If Set, read permission is granted for the input address range of this translation. If Clear, read permission is not granted for the input address range of this translation.
- Write permission (W):
  - If Set, write permission is granted for the input address range of this translation. If Clear, write permission is not granted for the input address range of this translation.
- Translated Address:
  - If either the R or W field is Set, and the U field is Clear, the translated address field contains the result of the translation for the respective input address. Endpoints can access the page through Translated-requests with this address.

For successful translation-requests-with-PASID, each translation returned in the translation-completion data specifies the same attributes as above, along with following attributes:

- Execute permission (EXE):
  - Remapping hardware provides a value of 0 in this field.
- Privilege Mode Access (PRIV):
  - If Set, R, W and EXE refer to permissions associated with privileged mode access. If Clear, R, W, and EXE refer to permissions associated with non-privileged access.
- Global Mapping (G):
  - If Set, the translation is common across all PASIDs at this endpoint. If Clear, the translation is specific to the PASID value specified in the PASID TLP Prefix in the associated Translation-request.
  - Remapping hardware provides a value of 0 in this field.

4.1.3 Translated Request

Translated-requests are regular memory read/write/atomics requests with Address Type (AT) field value of 10b. When generating a request to a given input (untranslated) address, the endpoint may look in the local Device-TLB for a cached translation (result of a previous translation-request) for the input address. If a cached translation is found with appropriate permissions and privilege, the endpoint may generate a translated-request (AT=10b) specifying the Translated address obtained from the Device-TLB lookup. Translated-requests can be issued as requests-without-PASID or requests-with-PASID.

4.1.4 Invalidation Request and Completion

Invalidation requests are issued by software through remapping hardware to invalidate translations cached at endpoint Device-TLBs.

Invalidation-requests-without-PASID specify the following attributes:

- Device ID
  - Identity of the device (bus/device/function) whose Device-TLB is the target of invalidation.
- Size (S):
  - When this field is Clear, the target region to invalidate is 4-KByte. When this field is Set, the target region to invalidate is greater than 4-KByte. Refer to the ATS specification within PCI Express Base Specification Revision 4.0 or later for more details.
- Untranslated Address
  - Specifies the base of the input (untranslated) address range to be invalidated.

Invalidation-requests-with-PASID specify the same attributes as above, along with a global-invalidate flag. If the global-invalidate flag is 1, the invalidation affects all PASID values.

Invalidation requests and completions carry additional tags (ITags) managed by hardware to uniquely identify invalidation requests and completions. Refer to the Address Translation Services in PCI Express Base Specification Revision 4.0 or later for more details on use of ITags.

4.2 Remapping Hardware Handling of Device-TLBs

Remapping hardware reports support for Device-TLBs through the Extended Capability Register (see Section 11.4.3). The translation-type (TT) field in the context-entries and Device TLB Enable (DTE) field in scalable-mode context entries can be programmed to enable or disable processing of translation-requests and translated-requests from specific endpoints by remapping hardware. The following sections describe the remapping hardware handling of ATS requests.

4.2.1 Handling of ATS Protocol Errors

The following upstream requests are always handled as Unsupported Request (UR) by hardware:

- Memory read or write request (with or without PASID) with AT field value of 'Reserved' (11b).
- Memory write request (with or without PASID) with AT field value of 'Translation Request' (01b).
- Requests-with-PASID with AT field value of 'Translated' (10b).

The following upstream requests (with or without PASID) are always handled as malformed packets:

- Memory read request with AT field value of 'Translation Request' with any of the following:
  - — Length specifying odd number of DWORDs (i.e. least significant bit of length field is non-zero)
  - — Length greater than N/4 DWORDs where N is the Read Completion Boundary (RCB) value (in bytes) supported by the Root-Complex.
  - — First and last DWORD byte enable (BE) fields not equal to 1111b.
- 'Invalidation Request' message.

When remapping hardware is disabled (TES=0 in Global Status Register), following upstream requests are treated as Unsupported Request (UR).

- Memory requests with non-zero AT field (i.e. AT field is not 'Untranslated').
- ATS 'Invalidation Completion' messages.

4.2.2 Root-Port Control of ATS Address Types

Root-ports supporting Access Control Services (ACS) capability can support 'Translation Blocking' control to block upstream memory requests with non-zero value in the AT field. When enabled, such requests are reported as ACS violation by the receiving root-port. Refer to the ACS Capability in PCI Express Specifications for more details. Upstream requests that cause ACS violations are blocked at the root-port as error and are not presented to remapping hardware.

4.2.3 Handling of Translation Requests

This section describes the handling of translation-requests when remapping hardware is enabled.

- The requester-id in the translation-request is used to parse the respective legacy root/context entry or scalable-mode root/context entry as described in Section 3.4.
- Table 30 describes all the error conditions under which remapping hardware explicitly blocks a translation request (with or without PASID) with a status code of Unsupported Request (UR) or Completer Abort (CA).
- Additionally, Table 30 also describes all the error conditions under which remapping hardware provides a translation-completion with status code of Success with R=W=U=S=0 in the translation-completion-data.

- If none of the error conditions are encountered, hardware handles the translation-request as follows:
  - If the input address in a translation-request-without-PASID is within the interrupt address range (FEEx_xxxxh)[1], such requests are treated as a special condition. Refer to S.1, S.2, and S.3 in Table 30 for details.
  - If remapping hardware successfully fetches the translation requested, and the translation has at least one of Read and Write permissions, a translation-completion with status code of Success is returned with translation-completion-data as follows:
    - **Read (R) bit**: The R bit in the translation-completion data is the effective read permission for this translation.
      - For translation-requests, R bit is 1 if the respective access rights checking (as described in Section 3.6.1, Section 3.7.1, and Section 3.8.1) allows read access to the page. Else, R bit is 0.
    - **Write (W) bit**: The W bit in the translation-completion data is the effective write permission for this translation.
      - For translation-requests with NW=1 (i.e., requests indicating translation is for read-only accesses), remapping hardware reporting no-write-flag support (NWFS=1 in the Extended Capability Register) returns the W bit as 0. Remapping hardware not supporting no-write-flag (NWFS=0) ignores value of NW field in translation-requests and functions as if NW is 0 (see below).
    - For translation-requests with NW=0, W bit is 1 if the respective access rights checking (as described in Section 3.6.1, Section 3.7.1 and Section 3.8.1) allows write access to the page. Else, W bit is 0.
    - **Execute (EXE) bit**: The EXE bit in the translation-completion data is the effective execute permission for this translation.
      - This bit is always 0 as implementations do not support requests with a value of 1 for Execute-Requested (ER).
    - **Privilege Mode (PRIV) bit**: The PRIV bit in the translation-completion data is the effective privilege for this translation.
      - For translation-request-with-PASID with PR=1, this bit is always 1. For all other requests, this bit is always 0.
      - For translation-request-with-PASID with PR=1, remapping hardware not supporting supervisor requests (SRS=0 in the Extended Capability Register) forces R=W=E=0 in addition to setting PRIV=1.
      - If the scalable-mode PASID-table entry used to process translation-request-with-PASID with PR=1 has a value of 0 in the SRE field, the remapping hardware forces R=W=E=0 in addition to setting PRIV=1.
      - Note: A translation-request-without-PASID that observes a value of 1 in RID_PRIV field of scalable-mode context-entry, returns R and W permissions associated with Privileged Mode Entities in the translation completion but with Priv field set to a value of 0.
    - **Global Mapping (G) bit**: The G bit in the translation-completion data is the effective privilege for this translation.
      - This bit is set to 0.
    - **Non-snooped access (N) bit**: The N bit in the translation-completion data indicates the use of the No-Snoop (NS) flag in accesses that use this translation.
      - Remapping hardware setup in legacy mode:

1. Translation-requests-with-PASID with input address in the range FEEx_xxxxh are processed normally through page-table translation, like any other translation-request-with-PASID.

- For requests that use second-stage translation with ATS allowed (TT=01b), remapping hardware supporting Snoop Control (ECAP_REG.SC=1) returns the SNP bit in the leaf second-stage paging structure entry controlling the translation as the N bit. Remapping hardware not supporting Snoop Control returns a value of 0 as the N bit.

— Remapping hardware setup in scalable mode:

- For requests that use first-stage or pass-through translation (PGTT=001b or PGTT=100b), remapping hardware returns the PGSNP bit in the scalable-mode PASID-table entry as the N bit.
- For requests that use second-stage or nested translation (PGTT=010b or PGTT=011b), remapping hardware returns a value of 1 as the N bit, if either the PGSNP field in the scalable-mode PASID-table entry is set, or the SNP field in the leaf second-stage paging structure entry controlling the translation are set. Otherwise, remapping hardware returns a value of 0 as the N bit.

Table 10. N Bit in Translation Completion

Mode	TT (Legacy) or PGTT (Scalable)	Page: Translation Completion.N (Non-snooped Access)
Legacy	TT=00b (second-stage with ATS blocked)	NA (Translation request are treated as UR)
	TT=01b (second-stage with ATS allowed)	N=SS.leaf.SNP
	TT=10 (pass-through)	NA (Translation request are treated as UR)
	TT=11 (reserved)	NA (Translation request are treated as UR)
Scalable	PGTT=001b (first-stage)	N=PASID-table-entry.PGSNP
	PGTT=010b (second-stage)	N=PASID-table-entry.PGSNP || SS.leaf.SNP
	PGTT=011b (nested)	
	PGTT=100b (pass-through)	N=PASID-table-entry.PGSNP
	PGTT=Other (reserved)	NA (Translation request are treated as UR)

- **Untranslated access only (U) bit**: The U bit in the translation-completion data indicates the address range for the translation can only be accessed using untranslated-requests.
  — Refer to S.1 in Table 30 for special conditions when remapping hardware will set this bit to 1.
  — For all other translation requests remapping hardware will set this bit to 0.
- **Size (S) bit**: The S bit in the translation-completion data indicates the page size for the translation.
  — This bit is 0 if translation returned is for 4-KByte page.

— This bit is 1 if translation returned if for page larger than 4-KByte. In this case, the size of the translation is determined by the lowest bit in the Translated Address field (bits 63:12) with a value of 0. For example, if bit 12 is 0, the translation applies to a 8-KByte page. If bit 12 is 1 and bit 13 is 0, the translation applies to a 16-KByte page, and so on. Refer to Address Translation Services in PCI Express Base Specification Revision 4.0 or later for details on translation size encoding.

— For translation requests that encounter pass-through translation (PGTT=100b) in scalable mode, hardware is strongly recommended to set this bit to a value of 1 and use the ADDR field to provide as large a region of translation as possible.

- **Translated Address (ADDR)**: If either the R or the W bit is 1, and the U bit is 0, the ADDR field in the translation-completion data contains the result of the translation.

  — For translation-requests that are subject to second-stage translation only, this is the translated address from the second-stage translation.

  — For translation-requests that are subject to first-stage translation only, this is the output address from the first-stage translation.

  — For translation-requests that are subject to nested translation, this is the output address from the nested translation.

  — For translation-requests that are subject to pass-through translation, this is the base address of a naturally aligned region that is equal in size to the largest page-size supported by remapping hardware and that contains the address that came in the translation-request.

4.2.3.1 Accessed, Extended Accessed, and Dirty Flags

Accessed, Extended Accessed, and Dirty flags are set at the time of processing a translation-request before a translation-completion is returned to the endpoint. Such flags may not be reflected accurately in limited cases (i.e., an endpoint issues a translation-request with intent for read-only access, but later incorrectly utilizes this translation to issue a translated-request with write access). Remapping hardware reporting No-Write-Flag Support (NWFS=1 in the Extended Capability Register) treats a translation-request with NW=0 as an atomic (i.e., read + write) and a translation-request with NW=1 as a read. Remapping hardware not supporting No-Write-Flag (NWFS=0) ignores the value of the NW field in the translation-request and treats it as an atomic (i.e,. read + write). Setting of Accessed, Extended Accessed, and Dirty flags follow the rules described in Section 3.6.2, Section 3.7.2, or Section 3.8.1, depending on the PGTT field value encountered by the translation-request.

4.2.3.2 Translation Requests for Multiple Translations

Translation-requests for multiple mappings indicate a length field greater than 2 DWORDs. Hardware implementations may handle these requests in any one of the following ways:

- Always return a single translation

  — Hardware performs translation only for the starting address specified in the translation-request, and a translation-completion is returned depending on the result of this processing. In this case, the translation-completion has a Length of 2 DWORDs, Byte Count of 8, and the Lower Address indicates a value of Read Completion Boundary (RCB) minus 8.

- Return multiple translations

  — Hardware performs translations starting with the address specified in the translation-request, until a Completer Abort (CA) or Unsupported Request (UR) condition as described in Section 4.2.3 is detected, or until a translation with different page-size than the previous translation in this request is detected. Remapping hardware may also limit fetching of translations to those that are resident within a cache line. When returning multiple translations (which may be less than the number of translations requested), hardware must ensure that successive translations must apply to the untranslated address range that abuts the previous translation in the same completion. Refer to Address Translation Services in PCI

Express Base Specification Revision 4.0 or later for requirements on translation-completions returning multiple mappings in one or two packets.

4.2.4 Host Permission Table

The Host Permission Table (HPT) allows software to control access to the physical address space when one or more devices uses translated requests. System software may create one or more domains which permit memory accesses from only those ATS-devices that are accepted as part of the domain's Trusted Computing Base (TCB). HPT is only supported when operating in scalable mode. The HPT is a 4-level structure described in Table 11 and an example HPT Walk in Figure 4-2. Each table entry corresponding to a valid page size can reference one or more valid pages using the Page Permission (PP*i*) fields as well as providing a pointer to the subsequent paging structure.

Table 11. Host Permission Table Structures

Paging Structure	Entry Name	Physical Address of Structure	Bits Selecting Entry	Bits Selecting PP*i*	Page Mapping
HPTL4 Table	HPTL4E	Scalable-mode PASID-table entry	HAW-1:41	N/A	N/A
HPTL3 Table	HPTL3E	HPTL4E	40:33	32:30	1-GByte Page (if PP*i*.R or PP*i*.W is Set)
HPTL2 Table	HPTL2E	HPTL3E	32:25	24:21	2-MByte Page (if PP*i*.R or PP*i*.W is Set)
HPTL1 Table	HPTL1E	HPTL2E	24:17	16:12	4-KByte Page

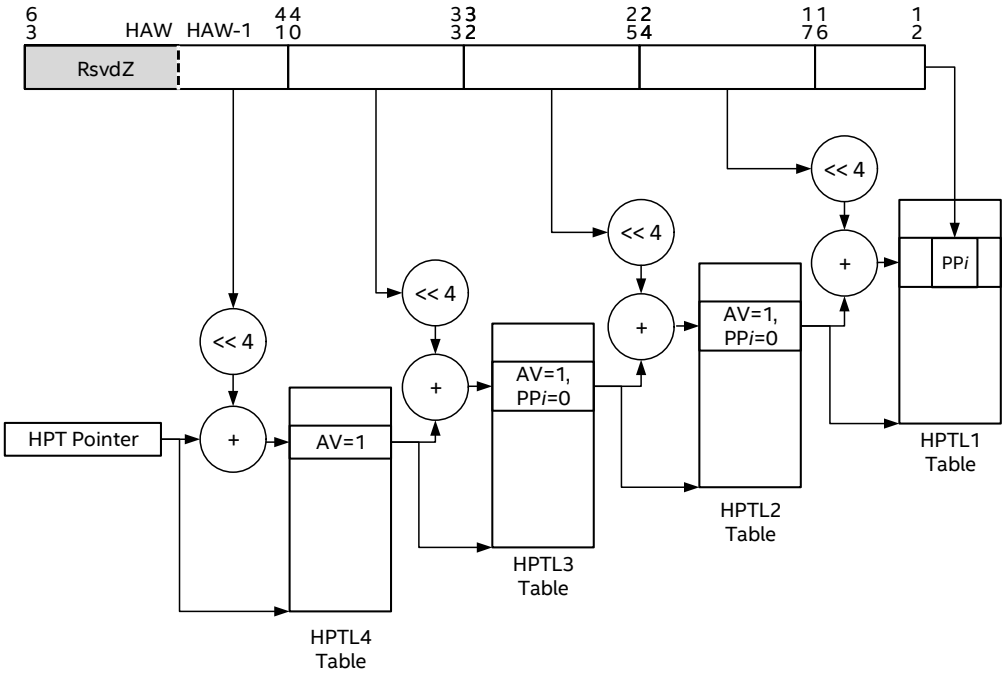


Figure 4-2. HPT Walk for Page Permissions of 4K Page

The following describe the Host Permission Table in more detail, how an HPT walk is performed, and how the page size is determined:

- When HPT Enable is Set in the scalable-mode context entry, remapping hardware locates a scalable-mode PASID table entry as defined in Section 3.4.3. A 4-KByte aligned HPTL4 table is located at the physical address specified in the HPT Root Pointer (HPTPTR) field in the scalable-mode PASID-table entry (see Section 9.6). An HPTL4 table comprises 128-bit entries (HPTL4Es). The size of the HPTL4 table is specified by the HPT Size (HPTSZ) field in the scalable-mode PASID-table entry. A HPTL4E is selected using the physical address defined as follows:
  - Bits 3:0 are all 0.
  - Bits 11:4 are bits 48:41 of the input address.
  - Bits 64:12 is the sum of bits 51:49 of the input address and the HPT Root Pointer in the scalable-mode PASID-table entry.
  - Implementations treat bits 51:HAW in the input address as RsvdZ.

  Because a HPTL4E is identified using bits HAW-1:41 of the input address, it controls access to a 2-TByte region of the input-address space.

  If Address valid is Clear, the HPT Walk is terminated.

- A 4-KByte aligned HPTL3 table is located at the physical address specified in the address (ADDR) field in the HPTL4E (see Section 9.12.1). A HPTL3 table comprises 256 128-bit entries (HPTL3Es). An HPTL3E is selected using the physical address defined as follows:
  - Bits 3:0 are all 0.
  - Bits 11:4 are bits 40:33 of the input address.
  - Bits 12 and higher are from the ADDR field in the HPTL4E.

  Because a HPTL3E is identified using bits HAW-1:33 of the input address, it controls access to a 8-GByte region of the input-address space.

- An HPTL3E has 8 PP*i* fields that each provide permissions for a 1-GByte region. The PP*i* is selected using bits 32:30 of the input address.
  - If PP*i* is a non-zero value, then the input address corresponds to a 1-GByte page, the permission of that page is determined by PP*i*, and the HPT Walk is complete.
  - If PP*i* is zero:
    - If Address valid is Clear, the HPT walk is terminated.
    - If Address Valid is Set, the HPT walk continues to the HTPL2 table.
- A 4-KByte aligned HPTL2 table is located at the physical address specified in the address (ADDR) field in the HPTL3E (see Section 9.12.2). A HPTL2 table comprises 256 128-bit entries (HPTL2Es). An HPTL2E is selected using the physical address defined as follows:
  - Bits 3:0 are all 0.
  - Bits 11:4 are bits 32:25 of the input address.
  - Bits 12 and higher are from the ADDR field in the HPTL3E.

  Because a HPTL2E is identified using bits HAW-1:25 of the input address, it controls access to a 32-MByte region of the input-address space.

- An HPTL2E has 16 PP*i* fields that each provide permissions for a 2-MByte region. The PP*i* is selected using bits 24:21 of the input address.
  - If PP*i* is a non-zero value, then the input address corresponds to a 2-MByte page, and the permission of that page is determined by PP*i*, and the HPT Walk is complete.
  - If PP*i* is zero:
    - If Address valid is Clear, the HPT walk is terminated.
    - If Address Valid is Set, the HPT walk continues to the HTPL1 table.

- A 4-KByte aligned HPTL1 table is located at the physical address specified in the address (ADDR) field in the HPTL2E (see Section 9.12.3). A HPTL1 table comprises 256 128-bit entries (HPTL1Es). An HPTL1E is selected using the physical address defined as follows:
  - Bits 3:0 are all 0.
  - Bits 11:4 are bits 24:17 of the input address.
  - Bits 12 and higher are from the ADDR field in the HPTL2E.

  Because an HPTL1E is identified using bits HAW-1:17 of the input address, it controls access to a 128-KByte region of the input-address space.
- An HPTL1E has 32 PP*i* fields that each provide permissions for a 4-KByte region. The PP*i* is selected using bits 16:12 of the input address. All input addresses resulting in access to an HPTL1E corresponds to a 4-KByte page, and the permission of that page is determined by PP*i*.

4.2.4.1 Access Rights

Translated requests can result in an HPT faults for either of two reasons: (1) an HPT walk is terminated before a non-zero PP*i* field is found for the request or (2) the HPT walk resulted in a PP*i* field with inadequate permissions for the request. Section 7.1.3 provides detailed hardware behavior on HPT faults and reporting to software.

When a PP*i* with a non-zero value is discovered in any of the HPTL3E, HPTL2E, or HPTL1E associated with the an input address, the permissions are used to determine the access rights for that request. See Table 12 for required permissions per request type and Table 48 for definition of PP*i* bit fields.

Table 12. Required Permissions for Request Types

Request Type	Required Permissions
Memory Read	R
Memory Write	W
Atomics	R and W

4.2.4.2 Snoop Behavior

The snoop behavior for memory accesses to the HPT specifies if the access is coherent (snoops the processor caches) or not. The snoop behavior is independent of the memory type described in Section 4.2.4.3. When the Page-Walk Snoop field in the scalable-mode PASID-table entry is Set, all accesses to HPT entries will snoop the processor caches. If Page-Walk Snoop is Clear, hardware is not required to snoop the processor caches when accessing HPT entries.

Remapping hardware may cache information from the HPT entries in translation caches. Refer to Chapter 6 for details on hardware translation caching and how software can enforce consistency with translation caches when modifying HPT structures in memory.

4.2.4.3 Memory Type

Accesses from hardware to HPT entries use memory-type of write-back (WB).

4.2.5 Handling of Translated Requests

This section describes the handling of Translated-requests on PCI/CXL links, when remapping hardware is enabled.

The Host Permission Table (HPT) allows software to control access to physical address space by a device using Translated Requests. HPT is only available in scalable mode. If HPT Enable is Set in the scalable-mode context entry, remapping hardware performs additional steps to verify that the device has required permissions to access the address in the translated request. If HPT Enable is Clear, the HPT-related steps are skipped.

- PCI / CXL.io link
  - Translated-requests received with a PASID TLP Prefix on hardware that does not support PASID in Translated Request are blocked by the root complex. (Hardware reports PASID in Translated Request as not supported in the Extended Capability Register.)
  - If a translated-request is to the interrupt address range (FEEx_xxxxh), it is blocked by remapping hardware and treated as a fault LGN.4/SGN.8. (See Section 7.1.3 for details of conditions and hardware behavior.)
  - If hardware detects any error condition that requires remapping hardware to block a translated-request, it is treated as a fault and the translated-request is handled as UR or CA. (See Section 7.1.3 for details of error conditions.)
  - If HPT Enable is Clear and remapping hardware has successfully processed the associated context entry or scalable-mode context entry without detecting any of the above error conditions, remapping hardware bypasses address translation and sets the output address for the translated-request equal to the input address. Remapping hardware assigns write-back (WB) memory type to all such translated requests.
  - If HPT Enable is Set and remapping hardware has successfully completed the HPT walk without detecting any of the above error conditions, remapping hardware bypasses address translation and sets the output address for the translated-request equal to the input address. Remapping hardware assigns write-back (WB) memory type to all such translated requests.
- CXL.cache link
  - Translated-request is allowed to access memory without any checks by remapping hardware.

4.3 Handling of Device-TLB Invalidations

The Address Translation Services (ATS) support in PCI Express defines the wire protocol for the Root-Complex to issue Device-TLB invalidation requests to an endpoint and to receive Device-TLB invalidation completion responses from the endpoint.

For remapping hardware supporting Device-TLBs, software submits the Device-TLB invalidation requests through the invalidation queue interface of the remapping hardware. Section 6.5.2 describes the queued invalidation interface details. Software is recommended to not submit any Device-TLB invalidation requests while address remapping hardware is disabled (TES=0 in Global Status Register).

Hardware processes a Device-TLB invalidation request as follows:

- Hardware allocates a free invalidation tag (ITag). ITags are used to uniquely identify an invalidation request issued to an endpoint. If there are no free ITags in hardware, the Device-TLB invalidation request is deferred until a free ITag is available. For each allocated ITag, hardware stores a counter (*InvCmpCnt*) to track the number of invalidation completions received with this ITag.
- Hardware starts an invalidation completion timer for this ITag, and issues the invalidation request message to the specified endpoint. If the invalidation command from software is for a translation with PASID, the invalidation request message is generated with the appropriate PASID TLP Prefix to identify the target PASID. The invalidation completion time-out value is recommended to be sufficiently larger than the PCI Express read completion time-outs.

Hardware processes a Device-TLB invalidation response received as follows:

- ITag-vector in the invalidation completion response indicates the ITags corresponding to completed Device-TLB invalidation requests. The completion count in the invalidation response

indicates the number of invalidation completion messages expected with the same ITag-vector and completion count.

- For each ITag Set in the ITag-vector, hardware checks if it is a valid (currently allocated) ITag for the source-id in the invalidation completion response. For each valid ITag, the corresponding InvCmpCnt counter is incremented and compared with the 'completion count' value in the invalidation response ('completion count' value of 0 indicates 8 invalidation completions). If the comparison matches, the Device-TLB invalidation request corresponding to the ITag is considered completed, and the ITag is freed.
- After processing all the valid ITags as described above, hardware processes invalid ITags in the ITag-vector. If an invalid ITag is detected, hardware reports an error condition by setting the Invalidation Completion Error (ICE) field in the Fault Status Register (see Section 11.4.7.1); and depending on the programming of the Fault Control Register a fault event may also be generated. Hardware implementations of this architecture with Major Version 5 or lower may detect an invalid ITag before processing all the valid ITags and hence report the Invalidation Completion Error (ICE) without processing some or all of the valid ITags.
- If the invalidation completion time-out expires for an ITag before the *InvCmpCnt* invalidation responses are received, hardware frees the ITag and reports it through the ITE field in the Fault Status Register. Depending on the programming of the Fault Control Register a fault event may be generated. Section 6.5.2.11 describes hardware behavior on invalidation completion time-outs.

4.4 Device TLB in System-on-Chip (SoC) Integrated Devices

The Figure 4-3 below shows a typical SoC integrated device that is connected to the Root Complex. All such devices that support Address Translation Service (ATS) must create an isolation boundary outside of which Host Physical Address (HPA) are not exposed. The part of the device outside the isolation boundary is referred to as "Device Core".

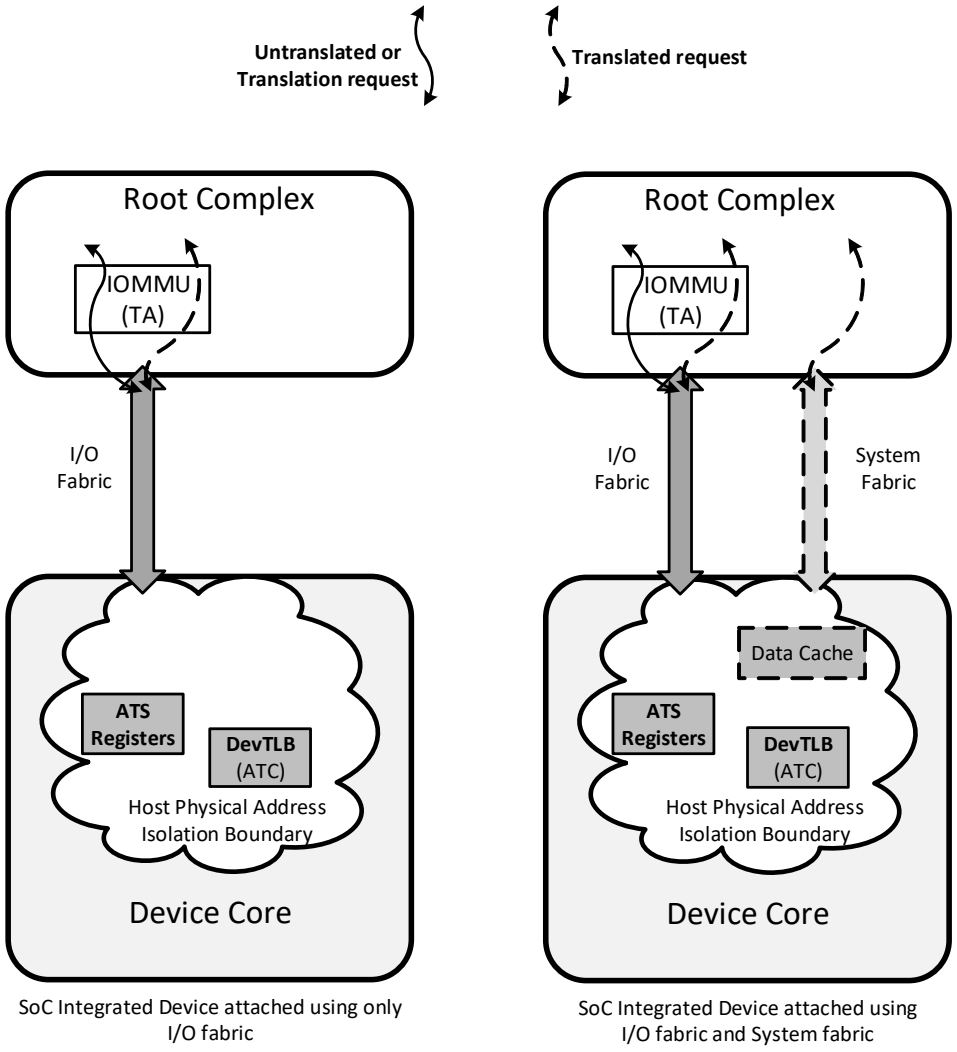


Figure 4-3. DevTLB in SoC Integrated Devices

Depending on the software usage model the Device Core may receive various kinds of address from software. These addresses are translated by Device TLB (i.e. DevTLB) into HPAs using ATS.

All SoC integrated devices must meet the following requirements:

- The HPA isolation boundary must not expose translated addresses (from DevTLB) to the Device Core
- The interface(s) that connect the SoC integrated device to the Root Complex must be completely contained within the HPA isolation boundary.
- The Device Core must not be able to put a transaction on the interface(s) that connects device to the Root Complex. The Device Core must always request that the HPA isolation partition send transaction to the Root Complex.

- The HPA isolation partition must host the DevTLB and must faithfully follow the ATS specification.
- The HPA isolation partition must host ATS Configuration registers defined by PCI Express Base Specification Revision 4.0 or later and listed below:
  - ATS Extended Capability
  - ATS Extended Capability Header
  - ATS Capability Register
  - ATS Control Register
- The Device Core can specify that a transaction bypass the DevTLB; in that case, the HPA isolation partition will put the transaction as an "Untranslated Request" on the I/O fabric.
- The Device Core (including software/firmware running on the device) must not be able to manipulate (read or modify) the contents of DevTLB or the data cache.
- If a device needs a data cache tagged with translated addresses (from DevTLB), such a cache must be hosted in the HPA isolation boundary.

SoC-integrated devices connecting to the Root Complex using the System Fabric must meet the following additional requirements:

- Before sending an ATS Invalidate Completion message to the IOMMU, the HPA isolation partition must ensure that all transactions using stale information on the System Fabric have reached the global observation point.
- When DMA remapping is enabled in the IOMMU, any access on the System Fabric from the HPA isolation partition must use HPA.
- The Root Complex may implement a product specific mechanism to inform the HPA isolation partition in each SoC integrated device that DMA remapping is enabled.

4.5 Guidance to Software on Enabling and Disabling ATS

Enabling ATS for a device requires software to program two independent bits and hence it can not be an atomic operation and there will be a window of time where the system is in an inconsistent state. It is strongly recommended that software quiesce DMA operations from the device before programming the required bits.

4.5.1 Recommended Software Sequence to Enable ATS

1. Software should quiesce all DMA operations from the device.
2. Software should program the scalable-mode context entry (or context entry) associated with the DevTLB to allow ATS requests.
3. Software should program Enable (E) field, in the ATS Control register of the device with a value of 1.

4.5.2 Recommended Software Sequence to Disable ATS

1. Software should quiesce all DMA operations from the device.
2. Software should program Enable (E) field, in the ATS Control register of the device with a value of 0.
3. Software should issue global Device-TLB Invalidate descriptor followed by Invalidation Wait descriptor to invalidate all translations from the DevTLB and also to drain all prior ATS traffic to the global observation point.
4. Software should program the scalable-mode context entry (or context entry) associated with the DevTLB to block ATS requests.

5 Interrupt Remapping

This chapter discusses architecture and hardware details for interrupt-remapping and interrupt-posting. These features collectively are called the interrupt remapping architecture.

5.1 Interrupt Remapping

The interrupt-remapping architecture enables system software to control and censor external interrupt requests generated by all sources including those from interrupt controllers (I/OxAPICs), MSI/MSI-X capable devices including endpoints, root-ports and Root-Complex integrated end-points.

Interrupts generated by the remapping hardware itself (Fault Events, Invalidation Completion Events, and Page Request Events) are not subject to interrupt remapping.

Interrupt requests appear to the Root-Complex as upstream DWORD sized memory write requests to the interrupt address range FEEx_xxxxh. Since interrupt requests arrive at the Root-Complex as write requests, interrupt-remapping is co-located with the remapping hardware units. The interrupt-remapping capability is reported through the Extended Capability Register.

5.1.1 Identifying Origination of Interrupt Requests

To support domain-isolation usages, the platform hardware must be capable of uniquely identifying the requester (Source-Id) for each interrupt message. The interrupt sources in a platform and use of source-id in these requests may be categorized as follows:

- Message Signaled Interrupts from PCI Express Devices
  - For message-signaled interrupt requests from PCI Express devices, the source-id is the requester identifier in the PCI Express transaction header. The requester-id of a device is composed of its PCI Bus/Device/Function number assigned by configuration software and uniquely identifies the hardware function that initiated the I/O request. Section 3.4.1 illustrates the requester-id as defined by the PCI Express specification. Section 3.12.5 describes use of source-id field by PCI Express devices using phantom functions.
- Message Signaled Interrupts from Root-Complex Integrated Devices
  - For message-signaled interrupt requests from root-complex integrated PCI or PCI Express devices, the source-id is its PCI requester-id.
- Message Signaled Interrupts from Devices behind PCI Express to PCI/PCI-X Bridges
  - For message-signaled interrupt requests from devices behind PCI Express-to-PCI/PCI-X bridges, the requester identifier in those interrupt requests may be that of the interrupting device or the requester-id with the bus number field equal to the bridge's secondary interface's bus number and device and function number fields value of zero. Section 3.12.1 describes legacy behavior of these bridges. Due to this aliasing, interrupt-remapping hardware does not isolate interrupts from individual devices behind such bridges.
- Message Signaled Interrupts from Devices behind Conventional PCI bridges
  - For message-signaled interrupt requests from devices behind conventional PCI bridges, the source-id in those interrupt requests is the requester-id of the legacy bridge device. Section 3.12.2 describes legacy behavior of these bridges. Due to this, interrupt-remapping

hardware does not isolate message-signaled interrupt requests from individual devices behind such bridges.

- Legacy pin interrupts
  - For devices that use legacy methods for interrupt routing (such as either through direct wiring to the I/OxAPIC input pins, or through INTx messages), the I/OxAPIC hardware generates the interrupt-request transaction. To identify the source of interrupt requests generated by I/OxAPICs, the interrupt-remapping hardware requires each I/OxAPIC in the platform (enumerated through the ACPI Multiple APIC Descriptor Tables (MADT)) to include a unique 16-bit source-id in its requests. BIOS reports the source-id for these I/OxAPICs via ACPI structures to system software. Refer to Section 8.3.1.1 for more details on I/OxAPIC identity reporting.
- Other Message Signaled Interrupts
  - For any other platform devices that are not PCI discoverable and yet capable of generating message-signaled interrupt requests (such as the integrated High Precision Event Timer - HPET devices), the platform must assign unique source-ids that do not conflict with any other source-ids on the platform. BIOS must report the 16-bit source-id for these via ACPI structures described in Section 8.3.1.2.

5.1.2 Interrupt Request Formats On Intel® 64 Platforms

Interrupt-remapping on Intel® 64 platforms support two interrupt request formats. These are described in the following sub-sections.

5.1.2.1 Interrupt Requests in Compatibility Format

Figure 5-1 illustrates the interrupt request in Compatibility format. The Interrupt Format field (Address bit 4) is Clear in Compatibility format requests. Refer to the Intel® 64 Architecture software developer's manuals for details on other fields in the Compatibility format interrupt requests. Platforms without interrupt-remapping capability support only Compatibility format interrupts.

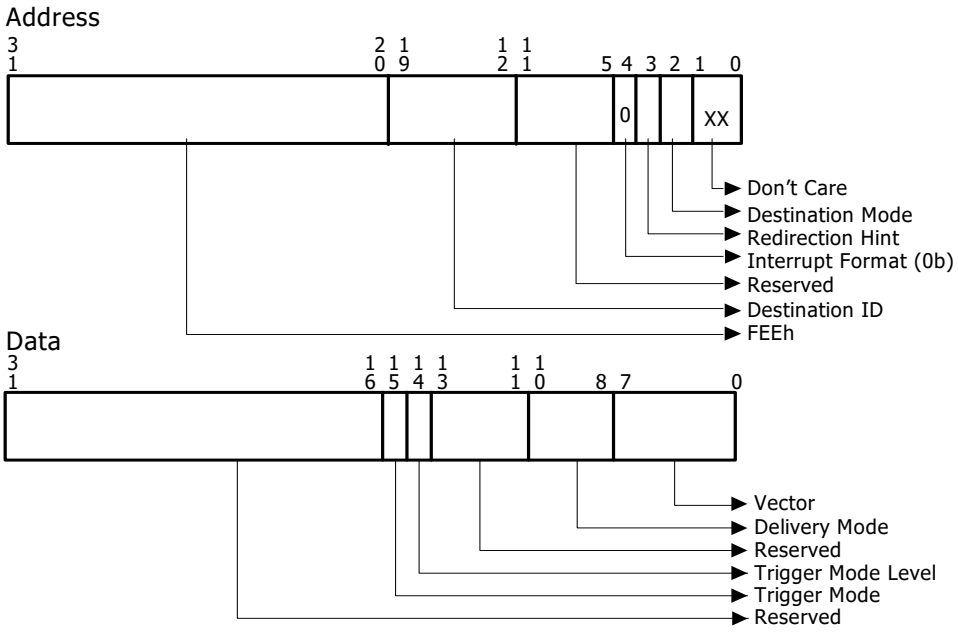


Figure 5-1. Compatibility Format Interrupt Request

5.1.2.2 Interrupt Requests in Remappable Format

Figure 5-2 illustrates the Remappable interrupt request format. The Interrupt Format field (Address bit 4) is Set for Remappable format interrupt requests. Remappable interrupt requests are applicable only on platforms with interrupt-remapping support.

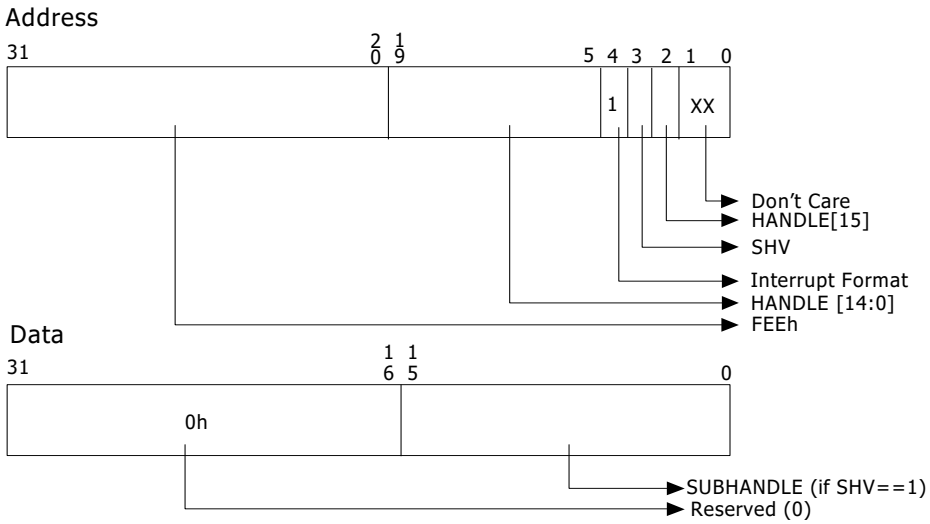


Figure 5-2. Remappable Format Interrupt Request

Table 13 describes the various address fields in the Remappable interrupt request format.

Table 13. Address Fields in Remappable Interrupt Request Format

Address Bits	Field	Description
31: 20	Interrupt Identifier	DWORD DMA write request with value of FEEh in these bits are decoded as interrupt requests by the Root-Complex.
19: 5	Handle[14:0]	This field along with bit 2 provides a 16-bit Handle. The Handle is used by interrupt-remapping hardware to identify the interrupt request. 16-bit Handle provides 64K unique interrupt requests per interrupt-remapping hardware unit.
4	Interrupt Format	This field must have a value of 1b for Remappable format interrupts.
3	SubHandle Valid (SHV)	This field specifies if the interrupt request payload (data) contains a valid Subhandle. Use of Subhandle enables MSI constructs that supports only a single address and multiple data values.
2	Handle[15]	This field carries the most significant bit of the 16-bit Handle.
1:0	Don't Care	These bits are ignored by interrupt-remapping hardware.

Table 14 describes the various data fields in the Remappable interrupt request format.

Table 14. Data Fields in Remappable Interrupt Request Format

Data Bits	Field	Description
31:16	Reserved	When SHV field in the interrupt request address is Set, this field treated as reserved (0) by hardware. When SHV field in the interrupt request address is Clear, this field is ignored by hardware.
15:0	Subhandle	When SHV field in the interrupt request address is Set, this field contains the 16-bit Subhandle. When SHV field in the interrupt request address is Clear, this field is ignored by hardware.

5.1.3 Interrupt Remapping Table

Interrupt-remapping hardware utilizes a memory-resident single-level table, called the Interrupt Remapping Table. The interrupt remapping table is expected to be setup by system software, and its base address and size is specified through the Interrupt Remap Table Address Register. Each entry in the table is 128-bits in size and is referred to as Interrupt Remapping Table Entry (IRTE).Section 9.9 illustrates the IRTE format.

For interrupt requests in Remappable format, the interrupt-remapping hardware computes the 'interrupt_index' as below. The Handle, SHV and Subhandle are respective fields from the interrupt address and data per the Remappable interrupt format.

```
if (address.SHV == 0) {
    interrupt_index = address.handle;
} else {
    interrupt_index = (address.handle + data.subhandle);
}
```

The Interrupt Remap Table Address Register is programmed by software to specify the number of IRTEs in the Interrupt Remapping Table (maximum number of IRTEs in an Interrupt Remapping Table is 64K). Remapping hardware units in the platform may be configured to share interrupt-remapping table or use independent tables. The interrupt_index is used to index the appropriate IRTE in the interrupt-remapping table. If the interrupt_index value computed is equal to or larger than the number of IRTEs in the remapping table, hardware treats the interrupt request as error.

Unlike the Compatibility interrupt format where all the interrupt attributes are encoded in the interrupt request address/data, the Remappable interrupt format specifies only the fields needed to compute the interrupt_index. The attributes of the remapped interrupt request is specified through the IRTE referenced by the interrupt_index.The interrupt-remapping architecture defines support for hardware to cache frequently used IRTEs for improved performance. For usages where software may need to dynamically update the IRTE, architecture defines commands to invalidate the IEC. Chapter 6 describes the caching constructs and associated invalidation commands.

5.1.4 Interrupt-Remapping Hardware Operation

The following provides a functional overview of the interrupt-remapping hardware operation:

- An interrupt request is identified by hardware as a DWORD sized write request to interrupt address ranges FEEx_xxxxh.
- When interrupt-remapping is not enabled (IRES field Clear in Global Status Register):
  - If Interrupt Remapping Required is reported as Set (ECAP_REG.IRREQ=1), the interrupt request is blocked.
  - If Interrupt Remapping Required is reported as Clear (ECAP_REG.IRREQ=0), all interrupt requests are processed per the Compatibility interrupt request format described in Section 5.1.2.1.
- When interrupt-remapping is enabled (IRES field Set in Global Status Register), interrupt requests are processed as follows:

— If Extended Interrupt Mode Enable Required (EIMER) is reported as Set through the Extended Capability Register and Extended Interrupt Mode is disabled (EIME field in Interrupt Remapping Table Address Register is Clear), all interrupt requests are blocked.

— Interrupt requests in Compatibility format (i.e., requests with Interrupt Format field Clear) are processed as follows:

- If Extended Interrupt Mode is enabled (EIME field in Interrupt Remapping Table Address Register is Set), the Compatibility format interrupts are blocked.
- If Compatibility format interrupts are disabled (CFIS field in the Global Status Register is Clear), Compatibility format interrupts are blocked.
- Otherwise, Compatibility format interrupts are processed as described by Section 5.1.2.1 without undergoing interrupt remapping.

— Interrupt requests in the Remappable format (i.e., request with Interrupt Format field Set) are processed as follows:

- The reserved fields in the Remappable interrupt requests are checked to be zero. If the reserved field checking fails, the interrupt request is blocked. Else, the Source-id, Handle, SHV, and Subhandle fields are retrieved from the interrupt request.
- Hardware computes the interrupt_index per the algorithm described in Section 5.1.3. The computed interrupt_index is validated to be less than the interrupt-remapping table size configured in the Interrupt Remap Table Address Register. If the bounds check fails, the interrupt request is blocked.
- If the above bounds check succeeds, the IRTE corresponding to the interrupt_index value is either retrieved from the Interrupt Entry Cache, or fetched from the interrupt-remapping table. If the Coherent (C) field is reported as Clear in the Extended Capability Register, the IRTE fetch from memory will not snoop the processor caches. Hardware must read the entire IRTE as a single operation and not use multiple reads to get the contents of the IRTE as software may change the contents of the IRTE atomically. Hardware implementations reporting Memory Type Support (MTS=1 in ECAP_REG) must use write-back (WB) memory type for IRTE fetches. If the Present (P) field in the IRTE is Clear, the interrupt request is blocked and treated as a fault.
- If IRTE is present (P=1), hardware performs verification of the interrupt requester per the programming of the SVT, SID and SQ fields in the IRTE as described in Section 9.9. If the source-id checking fails, the interrupt request is blocked.

— If IRTE has Mode field clear (IM=0):[1]

- Hardware interprets the IRTE in remappable format (as described in Section 9.9). If invalid programming of remappable-format IRTE is detected, the interrupt request is blocked.
- If above checks succeed, a remapped interrupt request is generated per the programming of the IRTE fields[2].

- Any of the above checks that result in interrupt request to be blocked is treated as a interrupt-remapping fault condition. The interrupt-remapping fault conditions are enumerated in the following section.

5.1.4.1 Interrupt Remapping Fault Conditions

The following table enumerates the various conditions resulting in faults when processing interrupt requests. A fault conditions is treated as 'qualified' if the fault is reported to software only when the Fault Processing Disable (FPD) field is Clear in the IRTE used to process the faulting interrupt request. Interrupt translation faults are non-recoverable and the faulting interrupt request is treated as an Unsupported Request by the remapping hardware.

1. If the IM field is 1, hardware interprets the IRTE in posted format (as described in Section 9.10). Refer to Section 5.2.3 for interrupt-posting hardware operation.
2. When forwarding the remapped interrupt request to the system bus, the 'Trigger Mode Level' field in the interrupt request on the system bus is always set to "asserted" (1b).

Table 15. Interrupt Remapping Fault Conditions

Interrupt Remapping Fault Conditions	Fault Reason	Qualified	Behavior
Decoding of the interrupt request per the Remappable request format detected one or more reserved fields as Set.	20h	No	Unsupported Request
The interrupt_index value computed for the Remappable interrupt request is greater than the maximum allowed for the interrupt-remapping table size configured by software, or hardware attempt to access the IRTE corresponding to the interrupt_index value referenced an address above Host Address Width (HAW).	21h	No	
The Present (P) field in the IRTE entry corresponding to the interrupt_index of the interrupt request is Clear.	22h	Yes	
Hardware attempt to access the interrupt-remapping table through the Interrupt-Remapping Table Address (IRTA) field in the Interrupt Remap Table Address Register resulted in error.	23h	No	
Hardware detected one ore more reserved fields that are not initialized to zero in an IRTE with Present (P) field Set. This also includes cases where software programmed various conditional reserved fields wrongly.	24h	Yes	
On Intel® 64 platforms, hardware blocked an interrupt request in Compatibility format either due to Extended Interrupt Mode Enabled (EIME field Set in Interrupt Remapping Table Address Register) or Compatibility format interrupts disabled (CFIS field Clear in Global Status Register).	25h	No	
Hardware blocked a Remappable interrupt request due to verification failure of the interrupt requester's source-id per the programming of SID, SVT and SQ fields in the corresponding IRTE with Present (P) field Set.	26h	Yes	
Hardware attempt to access the Posted Interrupt Descriptor (PID) through the Posted Descriptor Address High/Low fields of an IRTE for posted interrupts resulted in error.[1]	27h	Yes	
Hardware detected one or more reserved fields that are not initialized to zero in a Posted Interrupt Descriptor (PID).[1]	28h	Yes	
Hardware detected an untranslated request-without-PASID to the interrupt address range (FEEx_xxxxh) which does not meet all the requirements to be considered a valid interrupt. Refer to the PCI Express specification regarding the format required of message signaled interrupts.	29h	No	
Extended Interrupt Mode Enable (EIME) field with a value of 0 in Interrupt Remapping Table Address Register (IRTA_REG) used to process an interrupt request for implementation reporting Extended Interrupt Mode Enable Required (EIMER) as Set in the Extended Capability Register	2Ah	No	
Remapping hardware reporting Interrupt Remapping Required (IRREQ) as Set in the Extended Capability Register, received a MSI request when Interrupt Remapping is disabled.	2Bh	No	

1. Fault Reasons 27h and 28h are applicable only for interrupt requests processed through IRTEs programmed for Interrupt Posting as described in Section 9.10. Refer to Section 5.2 for details on Interrupt Posting.

5.1.5 Programming Interrupt Sources To Generate Remappable Interrupts

Software performs the following general steps to configure an interrupt source to generate remappable interrupts:

- Allocate a free interrupt remap table entry (IRTE) and program the remapped interrupt attributes per the IRTE format described in Section 9.9.
- Program the interrupt source to generate interrupts in remappable format with appropriate handle, subhandle and SHV fields that effectively encodes the index of the allocated IRTE as the interrupt_index defined in Section 5.1.3. The interrupt_index may be encoded using the handle, subhandle and SHV fields in one of the following ways:
  - SHV = 0; handle = interrupt_index;
  - SHV = 1; handle = interrupt_index; subhandle = 0;

— SHV = 1; handle = 0; subhandle = interrupt_index;

— SHV = 1; handle = interrupt_index - subhandle;

The following sub-sections describes example programming for I/OxAPIC, MSI and MSI-X interrupt sources to generate interrupts per the Remappable interrupt request format.

5.1.5.1 I/OxAPIC Programming

Software programs the Redirection Table Entries (RTEs) in I/OxAPICs as illustrated in Figure 5-3.

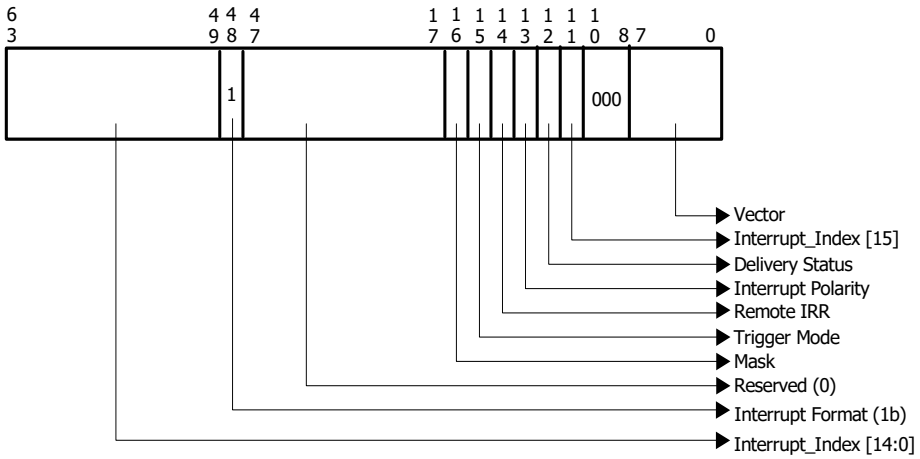


Figure 5-3. I/OxAPIC RTE Programming

- The Interrupt_Index[14:0] is programmed in bits 63:49 of the I/OxAPIC RTE. The most significant bit of the Interrupt_Index (Interrupt_Index[15]) is programmed in bit 11 of the I/OxAPIC RTE.
- Bit 48 in the I/OxAPIC RTE is Set to indicate the Interrupt is in Remappable format.
- RTE bits 10:8 is programmed to 000b (Fixed) to force the SHV (SubHandle Valid) field as Clear in the interrupt address generated.
- The Trigger Mode field (bit 15) in the I/OxAPIC RTE must match the Trigger Mode in the IRTE referenced by the I/OxAPIC RTE. This is required for proper functioning of level-triggered interrupts.
- For platforms using End-of-Interrupt (EOI) broadcasts, Vector field in the I/OxAPIC RTEs for level-triggered interrupts (i.e. Trigger Mode field in I/OxAPIC RTE is Set, and Trigger Mode field in the IRTE referenced by the I/OxAPIC RTE is Set), must match the Vector field programmed in the referenced IRTE. This is required for proper processing of End-Of-Interrupt (EOI) broadcast by the I/OxAPIC.
- Programing of all other fields in the I/OxAPIC RTE are not impacted by interrupt remapping.

5.1.5.2 MSI and MSI-X Register Programming

Figure 5-4 illustrates the programming of MSI/MSI-X address and data registers to support remapping of the message signaled interrupt.

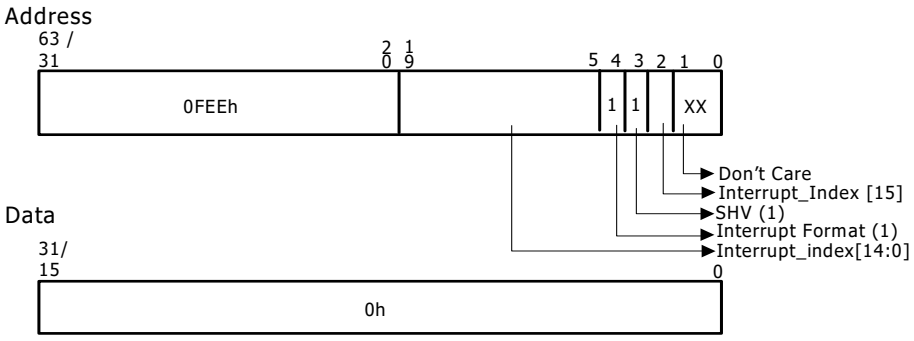


Figure 5-4. MSI-X Programming

Specifically, each address and data registers must be programmed as follows:

- Address register bits 63/31: 20 must be programmed with the interrupt address identifier value of 0FEEh.
- Address register bits 19:5 is programmed with Interrupt_Index[14:0] and address register bit 2 must be programmed with Interrupt_Index[15]. The Interrupt_Index is the index of the Interrupt Remapping Table Entry (IRTE) that remaps the corresponding interrupt requests.
  - Devices supporting MSI allows software to enable multiple vectors (up to 32) in powers of 2. For such multiple-vector MSI usages, software must allocate N contiguous IRTE entries (where N is the number of vectors enabled on the MSI device) and the interrupt_index value programmed to the Handle field must be the index of the first IRTE out of the N contiguous IRTEs allocated. The device owns the least significant log-N bits of the data register, and encodes the relative interrupt number (0 to N-1) in these bits of the interrupt request payload.
- Address register bit 4 must be Set to indicate the interrupt is in Remappable format.
- Address register bit 3 is Set so as to set the SubHandle Valid (SHV) field in the generated interrupt request.
- Data register is programmed to 0h.

5.1.6 Remapping Hardware Event Interrupt Programming

Interrupts generated by remapping hardware events are not subject to interrupt remapping. The following sections describe the programming of the Fault Event, Invalidation Completion Event, and Page Request Event data/address registers on Intel®64 platforms. The Trigger Mode and Level values are fixed at 0 for remapping hardware event interrupts. Implementations reporting Extended Interrupt Mode Enable Required (EIMER) as Set in the Extended Capability Register support programming in Intel® 64 x2APIC mode only.

5.1.6.1 Programming in Intel® 64 xAPIC Mode

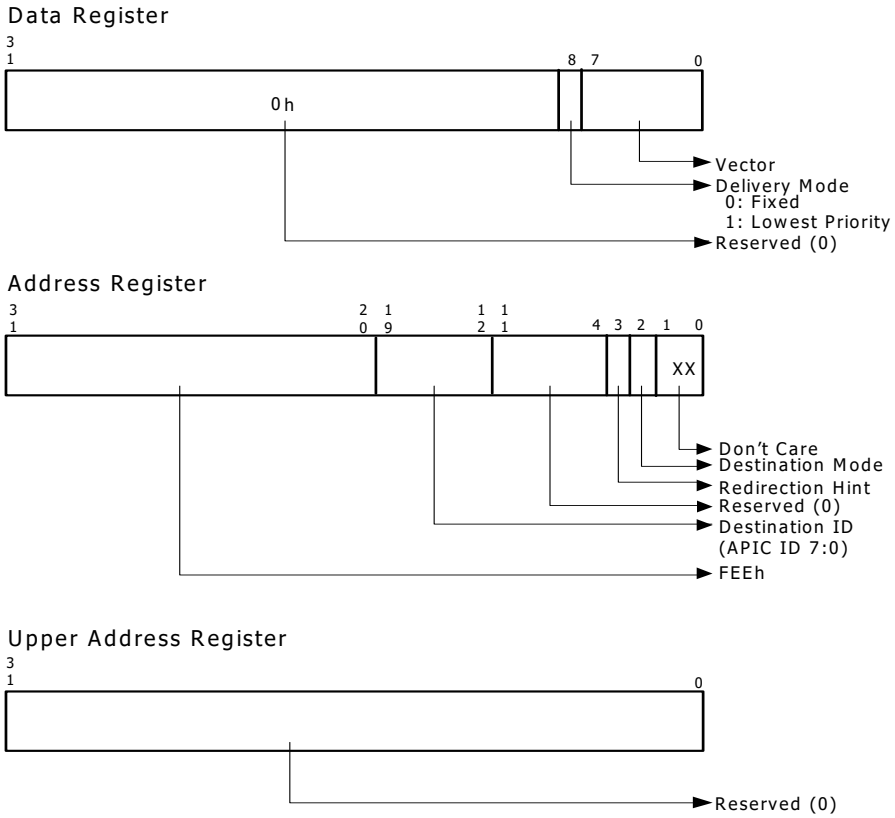


Figure 5-5. Remapping Hardware Interrupt Programming in Intel® 64 xAPIC Mode

5.1.6.2 Programming in Intel® 64 x2APIC Mode[1]

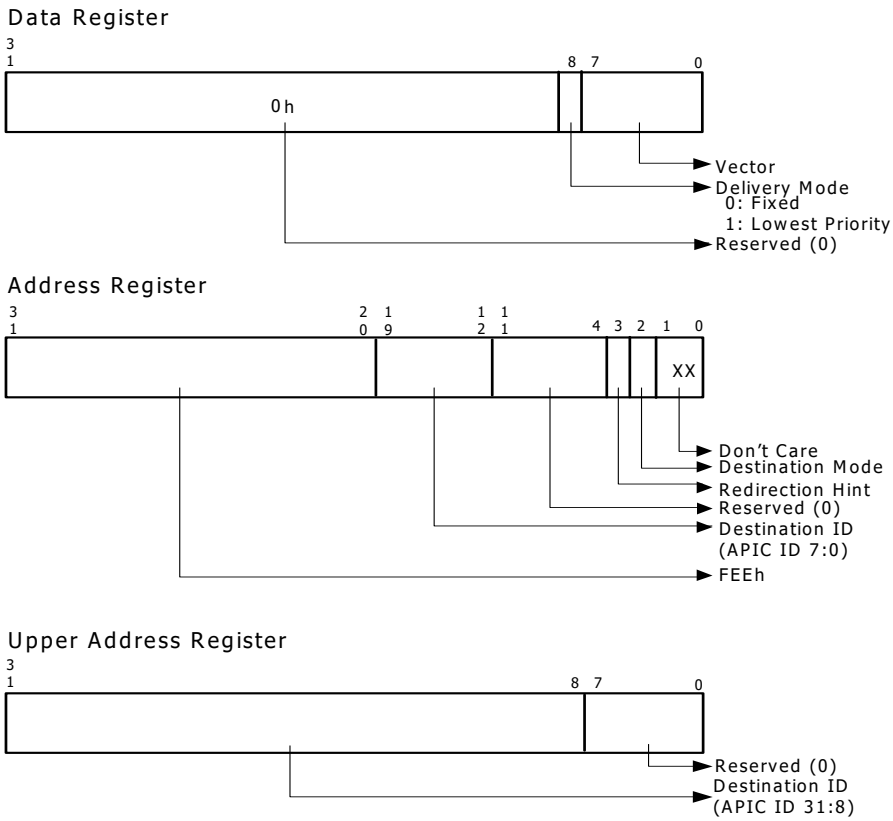


Figure 5-6. Remapping Hardware Interrupt Programming in Intel® 64 x2APIC Mode

5.1.7 Handling of Platform Events

Platforms supporting interrupt remapping are highly recommended to use side-band mechanisms (such as dedicated pins between chipset/board-logic and CPU), or in-band methods (such as platform/vendor defined messages) to deliver platforms events such as SMI/PMI/NMI/INIT/MCA. This is to avoid the dependence on system software to deliver these critical platform events.

Some existing platforms are known to use I/OxAPIC RTEs (Redirection Table Entries) to deliver SMI, PMI and NMI events. There are at least two existing initialization approaches for such platform events delivered through I/OxAPIC RTEs.

- Some existing platforms report to system software the I/OxAPIC RTEs connected to platform event sources through ACPI, enabling system software to explicitly program/enable these RTEs. Example for this include, the 'NMI Source Reporting' structure in ACPI MADT (for reporting NMI source).
- Alternatively, some existing platforms program the I/OxAPIC RTEs connected to specific platform event sources during BIOS initialization, and depend on system software to explicitly preserve

1. Hardware support for x2APIC mode is reported through the EIM field in the Extended Capability Register. x2APIC mode is enabled through the Interrupt Remapping Table Address Register.

these RTEs in the BIOS initialized state. (For example, some platforms are known to program specific I/OxAPIC RTE for SMI generation through BIOS before handing control to system software, and depend on system software preserving the RTEs pre-programmed with SMI delivery mode).

On platforms supporting interrupt-remapping, delivery of SMI, PMI and NMI events through I/OxAPIC RTEs require system software programming the respective RTEs to be properly remapped through the Interrupt Remapping Table. To avoid this management burden on system software, platforms supporting interrupt remapping are highly recommended to avoid delivering platform events through I/OxAPIC RTEs, and instead deliver them through dedicated pins (such as the processor's xAPIC LINTn input) or through alternative platform-specific messages.

5.2 Interrupt Posting

Interrupt-posting capability is an extension of interrupt-remapping hardware for extended processing of remappable format interrupt requests. Interrupt-posting enables a remappable format interrupt request to be posted (recorded) in a coherent main memory resident data-structure, with an optional notification event to the CPU complex to signal pending posted interrupt.

Interrupt-posting capability (along with the support in Intel® 64 processors for posted-interrupt processing and APIC Virtualization) enables a Virtual Machine Monitor (VMM) software to efficiently process interrupts from devices assigned to virtual machines. Section 2.5.3 describes high-level usages and benefits of interrupt-posting. Refer to *'Intel® 64 Architecture Software Developer's Manual, Volume 3B: System Programming'* for details on Intel® 64 processor support for APIC virtualization and posted-interrupt processing.

Remapping hardware support for interrupt-posting capability is reported through the Posted Interrupt Support (PI) field in the Capability register (CAP_REG). Section 11.4.2 describes interrupt-posting capability reporting.

5.2.1 Interrupt Remapping Table Support for Interrupt Posting

All remappable interrupt requests are processed through the Interrupt Remapping Table as described in Section 5.1.3. The IRTE Mode (IM) field in an Interrupt Remapping Table Entry (IRTE) specifies if remappable interrupt requests processed through that IRTE is subject to interrupt-remapping or interrupt-posting.

- If the IM field is 0 in an IRTE, the IRTE is interpreted in remappable format (described in Section 9.9) to remap interrupt requests processed through it. The interrupt-remapping hardware operation is described in Section 5.1.4.
- If the IM field is 1 in an IRTE, the IRTE is interpreted in posted format (described in Section 9.10) to post interrupt requests processed through it. The interrupt-posting hardware operation is described in Section 5.2.3.

IRTE entries in posted format support following new fields:

- Address of the Posted Interrupt Descriptor data structure to post (record) the interrupt to. Section 5.2.2 describes the Posted Interrupt Descriptor.
- Urgent (URG) qualification to indicate if interrupt requests processed through this IRTE require real-time processing or not. Section 5.2.3 describes the hardware operation with this field.
- Vector field specifies the vector to use when posting interrupts processed through an IRTE. Unlike remappable-format (where the Vector field is used when generating the remapped interrupt request), the Vector field for posted-format IRTEs is used to determine which bit to Set when posting the interrupt to the Posted Interrupt Descriptor referenced by the IRTE.

As with interrupt remapping, interrupts generated by the remapping hardware itself are not subject to interrupt posting.

5.2.2 Posted Interrupt Descriptor

Posted Interrupt Descriptor is a 64-byte aligned and sized structure in memory used by interrupt-posting hardware to post (record) interrupt requests subject to posting. Section 9.11 describes the Posted Interrupt Descriptor Format. System software must allocate the Posted Interrupt Descriptors in coherent (write-back) main memory.

The Posted Interrupt Descriptor hosts the following fields:

- Posted Interrupt Request (PIR) field provides storage for posting (recording) interrupts (one bit per vector, for up to 256 vectors).
- Outstanding Notification (ON) field indicates if there is a notification event outstanding (not processed by processor or software) for this Posted Interrupt Descriptor. When this field is 0, hardware modifies it from 0 to 1 when generating a notification event, and the entity receiving the notification event (processor or software) resets it as part of posted interrupt processing.
- Suppress Notification (SN) field indicates if a notification event is to be suppressed (not generated) for non-urgent interrupt requests (interrupts processed through an IRTE with URG=0).
- Notification Vector (NV) field specifies the vector for notification event (interrupt).
- Notification Destination (NDST) field specifies the physical APIC-ID of the destination logical processor for the notification event.

5.2.3 Interrupt-Posting Hardware Operation

Interrupt requests in remappable format are processed by hardware as described in Section 5.1.4. When such processing encounters a IRTE entry in posted format (IM=1), the interrupt request is processed through posting (instead of remapping). The following provides a functional overview of the interrupt-posting hardware operation:

- If IRTE retrieved has Mode field as set (IM=1)[1]
  - Hardware interprets the IRTE in posted format (as described in Section 9.10). If invalid programming of posted-format IRTE is detected, the interrupt request is blocked.
  - If above checks succeed, the IRTE provides the pointer to the Posted Interrupt Descriptor (PDA-L/PDA-H), the vector value (Vector) to be posted, and if the interrupt request is qualified as urgent (URG) or not.
- Hardware performs a coherent atomic read-modify-write operation of the posted-interrupt descriptor as follows:
  - Hardware implementations reporting Memory Type Support (MTS=1 in ECAP_REG) must use write-back (WB) memory type with the atomic operation that updates posted-interrupt descriptor.
  - This atomic read-modify-write operation will always snoop processor caches irrespective of the value of Pagewalk Coherency (C) field in Extended Capability Register (ECAP_REG).
  - Read contents of the Posted Interrupt Descriptor, claiming exclusive ownership of its hosting cache-line. If invalid programming (e.g., non-zero reserved fields) of Posted Interrupt Descriptor is detected, release ownership of the cache-line, and block the interrupt request.
  - If above checks succeed, retrieve current values of Posted Interrupt Requests (PIR bits 255:0), Outstanding Notification (ON), Suppress Notification (SN), Notification Vector (NV), and Notification Destination (NDST) fields in the Posted Interrupt Descriptor.
  - Modify the following descriptor field values atomically:
    - Set bit in PIR corresponding to the Vector field value from the IRTE
    - Compute X = ((ON == 0) & (URG | (SN == 0)))

1. If the IM field is 0, hardware interprets the IRTE in remapped format (described in Section 9.9). Refer to Section 5.1.4 for interrupt-remapping hardware operation.

- If (X == 1), Set ON field.

— Promote the cache-line to be globally observable, so that the modifications are visible to other caching agents. Hardware may write-back the cache-line anytime after this step.

— If (X == 1) in previous step, generate a notification event (interrupt) with attributes as follows:

- NSDT field specifies the physical APIC-ID of destination logical CPU. Refer to Section 9.11 on how this field is interpreted for xAPIC and x2APIC modes.
- NV field specifies the vector to be used for the notification interrupt to signal the destination CPU about pending posted interrupt.
- Delivery mode field for notification interrupt is forced to Fixed (000b)
- Re-direction Hint field for notification interrupt is forced to Clear (0b)
- Trigger Mode field for notification interrupt is forced to Edge (0b)
- Trigger Mode Level field for notification interrupt is forced to Asserted (1b).

- Any of the above checks that result in interrupt request to be blocked is treated as a interrupt-remapping fault condition as enumerated in Section 5.1.4.1.

5.2.4 Ordering Requirements for Interrupt Posting

This section summarizes the ordering requirements to be met by interrupt-posting hardware when posting interrupts.

- Interrupt requests are posted transactions and follow Interrupt ordering rules. This ensures that an interrupt request will not be observed by software until prior inbound DMA writes are committed to their destinations.
  — This requirement needs to be maintained even if the interrupt requests are posted. i.e., before an interrupt is posted (recorded) in the posted-interrupt descriptor and made visible to software, all preceding DMA writes must be completed.
- Since interrupt requests are posted transactions, upstream read completions must push preceding interrupt requests.
  — This requirement needs to be maintained even if one or more of the preceding interrupt requests are posted. i.e., An upstream read completion must wait until all preceding interrupts (irrespective of if they are remapped or posted) are completed. In case of an interrupt that is posted, 'completion' of the interrupt means, both the atomic update of the posted interrupt descriptor and the associated notification event are completed.
- In the interrupt-posting operation, hardware must make sure that modifications to a posted-interrupt descriptor is observable to software before issuing the notification event for that descriptor.
- Ordering between distinct interrupt requests does not need to be maintained within the interrupt-posting hardware.

5.2.5 Using Interrupt Posting for Virtual Interrupt Delivery

This section is informative and intended to illustrate a simplified example[1] usage of how a Virtual Machine Monitor (VMM) software may use interrupt-posting hardware to support efficient delivery of virtual interrupts from assigned devices to virtual machines.

VMM software may enable interrupt-posting for a virtual machine as follows:

1. This simplified usage example assumes the VMM software typically runs with interrupts masked, except perhaps when placing the logical CPUs in low power states. The example illustrated here may be extended to cover other usage scenarios.

- For each virtual processor in the virtual machine, the VMM software may allocate a Posted Interrupt Descriptor. Each such descriptor is used for posting all interrupts that are to be delivered to the respective virtual processor.
  - Software must block devices from accessing the memory where Posted Interrupt Descriptors are located. One way to achieve this is by setting up remapping tables so that accesses from devices to Posted Interrupt Descriptors are blocked by the remapping hardware.
  - If a device is able to write to the Posted Interrupt Descriptor, atomicity of posted interrupt operation is not guaranteed.
- The VMM software allocates two physical interrupt vectors (across all logical CPUs in the platform) for notification events.
  - One of this physical vectors may be used as the 'Active Notification Vector' (ANV) for posted interrupt notifications to any virtual processor that is active (executing) at the time of posting an interrupt to it.
  - The other physical vector allocated may be used as the 'Wake-up Notification Vector' (WNV) for posted interrupt notifications to any virtual processor that is blocked (halted) at the time of posting an interrupt to it.
- For each interrupt source from any assigned device(s) to this virtual machine, the VMM software may intercept and virtualize the guest software programming of respective interrupt resources (IOxAPIC entries and/or MSI/MSI-X registers). Through this virtualization, the VMM software detects the target virtual processor and virtual vector assigned by guest software.
- For each such interrupt source, the VMM software allocates a posted-format IRTE.
  - The vector field in each such IRTE is programmed by the VMM software with the respective virtual vector value assigned for the interrupt source by guest software.
  - The posted descriptor address field in each such IRTE is programmed by the VMM software to reference the posted descriptor allocated for the virtual processor assigned by guest software for the interrupt source.
  - The urgent (URG) field in an IRTE is Set by the VMM software if the respective interrupt source is designated as requiring immediate (non-deferred) processing.
- The VMM software configures the processor hardware to enable APIC virtualization (including 'virtual-interrupt delivery' and 'process posted interrupts' capabilities) for the virtual processors.
  - The 'posted-interrupt notification vector' for the virtual processors are configured with the 'Active Notification Vector' (ANV) value described earlier in this section.
  - The 'posted-interrupt descriptor' for the virtual processors are configured with the address of the Posted Interrupt Descriptor allocated for respective virtual processors.
- The VMM software scheduler may manage a virtual processor's scheduling state as follows:
  - When a virtual processor is selected for execution, the virtual processor state is designated as 'active' before entering/resuming it. This state is specified in its Posted Interrupt Descriptor by programming its Notification Vector (NV) field with the ANV vector value[1]. This allows all interrupts for this virtual processor that are received while it is active (running) are processed by the processor hardware without transferring control to the VMM software. The processor hardware processes these notification events (with ANV vector value) by transferring any posted interrupts in the Posted Interrupt Descriptor to the Virtual-APIC page of the virtual processor and directly delivering it (without VMM software intervention) to the virtual processor. Refer to '*Intel® 64 Architecture Software Developer's Manual, Volume 3: System*

1. There may be varying approaches for VMM software to manage notification vectors. For example, an alternate approach may be for VMM software to allocate unique Activation Notification Vectors (ANV) for each virtual processor (as opposed to sharing the same ANV for all virtual processors). This approach may enable such VMM software to avoid switching between active and wake-up vector values in the Posted Interrupt Descriptor on virtual processor scheduling state changes, and instead update them only on virtual processor migrations across logical processors.

Programming Guide' for details on Intel® 64 processor support for APIC Virtualization and Posted-Interrupt Processing.

  - When a virtual processor is preempted (e.g., on quantum expiry), the virtual processor state is designated as 'ready-to-run'. This state is specified in its Posted Interrupt Descriptor by programming the Suppress Notification (SN) field to 1. This allows all non-urgent interrupts for this virtual processor received while it is in preempted state to be posted to its Posted Interrupts Descriptor without generating a notification interrupt (thereby avoiding disruption of currently running virtual processors). If there are interrupt sources qualified as urgent for targeting this virtual processor, the VMM software may also modify the NV field in the Posted Interrupt Descriptor to WNV vector value. This enables the VMM software to receive notifications (with WNV vector value) when urgent interrupts are posted when virtual processor is not running, allowing appropriate software actions (such as preempting the current running virtual processor and immediately scheduling this virtual processor).
  - When a virtual processor halts (e.g., on execution of HLT instruction), the VMM software may get control, blocks further execution of the virtual processor, and designate the virtual processor state as 'halted'. This state is specified in its Posted Interrupt Descriptor by programming its Notification Vector (NV) field with the WNV vector value. This enables the VMM software to receive notifications (with WNV vector value) when any interrupt (urgent or non-urgent) is posted for this virtual processor, allowing appropriate software action (such as to schedule the virtual processor for future or immediate activation).

- When entering/resuming a virtual processor, the VMM software may process any pending posted interrupts in its posted descriptor as follows:
  - VMM first transitions the virtual CPU to 'active' state by programming the notification vector in the Posted Interrupt Descriptor to ANV vector value.
  - VMM may check if there are pending interrupts in the posted descriptor (e.g. by scanning PIR field for non-zero value).
  - If there are pending posted interrupts, VMM may generate a self-IPI[1] (Inter Processor Interrupt to the same logical CPU) with vector value of ANV, through the Local xAPIC. This interrupt is recognized by the processor as soon as interrupts are enabled in the virtual processor enter/resume path. Since the virtual processor is configured with ANV vector value as the 'posted-interrupt notification vector', this results in processor hardware processing it same as any notification event it may receive while the virtual processor is active. This approach enables the VMM software to 'off-load' the posted interrupt processing (such as delivering the interrupt to the virtual processor through the Virtual-APIC) to the processor hardware, irrespective of the scheduling state of the virtual processor when the interrupt was posted by remapping hardware to the Posted Interrupt Descriptor.
- The VMM software may also apply the 'posted-interrupt processing' capability of the processor to inject virtual interrupts generated by VMM software to a virtual machine (in addition to interrupts from direct assigned devices to the virtual machine). This may be done by the VMM software atomically 'posting' a virtual interrupt to the Posted Interrupt Descriptor (using atomic/LOCK instructions that enforces cache-line update atomicity) and generating a notification event (as IPI) to the logical processor identified as notify destination in the Posted interrupt Descriptor.
- The VMM software may handle virtual processor migrations across logical processors by atomically updating the Notification Destination (NDST) field in the respective Posted Interrupt Descriptor to the physical APIC-ID of the logical processor to which the virtual processor is migrated to. This enables all new notification events from the posted descriptor of this virtual processor to be routed to the new logical processor.

1. The usage illustrated in this section assumes the VMM software is executing with interrupts disables, and interrupts are enabled by the processor hardware as part of entering/resuming the virtual processor. For VMM software implementations that have interrupt enabled in the VMM, precaution must be taken by the VMM software to disable interrupt on the logical processor before generating the self-IPI and resuming the virtual processor.

5.2.6 Interrupt Posting for Level Triggered Interrupts

Level-triggered interrupts generated through IOxAPICs Redirection Table Entries (illustrated in Figure 5-3) can be processed through Interrupt Remap Table Entries (IRTE) for Posted Interrupts (illustrated in Section 9.10). However, unlike with interrupt-remapping, all interrupts (including level interrupts) processed by the posted interrupt processing hardware are treated as edge-triggered interrupts. Thus VMM software enabling posting of Level-triggered interrupts must take special care to properly virtualize the End of Interrupt (EOI) processing by the virtual processor. For example, the VMM software may set up the virtual processor execution controls to gain control on EOI operation to the Virtual APIC controller by guest software, and virtualize the operation by performing a Directed-EOI to the IOxAPIC that generated the level-triggered interrupt. A Directed-EOI is performed by software writing directly to the IOxAPIC EOI register. Refer to the IOxAPIC specification for details on IOxAPIC EOI register.

5.3 Memory Type and Snoop Behavior Summary

The table below summarizes cache snooping behavior for memory accesses during the interrupt translation process. The table also summarizes the memory type used when accesses are made on a coherent link. The memory type value provided by hardware is not used by a non-coherent link.

- A value of 1 implies memory access snoops processor caches. A value of 0 implies that the memory access does not snoop processor caches.
- ECAP.C is the Page-walk Coherency field in Extended Capability Register (ECAP_REG).
- Hardware implementations reporting Memory Type Support (MTS=1 in ECAP_REG) must use write-back (WB) memory type for IRTE and PID access.

Table 16. Memory Type and Snoop Behavior for Interrupt Remap Structures

Interrupt Structure Access	Snoop	Memory Type
Read of Interrupt Remap Table Entry	ECAP.C	WB
Atomic Update of Posted Interrupt Descriptor	1	WB

6 Caching Translation Information

Remapping hardware may accelerate the address-translation process by caching data from the memory-resident paging structures. Because the hardware does not ensure that the data that it caches are always consistent with the structures in memory, it is important for software to comprehend how and when the hardware may cache such data, what actions can be taken to remove cached data that may be inconsistent, and when it should do so.

6.1 Caching Mode

The Caching Mode (CM) field in Capability Register indicates if the hardware implementation caches not-present or erroneous translation-structure entries. When the CM field is reported as Set, any software updates to remapping structures other than first-stage mapping (including updates to not-present entries or present entries whose programming resulted in translation faults) requires explicit invalidation of the caches.

Hardware implementations of this architecture must support operation corresponding to CM=0. Operation corresponding to CM=1 may be supported by software implementations (emulation) of this architecture for efficient virtualization of remapping hardware. Software managing remapping hardware should be written to handle both caching modes.

Software implementations virtualizing the remapping architecture (such as a VMM emulating remapping hardware to an operating system running within a guest partition) may report CM=1 to efficiently virtualize the hardware. Software virtualization typically requires the guest remapping structures to be shadowed in the host. Reporting the Caching Mode as Set for the virtual hardware requires the guest software to explicitly issue invalidation operations on the virtual hardware for any/all updates to the guest remapping structures. The virtualizing software may trap these guest invalidation operations to keep the shadow translation structures consistent to guest translation structure modifications, without resorting to other less efficient techniques (such as write-protecting the guest translation structures through the processor's paging facility).

6.2 Address Translation Caches

This section provides architectural behavior of following remapping hardware address translation caches:

- Context-cache
  - Caches context-entry, or scalable-mode context-entry encountered on a address translation of requests.
- PASID-cache
  - Caches scalable-mode PASID-table entries encountered on address translation of requests.
- I/O Translation Look-aside Buffer (IOTLB)
  - Caches the effective translation for a request. This can be the result of second-stage only page-walk, first-stage only page-walk, or nested page-walk - depending on the type of request (with or without PASID) that is address translated, and the programming of the DMA remapping hardware and various translation structures.

- Paging-structure Caches
  - — Caches the intermediate paging-structure entries (i.e., entries referencing a paging-structure entry) encountered on a first-stage page-walk or second-stage page-walk (including for nested translation).
- HPT Caches
  - — Caches storing leaf contents with effective permissions for translated-requests.
  - — Caches storing intermediate paging-structure entries encountered during an HPT walk.

6.2.1 Tagging of Cached Translations

Remapping architecture supports tagging of various translation caches as follows. Tags can be used by remapping hardware for cache lookup during address translation processing, or for selecting cache entries to be invalidated. For a lookup to be considered a match, all the associated tags must match. Depending on the type of invalidation, one or more tag matches are sufficient to consider the entry as a match.

Table 17. Cache Tagging

Caches	Tags for Lookup/ Invalidation	Legacy Mode	Scalable Mode
Context Cache	Lookup	• Source-ID	• Source-ID
	Invalidation	• Source-ID • Domain-ID	• Source-ID
PASID Cache	Lookup	NA	• PASID • Source-ID
	Invalidation	NA	• PASID • Domain-ID
First-stage paging structure cache	Lookup	NA	• PASID • Domain-ID • Address
	Invalidation	NA	• PASID • Domain-ID • Address

Table 17. Cache Tagging

Caches	Tags for Lookup/ Invalidation	Legacy Mode	Scalable Mode
Second-stage paging structure cache	Lookup	• Domain-ID • Address	• Domain-ID • Address
	Invalidation	• Domain-ID • Address	• Domain-ID • Address
IOTLB	Lookup	• Source-ID • Address	Req-without-PASID • Entry allocated by request-without-PASID • Source-ID • Address Req-with-PASID • Entry allocated by request-with-PASID • Source-ID • PASID • Address
	Invalidation	• Domain-ID • Address	• Domain-ID • PGTT • PASID • Address
HPT leaf cache	Lookup	NA	Req-without-Pasid • Entry allocated by request-without-PASID • Source-ID • Address Req-with-PASID • Entry allocated by request-with-PASID • Source-ID • Address • PASID
	Invalidation	NA	• HPT Domain-ID • Address
HPT paging-structure cache	Lookup	NA	• HPT Domain-ID • Address
	Invalidation	NA	• HPT Domain-ID • Address

- Address tagging:
  - IOTLB entries are tagged by the upper bits of the input-address (called the page number) in the request that resulted in allocation of the respective cache entry.

Table 18. Address Tags for IOTLB

Type of Mapping	4K pages	2M pages	1G pages
first-stage only or nested mapping	Addr[N:12]	Addr[N:21]	Addr[N:30]
second-stage only mapping	Addr[MGAW:12]	Addr[MGAW:21]	Addr[MGAW:30]
pass-through mapping	Addr[HAW:12]	Addr[HAW:21]	Addr[HAW:30]

N=56 for 5-level paging and N=47 for 4-level paging.

  - Paging-structure caches are tagged by the respective bits of the input-address.

Table 19. Address Tags for Paging-structure Caches

Type of Mapping	PML5E	PML4E	PDPE	PDE
FS page-structure cache	Addr[56:48]	Addr[56:39]	Addr[56:30]	Addr[56:21]
SS page-structure cache	Addr[MGAW:48]	Addr[MGAW:39]	Addr[MGAW:30]	Addr[MGAW:21]

  - HPT caches storing page permissions and/or paging-structures are tagged by the respective bits of the input-address.

Table 20. Address Tags for HPT caches

HPTL4E	HPTL3E	HPTL2E	HPTL1E
Addr[HAW-1:41]	Addr[HAW-1:33]	Addr[HAW-1:25]	Addr[HAW-1:17]

- PASID tagging:
  - In scalable mode, requests-without-PASID are treated as requests-with-PASID when looking up the paging-structure cache, and PASID-cache. Such lookups use the PASID value from the RID_PASID field in the scalable-mode context-entry used to process the request-without-PASID. Refer to Section 9.4 for more details on scalable-mode context-entry. Additionally, after translation process when such requests fill into IOTLB, the entries are tagged with PASID value obtained from RID_PASID field but are still marked as entries for requests-without-PASID. Tagging of such entries with PASID value is required so that PASID-selective P_IOTLB invalidation can correctly remove all stale mappings. Implementation may allow requests-with-PASID from a given Requester-ID to hit entries brought into IOTLB by requests-without-PASID from the same Requester-ID to improve performance.
- Interrupt-index tagging:
  - Interrupt-remapping cache is architecturally tagged by the interrupt-index of remappable-format interrupt requests that resulted in allocation of the interrupt-entry-cache entry.

Tagging of cached translations enable remapping hardware to cache information to process requests from multiple endpoint devices targeting multiple address-spaces. Tagging also enable software to efficiently invalidate groups of cached translations that are associated with the same tag value.

6.2.2 Context-Cache

Context-cache is used to cache context-entries or scalable-mode context-entries used to address translate requests. Each cached entry is referenced by the source-id in the request.

For implementations reporting Caching Mode (CM) as 0 in the Capability Register, if any fault conditions are encountered as part of accessing a context-entry, or scalable-mode context-entry, the resulting entry is not cached in the context-cache (and hence software is not required to invalidate the context-cache on modifications to such entries). See Section 7.1.3 for list of conditions that are treated as a fault.

For implementations reporting Caching Mode (CM) as 1 in the Capability Register, the implementation may cache entries that would result in a fault, such as entries with P=0. Such implementations require explicit invalidation by software to invalidate such cached entries. In legacy mode, the reserved domain-id value of 0 is used to tag the cached context entries that would result in a fault. In scalable mode, context cache entries are not tagged with domain-id, so such entries don't use the reserved domain-id of 0.

6.2.2.1 Context-Entry Programming Considerations

When modifying root-entries, scalable-mode root-entries, context-entries, or scalable-mode context entries software must serially invalidate the context-cache, PASID-cache (if applicable), and the IOTLB. The serialization is required since hardware may utilize information from the context-caches (e.g. Domain-ID) to tag new entries inserted to the PASID-cache and IOTLB for processing in-flight requests. Section 6.5 describes the invalidation operations.

Software must follow the additional guidelines below when using remapping hardware in Legacy Mode:

- Software must ensure that, if multiple context-entries are programmed with the same Domain-id (DID), such entries must be programmed with the same values for the Second-Stage Page Translation Pointer (SSPTPTR) and Address Width (AW) fields. This is required since hardware implementations tag the various translation caches with DID (see Section 6.2.1). Context-entries with the same value in the SSPTPTR field are recommended to use the same DID value for best hardware efficiency.
- When modifying fields in present (P=1) context entries, software must ensure that at any point of time during the modification (performed through single or multiple write operations), the before and after state of the entry being modified is individually self-consistent. For example, software performing an SSPTPTR or Translation Type (TT) field update must use a 16-Byte aligned atomic write of Intel® 64 processors to simultaneously update the DID field. This is required as remapping hardware tags some of the translation caches with DID and may be fetching these entries at any point of time while they are being modified by software. Software modifying these present (P=1) entries are also responsible to ensure these does not impact in-flight transactions from the affected endpoint devices[1].
- Software must not use domain-id value of 0 on when programming context-entries on implementations reporting CM=1 in the Capability register. (See Section 6.2.2 for details.)

6.2.3 PASID-Cache

PASID-cache is used to cache scalable-mode PASID-table entries that are used for address translations. This cache is only used when hardware is operating in scalable mode (RTADDR_REG.TTM=01b). Request-without-PASID uses the PASID value from the RID_PASID field in the scalable-mode context entry to reference the PASID-cache. Hardware implementations should take care that the same PASID coming from multiple source-ids does not thrash itself in PASID-cache.

For implementations reporting Caching Mode (CM) as 0 in the Capability Register, if any fault conditions are encountered leading up to or as part of accessing a scalable-mode PASID-table entry, the entry is not cached in the PASID-cache (and hence software is not required to invalidate the PASID Cache on modifications to such entries). (see Section 7.1.3 for list of conditions that are treated as a fault).

1. Example usages for modifying present context entry may include modifying the translation-type (TT) field to transition between pass-through and non-pass-through modes. Section 9.3 for details on these fields.

For implementations reporting Caching Mode (CM) as 1 in the Capability Register, the implementation may cache entries that would result in a fault, such as entries with P=0. Such implementations require explicit invalidation by software to invalidate such cached entries. The reserved domain-id value of 0 is used to tag the cached entries that would result in a fault.

6.2.3.1 Scalable-Mode PASID-Table Entry Programming Considerations

When modifying fields in present (P=1) entries, software must ensure that at any point of time during the modification (performed through single or multiple write operations), the before and after state of the entry being modified is individually self-consistent. For example, software performing an SSPTPTR or PGTT field update must use a 16-Byte aligned atomic write of Intel® 64 processors to simultaneously update the DID field. This is required as remapping hardware tags some of the translation caches with DID and may be fetching these entries at any point of time while they are being modified by software. Software modifying these present (P=1) entries are also responsible to ensure these do not impact in-flight transactions from the affected endpoint devices[1].

When modifying scalable-mode PASID-table entries software must serially invalidate the PASID-cache and the IOTLB. The serialization is required since hardware may utilize information from the PASID-cache (e.g., Domain-ID) to tag new entries inserted to the IOTLB for processing in-flight requests. Section 6.5 describes the invalidation operations.

Software must ensure that, if multiple scalable-mode PASID-table entries are programmed with the same Domain-id (DID), such entries must be programmed with the same value for the Second-Stage Page Translation Pointer (SSPTPTR) field, and the same value for the Address Width (AW) field. This is required since hardware implementations tag the various translation caches with domain-id (see Section 6.2.1). Scalable-mode PASID-table entries with the same value in the SSPTPTR field are recommended to use the same domain-id value for best hardware efficiency.

Software must program a valid value in the DID field of all scalable-mode PASID-table entries, including entries where the PASID Granular translation type (PGTT) field is set to first-stage-only or pass-through (PGTT equal to 001b or 100b). Scalable-mode PASID table entries programmed for first-stage translation or pass-through (PGTT equal to 001b or 100b) must be programmed with a DID value that is different from those used in any PASID table entries that are programmed for second-stage or nested translation (PGTT equal to 010b or 011b). This is required since hardware implementations tag various caches with domain-id as described in Section 6.2.1. Scalable-mode PASID-table entries with PGTT value of 001b or 100b are recommended to use the same domain-id value for best hardware efficiency.

Software must ensure that, if multiple scalable-mode PASID table entries for a given PASID (across different Source IDs) are programmed with the same Domain-ID (DID), such entries must be programmed with the same value for the Second-Stage Page Translation Pointer (SSPTPTR) field, the Address Width (AW) field, the First-Stage Page Translation Pointer (FSPTPTR) field, the First-Stage Paging Mode (FSPM) field, and the PASID Granular Translation Type (PGTT) field.

Software must ensure that, if multiple scalable-mode PASID-table entries are programmed with the same value for tuple {FSPTPTR, SSPTPTR}, such entries must be programmed with the same values for tuple {FSPM, EAFE}. This is required because FSPM and EAFE are considered properties of the first-stage page-table structure and are independent of the PASID(s) used to access the page-table structure. For PASID-table entries with PGTT value of First-stage only (PGTT=001) the SSPTPTR value in the tuple can be ignored. This requirement does not apply to PASID-table entries with Second-stage-only and Pass-through translations as they don’t have first-stage page-tables.

Software must ensure that, if multiple scalable-mode PASID-table entries are programmed with the same value for SSPTPTR, such entries must be programmed with the same value for SSADE. This is required because SSADE is considered a property of the second-stage page-table structure and is

1. Example usages for modifying present scalable-mode PASID-table entries may include modifying the PASID Granular Translation-Type (PGTT) field to transition between pass-through and non-pass-through modes. Refer to Section 9.6 for details on these fields.

independent of the Domain-ID(s) attached to the second-stage table. This requirement does not apply to PASID-table entries with First-stage-only and Pass-through translations as they don't have second-stage page-tables.

Software must not use a domain-id value of 0 when programming scalable-mode PASID-table entries on implementations reporting CM=1 in the capability register. (See Section 6.2.3 for details.)

When programming a PASID-table entry that contains a PGTT field with a value of 011b (nested), host software must check all first-stage table related fields (whose values are provided by guest software) to not trigger any architecture faults. Host software must reject guest software's request to program PASID-table entry fields such as FSPTPTR, EAFE, WPE, and FSPM with bad values.

6.2.4 IOTLB

Remapping hardware caches information about the translation of input-addresses in the IOTLB. IOTLB may cache information with different functionality as below:

- First-stage mappings:
  - Each of these is a mapping from a input page number in a request to the physical page frame to which it translates (derived from first-stage translation), along with information about access privileges and memory typing (if applicable).
- Second-stage mappings:
  - Each of these is a mapping from a input page number in a request to the physical page frame to which it translates (derived from second-stage translation), along with information about access privileges and memory typing (if applicable).
- Nested mappings:
  - Each of these is a mapping from a input page number in a request to the physical page frame to which it translates (derived from both first-stage and second-stage translation), along with information about access privileges and memory typing (if applicable).
- Pass-through mappings:
  - Each of these is a mapping from an input page number in a request to the physical page frame to which it translates (derived as pass-through mapping).

Each entry in an IOTLB is an individual translation. Each translation is referenced by a page number. Each entry architecturally contains the following information:

- IOTLB entries hosting first-stage mappings:
  - The physical address corresponding to the page number (the page frame).
  - The access rights from the first-stage paging-structure entries used to translate input-addresses with the page number (see Section 3.6.1)
    - The logical-AND of the R/W flags.
    - The logical-AND of the U/S flags.
  - Attributes from a first-stage paging-structure entry that identifies the final page frame for the page number (either a PTE or a first-stage paging-structure entry with PS=1):
    - The dirty flag (see Section 3.6.2).
    - The memory type (see Section 3.11).
- IOTLB entries hosting second-stage mappings:
  - The physical address corresponding to the page number (the page frame).
  - The access rights from the second-stage paging-structure entries used to translate input-addresses with the page number (see Section 3.7.1)

    - The logical-AND of the R flags.
    - The logical-AND of the W flags.
  - Attributes from a second-stage paging-structure entry that identifies the final page frame for the page number (either an SS-PTE or a second-stage paging-structure entry with PS=1):
    - The memory type (see Section 3.11).
    - The snoop (SNP) bit (see Section 3.10 and Section 4.2.3).
    - The dirty flag (necessary only if SSADE=1)
- IOTLB entries hosting nested mappings:
  - The physical address corresponding to the page number (the page frame).
  - The combined access rights from the first-stage paging-structure and second-stage paging-structure entries used to translate input-addresses with the page number (see Section 3.8.1)
    - The logical-AND of the R/W flags (from first-stage translation) and W flags (from second-stage translation of the result of first-stage translation).
    - The logical-AND of the U/S flags (from first-stage translation).
  - Attributes from a first-stage paging-structure entry that identifies the final page frame for the page number (either a PTE or a paging-structure entry with PS=1):
    - The dirty flag (see Section 3.6.2).
  - Combined attributes from first-stage and second-stage paging-structure entries that identifies the final page frame for the page number (either a page-table-entry or a paging-structure entry with PS=1):
    - The memory type (see Section 3.11).

IOTLB entries may contain other information as well. A remapping hardware may implement multiple IOTLBs, and some of these may be for special purposes, e.g., only for instruction fetches. Such special-purpose IOTLBs may not contain some of this information if it is not necessary. For example, a IOTLB used only for instruction fetches need not contain information about the R/W and dirty flags.)

As noted in Section 6.2.1, any IOTLB entries created by hardware are associated with appropriate tags (e.g., source-id of request that allocated the entry, PASID value if request is associated to a PASID, domain-id from the context entry or scalable-mode PASID-table entry that led to the translation, etc.).

Remapping hardware need not implement any IOTLBs. Remapping hardware that do implement IOTLBs may evict or invalidate any IOTLB entry at any time. Software should not rely on the existence of IOTLBs or on the retention of IOTLB entries.

6.2.4.1 Details of IOTLB Use

For implementations reporting Caching Mode (CM) as 0 in the Capability Register, IOTLB caches only valid mappings (i.e. results of successful page-walks that did not result in a translation fault). Specifically, if any of the translation fault conditions described in Section 7.1.3 are encountered, the results are not cached in the IOTLB.

For implementations reporting Caching Mode (CM) as Set in the Capability Register, these translation fault conditions may cause caching of the faulted translation in the IOTLB. The caching of such faulted translations in IOTLB follows same tagging as if there was no faults (i.e., source-id of request that allocated the entry, PASID value if request is associated with a PASID, domain-id from the context entry or scalable-mode PASID-table entry that led to the translation, etc.). The CM capability does not apply to first-stage mappings. Even when CM is 1, hardware does not cache translations in the IOTLB that encountered a fault in first-stage mapping.

With first-stage translation, before caching a translation, hardware sets the accessed (A) flag to 1 in each of the first-stage paging-structure entries used for the translation, if not already set. If EAFE = 1 in the scalable-mode PASID-table entry, hardware also sets the extended-accessed (EA) flag to 1 in each of the paging-structure entries, if not already set.

With second-stage translation, when CM is 0 and SSADE in the PASID-table entry used to process the request is 1, before caching a translation, hardware sets the accessed (A) flag to 1 in each of the second-stage paging-structure entries used for the translation, if not already set.

With nested translation, hardware applies these rules for both first- and second-stage translation. Setting of accessed and extended-accessed flags in the first-stage paging structure entries is subject to write permission checks at second-stage translation.

When CM is 1, hardware applies the above rules to translations where all paging structures leading to the translation are present. However the A and EA flags are not set when caching not present entries.

If the page number of a input-address corresponds to a IOTLB entry tagged with the right source-id (and PASID, if applicable), the hardware may use that IOTLB entry to determine the page frame, access rights, and other attributes for accesses to that input-address. In this case, the hardware may not actually consult the paging structures in memory. The hardware may retain a IOTLB entry unmodified even if software subsequently modifies the relevant paging-structure entries in memory. See Section 6.5 for how software can ensure that the hardware uses the modified paging-structure entries.

If the paging structures specify a translation using a page larger than 4-KBytes, some hardware implementations may choose to cache multiple smaller-page IOTLB entries for that translation. Each such IOTLB entry would be associated with a page number corresponding to the smaller page size (e.g., bits N:12 of a input-address with first-stage translation, where N is 56 bits with 5-level paging or 47 bits with 4-level paging), even though part of that page number (e.g., bits 20:12) is part of the offset with respect to the page specified by the paging structures. The upper bits of the physical address in such a IOTLB entry are derived from the physical address in the PDE used to create the translation, while the lower bits come from the input-address of the access for which the translation is created.

There is no way for software to be aware that multiple translations for smaller pages have been used for a large page. If software modifies the paging structures so that the page size used for a 4-KByte range of input-addresses changes, the IOTLBs may subsequently contain multiple translations for the address range (one for each page size). A reference to a input- address in the address range may use any of these translations. Which translation is used may vary from one execution to another, and the choice may be implementation-specific.

6.2.5 Caches for Paging Structures

Remapping hardware may cache frequently used paging-structure entries that reference other paging-structure entries (as opposed to page frames). Depending on the type of the paging-structure entry cached, the paging-structure caches may be classified as PML5-cache, PML4-cache, PDPE-cache, and PDE-cache. These may cache information with different functionality as below:

- First-stage-paging-structure entries:
  - Each of these is a mapping from the upper portion of a input-address in a request to the physical address of the first-stage paging structure used to translate the corresponding region of the input-address space, along with information about access privileges. For example, with 5-level paging, bits 56:48 of the input-address would map to the address of the relevant first-stage PML4 table; with 4-level paging, bits 47:39 of the input-address would map to the address of the relevant first-stage page-directory-pointer table.
- Second-stage-paging-structure entries:
  - Each of these is a mapping from the upper portion of a input-address to the physical address of the second-stage paging structure used to translate the corresponding region of the input-address space, along with information about access privileges. For example, bits MGAW:39 of

the input-address would map to the address of the relevant second-stage page-directory-pointer table. When hardware is operating in scalable mode (RTADDR_REG.TTM=01b) and PASID Granular Translation Type (PGTT) field in scalable-mode PASID-table entry is programmed as nested, the input-address can be the input-address in a request (with or without PASID), or can be the second-stage address of a first-stage paging-structure entry (accessed as part of a nested translation).

- Combined-paging-structure entries:
  — Each of these is a mapping from the upper portion of a input-address in a request to the physical address of the first-stage paging structure (after nesting through second-stage translation) used to translate the corresponding region of the input-address space, along with information about access privileges.

Hardware implementations may implement none or any of these paging-structure-caches, and may use separate caches in implementation specific ways to manage different types of cached mappings (e.g., first-stage and nested mappings may be held in one cache and second-stage in a different cache, or any other formations).

6.2.5.1 PML5-cache

When 5-level paging is effective, each entry in a PML5-cache holds the following information:

- PML5-cache entries hosting first-stage PML5Es:
  — Each PML5-cache entry caching a first-stage PML5E is referenced by a 9-bit value and is used for input-addresses for which bits 56:48 have that value.
  — The entry contains information from the PML5E used to translated such input-addresses:
    - The physical address from the PML5E (address of first-stage PML4 table).
    - The value of R/W flag of the PML5E.
    - The value of U/S flag of the PML5E.
- PML5-cache entries hosting second-stage PML5Es:
  — Each PML5-cache entry caching a second-stage mapping is referenced by a N-bit value and is used for input-addresses for which bits MGAW:48 have that value.
  — The entry contains information from the SS-PML5E used to translate such input-addresses:
    - The physical address from the SS-PML5E (address of second-stage PML4 table).
    - The value of R flag of the SS-PML5E.
    - The value of W flag of the SS-PML5E.
- PML5-cache entries hosting nested PML5Es:
  — Each PML5-cache entry caching a nested mapping is referenced by a 9-bit value and is used for input-addresses for which bits 56:48 have that value.
  — The entry contains information from the first-stage PML5E used to translate such input-addresses, combined with information from the nested second-stage translation of the physical address from that PML5E:
    - The physical address from the second-stage translation of the address in the PML5E (physical-address of first-stage PML4 table).
    - The R/W flag from the PML5E.
    - The value of U/S flag of the PML5E.

The following describes how a hardware implementation may use the PML5-cache:

- If the hardware has a PML5-cache entry for a input-address, it may use that entry when translating the input-address (instead of the PML5E in memory).

- For first-stage mappings, hardware does not create a PML5-cache entry unless the P flag is 1 and all reserved bits are 0 in the PML5E in memory. For nested mappings, hardware also does not create a PML5-cache entry unless there is a second-stage translation with read permission for the address in PML5E. For second-stage mappings, hardware does not create a PML5E-cache entry unless at least one of R and W flags is 1 and all reserved bits are 0 in the SS-PML5E in memory[1].
- For first-stage mappings, before creating a PML5-cache entry, hardware sets the accessed (A) flag to 1 in the PML5E in memory, if it is not already 1. Hardware also sets the extended-accessed (EA) flag to 1, if EAFE=1. With nested translation, setting of accessed and extended-accessed flags are subject to write permission checks at second-stage translation.
- The hardware may create a PML5-cache entry even if there are no translations for any input-address that might use that entry (e.g., because the P flags are 0 in all entries in the referenced PML4 table).

If the hardware creates a PML5-cache entry, the hardware may retain it unmodified even if software subsequently modifies the corresponding PML5E (SS-PML5E) in memory.

6.2.5.2 PML4-cache

Each entry in a PML4-cache holds the following information:

- PML4-cache entries hosting first-stage PML4Es:
  - Each PML4-cache entry caching a first-stage PML4E is referenced by a 9-bit (or 18-bit with 5-level paging) value and is used for input-addresses for which bits N:39 (N=56 with 5-level paging and N=47 with 4-level paging) have that value.
  - The entry contains information from the PML5E and PML4E used to translated such input-addresses:
    - The physical address from the PML4E (address of first-stage page-directory-pointer table).
    - The logical-AND of the R/W flags in the PML5E and PML4E (with 5-level paging), or the value of R/W flag of the PML4E (with 4-level paging).
    - The logical-AND of the U/S flags in the PML5E and PML4E (with 5-level paging), or the value of U/S flag of the PML4E (with 4-level paging).
- PML4-cache entries hosting second-stage PML4Es:
  - Each PML4-cache entry caching a second-stage mapping is referenced by a N-bit value and is used for input-addresses for which bits MGAW:39 have that value.
  - The entry contains information from the SS-PML5E and SS-PML4E used to translate such input-addresses:
    - The physical address from the SS-PML4E (address of second-stage page-directory-pointer table).
    - The logical-AND of the R flags in the SS-PML5E and SS-PML4E (with 5-level translation), or the value of R flag of the SS-PML4E (with 4-level translation).
    - The logical-AND of the W flags in the SS-PML5E and SS-PML4E with 5-level translation), or the value of W flag of the SS-PML4E (with 4-level translation).
- PML4-cache entries hosting nested PML4Es:
  - Each PML4-cache entry caching a nested mapping is referenced by a 9-bit (or 18-bit with 5-level paging) value and is used for input-addresses for which bits N:39 (where N=56 with 5-level paging and N=47 with 4-level paging) have that value.
  - The entry contains information from the first-stage PML5E and PML4E used to translate such input-addresses, combined with information from the nested second-stage translation of the physical address from that PML4E:

1. This behavior applies for implementations reporting Caching Mode (CM) as 0 in the Capability register. See Section 6.1 for caching behavior on implementations reporting CM=1.

- The physical address from the second-stage translation of the address in the PML4E (physical-address of the first-stage page-directory-pointer table).
- With 5-level paging: The logical-AND of the R/W flags in the PML5E and PML4E. With 4-level paging: The R/W flag from the PML4E
- With 5-level paging: The logical-AND of the U/S flags in the PML5E and PML4E. With 4-level paging: The value of U/S flag of the PML4E.

The following items detail how a hardware implementation may use the PML4-cache:

- If the hardware has a PML4-cache entry for a input-address, it may use that entry when translating the input-address (instead of the PML5E and PML4E in memory).
- For first-stage mappings, hardware does not create a PML4-cache entry unless the P flag is 1 and all reserved bits are 0 in the PML5E and PML4E in memory. For nested mappings, hardware also does not create a PML4-cache entry unless there is a second-stage translation with read permission for the address in PML5E and PML4E. For second-stage mappings, hardware does not create a PML4E-cache entry unless at least one of R and W flags is 1 and all reserved bits are 0 in the SS-PML5E and SS-PML4E in memory[1].
- For first-stage mappings, before creating a PML4-cache entry, hardware sets the accessed (A) flag to 1 in the relevant PML5E and PML4E in memory, if it is not already 1. Hardware also sets the extended-accessed (EA) flag to 1 in these entries, if EAFE=1. With nested translation, setting of accessed and extended-accessed flags are subject to write permission checks at second-stage translation[1].
- The hardware may create a PML4-cache entry even if there are no translations for any input-address that might use that entry (e.g., because the P flags are 0 in all entries in the referenced page-directory-pointer table).
- If the hardware creates a PML4-cache entry, the hardware may retain it unmodified even if software subsequently modifies the corresponding PML5E (SS-PML5E) or PML4E (SS-PML4E) in memory.

6.2.5.3 PDPE-cache

Each entry in a PDPE-cache holds the following information:

- PDPE-cache entries hosting first-stage PDPEs:
  - — Each PDPE-cache entry caching a first-stage PDPE is referenced by an 18-bit (27-bit with 5-level paging) value and is used for input-addresses for which bits N:30 (where N=56 with 5-level paging and N=47 with 4-level paging) have that value.
  - — The entry contains information from the PML5E (with 5-level paging), PML4E and PDPE used to translate such input-addresses:
    - The physical address from the PDPE (address of first-stage page-directory). (No PDPE-cache entry is created for a PDPE that maps a page.)
    - The logical-AND of the R/W flags in the PML5E (with 5-level paging), PML4E and PDPE.
    - The logical-AND of the U/S flags in the PML5E (with 5-level paging), PML4E and PDPE.
- PDPE-cache entries hosting second-stage PDPEs:
  - — Each PDPE-cache entry caching an SS-PDPE is referenced by a N-bit value and is used for input-addresses for which bits MGAW:30 have that value.
  - — The entry contains information from the SS-PML4E and SS-PDPE used to translated such input-addresses:
    - The physical address from the SS-PDPE (address of second-stage page-directory). (No PDPE-cache entry is created for an SS-PDPE that maps a page.)

1. This behavior applies for implementations reporting Caching Mode (CM) as 0 in the Capability register. See Section 6.1 for caching behavior on implementations reporting CM=1.

- The logical-AND of the R flags in the SS-PML5E (with 5-level translation), SS-PML4E and SS-PDPE.
- The logical-AND of the W flags in the SS-PML5E (with 5-level translation), SS-PML4E and SS-PDPE.

- PDPE-cache entries hosting nested PDPEs:
  - Each PDPE-cache entry caching a nested mapping is referenced by a 18-bit (27-bit with 5-level paging) value and is used for input-addresses for which bits N:30 (where N=56 with 5-level paging and N=47 with 4-level paging) have that value.
  - The entry contains information from the PML5E (with 5-level paging), PML4E and PDPE used to translated such input-addresses, combined with information from the nested second-stage translation of the physical address from that PDPE:
    - The physical address from the second-stage translation of the address in the PDPE (physical-address of first-stage page-directory). (No PDPE-cache entry is created for a PDPE that maps a page.)
    - The logical-AND of the R/W flags in the PML5E (with 5-level paging), PML4E and PDPE.
    - The logical-AND of the U/S flags in the PML5E (with 5-level paging), PML4E and PDPE.

The following items detail how a hardware implementation may use the PDPE-cache:

- If the hardware has a PDPE-cache entry for a input-address, it may use that entry when translating the input-address (instead of the PML5E, PML4E and PDPE in memory).
- For first-stage mappings, hardware does not create a PDPE-cache entry unless the P flag is 1 and all reserved bits are 0 in the PML5E, PML4E and the PDPE in memory. For nested mappings, hardware also does not create a PDPE-cache entry unless there is a second-stage translation with read permission for the address in the PML5E, PML4E and the PDPE. For second-stage mappings, hardware does not create a PDPE-cache entry unless at least one of R and W flags is 1 and all reserved bits are 0 in the SS-PML5E, SS-PML4E and the SS-PDPE in memory[1].
- For first-stage mappings, before creating a PDPE-cache entry, hardware sets the accessed (A) flag to 1 in the relevant PML5E, PML4E and PDPE in memory, if it is not already 1. Hardware also sets the extended-accessed (EA) flag to 1 in these entries, if EAFE=1. With nested translation, setting of accessed and extended-accessed flags are subject to write permission checks at second-stage translation.
- The hardware may create a PDPE-cache entry even if there are no translations for any input-address that might use that entry (e.g., because the P flags are 0 in all entries in the referenced page-directory)
- If the hardware creates a PDPE-cache entry, the hardware may retain it unmodified even if software subsequently modifies the corresponding PML5E (SS-PML5E), PML4E (SS-PML4E) or PDPE (SS-PDPE) in memory.

6.2.5.4 PDE-cache

Each entry in a PDE-cache holds the following information:

- PDE-cache entries hosting first-stage PDEs:
  - Each PDE-cache entry caching a first-stage PDE is referenced by an 27-bit (36-bit with 5-level paging) value and is used for input-addresses for which bits N:21 (where N=56 with 5-level paging and N=47 with 4-level paging) have that value.
  - The entry contains information from the PML5E (with 5-level paging), PML4E, PDPE, and PDE used to translate such input-addresses:
    - The physical address from the PDE (address of first-stage page-table). (No PDE-cache entry is created for a PDE that maps a page.)

1. This behavior applies for implementations reporting Caching Mode (CM) as 0 in the Capability register. See Section 6.1 for caching behavior on implementations reporting CM=1.

- The logical-AND of the R/W flags in the PML5E (with 5-level paging), PML4E, PDPE, and PDE.
- The logical-AND of the U/S flags in the PML5E (with 5-level paging), PML4E, PDPE, and PDE.

- PDE-cache entries hosting second-stage PDEs:
  - Each PDE-cache entry caching an SS-PDE is referenced by a N-bit value and is used for input-addresses for which bits MGAW:21 have that value.
  - The entry contains information from the SS-PML5E (with 5-level translation), SS-PML4E, SS-PDPE, and SS-PDE used to translated such input-addresses:
    - The physical address from the SS-PDE (address of second-stage page-table). (No PDE-cache entry is created for an SS-PDE that maps a page.)
    - The logical-AND of the R flags in the SS-PML5E (with 5-level translation), SS-PML4E, SS-PDPE, and SS-PDE.
    - The logical-AND of the W flags in the SS-PML5E (with 5-level translation), SS-PML4E, SS-PDPE, and SS-PDE.
- PDE-cache entries hosting nested PDEs:
  - Each PDE-cache entry caching a nested mapping is referenced by a 27-bit (36-bit with 5-level paging) value and is used for input-addresses for which bits N:21 (where N=56 with 5-level paging and N=47 with 4-level paging) have that value.
  - The entry contains information from the PML5E (with 5-level paging), PML4E, PDPE and PDE used to translated such input-addresses, combined with information from the nested second-stage translation of the physical address from that PDE:
    - The physical address from the second-stage translation of the address in the PDE (physical-address of first-stage page-table). (No PDE-cache entry is created for a PDE that maps a page.)
    - The logical-AND of the R/W flags in the PML5E (with 5-level paging), PML4E, PDPE and PDE.
    - The logical-AND of the U/S flags in the PML5E (with 5-level paging), PML4E, PDPE and PDE.

The following items detail how a hardware implementation may use the PDE-cache:

- If the hardware has a PDE-cache entry for a input-address, it may use that entry when translating the input-address (instead of the PML5E, PML4E, the PDPE, and the PDE in memory).
- For first-stage mappings, hardware does not create a PDE-cache entry unless the P flag is 1 and all reserved bits are 0 in the PML5E, PML4E, the PDPTE, and the PDE in memory. For nested mappings, hardware also does not create a PDE-cache entry unless there is a second-stage translation with read permission for the address in the PML5E, PML4E, the PDPE, and the PDE. For second-stage mappings, hardware does not create a PDE-cache entry unless at least one of R and W flags is 1 and all reserved bits are 0 in the SS-PML5E, SS-PML4E, the SS-PDPE, and the SS-PDE in memory[1].
- For first-stage mappings, before creating a PDE-cache entry, hardware sets the accessed (A) flag to 1 in the relevant PML5E, PML4E, PDPE, and PDE in memory, if it is not already 1. Hardware also sets the extended-accessed (EA) flag to 1 in these entries, if EAFE=1. With nested translation, setting of accessed and extended-accessed flags are subject to write permission checks at second-stage translation.
- The hardware may create a PDE-cache entry even if there are no translations for any input-address that might use that entry (e.g., because the P flags are 0 in all entries in the referenced page-table).

1. This behavior applies for implementations reporting Caching Mode (CM) as 0 in the Capability register. See Section 6.1 for caching behavior on implementations reporting CM=1.

- If the hardware creates a PDE-cache entry, the hardware may retain it unmodified even if software subsequently modifies the corresponding PML5E (SS-PML5E), PML4E (SS-PML4E), PDPE (SS-PDPE), or PDE (SS-PDE) in memory.

6.2.5.5 Details of Paging-Structure Cache Use

For implementations reporting Caching Mode (CM) as Clear in the Capability Register, paging-structure-caches host only valid mappings (i.e. results of successful page-walks up to the cached paging-structure entry that did not result in a translation fault).

For implementations reporting Caching Mode (CM) as Set in the Capability Register, these translation fault conditions may cause caching of the faulted translation in the paging-structure caches. The caching of such faulted translations in paging-structure caches follows same tagging as if there was no faults (i.e. domain-id from the context entry or scalable-mode PASID-table entry that led to the translation, PASID value attached to the request, etc.). The CM capability does not apply to first-stage mappings. Even when CM is 1, hardware does not cache translations in first-stage paging-structure-caches that encountered a fault in first-stage mapping.

Information from a paging-structure entry can be included in entries in the paging-structure-caches for other paging-structure entries referenced by the original entry. For example, with 4-level paging, if the R/W flag is 0 in a PML4E, then the R/W flag will be 0 in any PDPTE-cache entry for a PDPTE from the page-directory-pointer table referenced by that PML4E. This is because the R/W flag of each such PDPTE-cache entry is the logical-AND of the R/W flags in the appropriate PML4E and PDPTE.

The paging-structure caches contain information only from paging-structure entries that reference other paging structures (and not those that map pages). For first-stage-paging-structure cache entries, because the G flag is not used in such paging-structure entries, the global-page feature does not affect the behavior of the paging-structure caches.

As noted in Section 6.2.1, any entries created in paging-structure caches are associated with the target domain-ID (and PASID when applicable).

A remapping hardware implementation may or may not implement any of the paging-structure caches. Software should rely on neither their presence nor their absence. The hardware may invalidate entries in these caches at any time. Because the hardware may create the cache entries at the time of translation and not update them following subsequent modifications to the paging structures in memory, software should take care to invalidate the cache entries appropriately when causing such modifications. The invalidation of IOTLBs and the paging-structure caches is described in Section 6.5.

6.2.6 HPT Cache

The HPT cache contains permissions for host physical addresses accessible by devices using translated requests. It also may contain intermediate HPT entries that reference other HPT tables. The HPT cache may contain entries for pages with no permission. Software is required to perform HPT cache invalidations when it modifies any part of an HPT entry. (i.e., adding/removing page permissions or changes to the Address Valid or Address fields.)

An HPT cache may be implemented by hardware storing only leaf contents, only paging-structure contents, or both. Only an HPT cache storing leaf contents can service a translated-requests without performing an HPT walk.

The HPT leaf cache must be tagged with Source-ID and Address. Request-without-PASID always use the Source-ID and Address for HPT lookups. Requests-with-PASID also use Source-ID and Address, and additionally use PASID. The HPT paging-structure cache uses the HPT Domain-ID from the scalable-mode PASID-table entry and the Address for HPT lookups.

Hardware implementations must not cache information, in HPT cache, from an HPT entry unless all conditions below are met:

- All reserved bits are 0 in the HPT entry in memory.

- At least one of the PP*i* fields are non-zero or the Address Valid field is Set.

An HPT cache may contain entries for multiple levels associated with the address from the request. In such a scenario the lowest level entry should be used to satisfy the request and the contents of any higher level entires should be ignored.

6.2.6.1 Prefetching of HPT Cache

The remapping hardware may prefetch HPT entries after it returns read and/or write permission in a Translation Completion. The translated address is used to prefetch the corresponding HPT entry for the translated address into one or more HPT caches. This action may be prevented by setting the HPT Prefetch Disable field in the in the scalable-mode PASID-table entry (see Section 9.6).

If Enable PASID in Translated Requests is Set in the scalable-mode context entry used to locate the HPT entry, the resulting HPT cache entry is allocated matching the type of Translation Request, e.g., if the Translation Request was without PASID the HPT entry would be allocated without PASID and if the Translation Request was with PASID the HPT entry would be allocated with PASID.

If Enable PASID in Translated Requests is Clear in the scalable-mode context entry used to locate the HPT entry, the PASID in the original Translation Request is ignored when performing the HPT prefetch. Instead, the value of RID_PASID in the scalable-mode context entry is used, to match the HPT lookup that will be performed for a subsequent Translated Request, and the HPT cache entry is allocated without PASID.

6.2.6.2 Scalable-Mode HPT Entry Programming Considerations

Software should use the following recommendations when enabling HPT:

- When a device supports translated request with PASID and the same HPT is used for all PASIDs of a Source-ID, software should not enable device support for Translated Request with PASID for that Source-ID. This would be the case, for example, of an SR-IOV device when a VF is assigned to a tenant and all PASIDs within the tenant are using the same HPT (e.g., a per-VF HPT).
- For a device that does not support translated requests with PASID or if it is not enabled in the device, software should clear the Enable PASID in Translated Request field (EPTR) in the scalable-mode context entry for that device. In this case, the HPT tables pointed to by the HPTPTR field in the PASID table entry for the RID_PASID should be programmed to include all HPAs accessible by any PASID used by the device.
- When the HPT Enable field and the Enable PASID in Translated Requests field are both Set in a scalable-mode context entry, each scalable-mode PASID-table entry must have the HPTPTR field programmed to point to an initialized HPTL4 table in memory.
- When the HPT Enable field is Set and the Enable PASID in Translated Requests field is Clear, only the scalable-mode PASID-table entry corresponding to RID_PASID needs to have the HPTPTR field programmed. The HPTPTR fields of other PASID-table entries are not used.
- Software must ensure that, if multiple scalable-mode PASID-table entries are programmed with the same HPT Domain-id (HPTDID), such entries must be programmed with the same value for the HPT root pointer (HPTPTR) field, and the same value for the HPT size (HPTSZ) field. This is required since hardware implementations tag the various translation caches with HPT domain-id (see Section 6.2.1). Scalable-mode PASID-table entries with the same value in the HPTPTR field are recommended to use the same HPT domain-id value for best hardware efficiency.
- When modifying fields in present scalable-mode PASID-table entries (P=1), software must ensure that at any point of time during the modification (performed through single or multiple write operations), the before and after state of the entry being modified is individually self-consistent. For example, software performing a HPTPTR or HPTSZ field update must use a 16-Byte aligned atomic write of Intel® 64 processors to simultaneously update the HPTDID field. This is required as remapping hardware tags some of the HPT caches with HPTDID and may be fetching these entries at any point of time while they are being modified by software.

- All HPT structures referenced by either an HPTPTR in a scalable-mode PASID-table entry, or an address field with an accompanied Address Valid as Set (AV=1 in HPTL4E, HPTL3E, or HPTL2E) must be initialized to zero prior to use by remapping hardware.

6.2.7 Translating Address Using Caches in Legacy Mode

If the hardware finds an IOTLB entry that is for the page number of the input-address and that is associated with the Source-ID in the request, it may use the physical address, access rights, and other attributes from that entry.

Hardware may use the source-ID of the request to select a context-cache entry. It can use that entry to qualify the request based on the attributes in the entry. If the hardware does not find a matching context-cache entry, it can traverse the root-table and context-table to obtain and cache the context-entry. Context-cache entry provides hardware with the Domain-ID attached to the Source-ID. If the context-cache entry indicates pass-through access, the request is processed as if it found a IOTLB entry with a matching unity translation. Else, it continues the translation process as follows

If the hardware does not find a relevant IOTLB entry, it may use the bits MGAW:21 of the input-address to select an entry from the PDE-cache that is associated with the Domain-ID. It can then use that entry to complete the translation process (locating a PTE, etc.) as if it had traversed the PML5E, PML4E, PDPE and PDE corresponding to the PDE-cache entry.

If the hardware does not find a relevant IOTLB entry and a relevant PDE-cache entry, it may use bits MGAW:30 of the input-address to select an entry from the PDPE cache that is associated with the Domain-ID. It can then use that entry to complete the translation process (locating an SS-PDE, etc.) as if it had traversed the PML5E, PML4E and the PDPE corresponding to the PDPE-cache entry.

If the hardware does not find a relevant IOTLB entry, a relevant PDE-cache entry, or a relevant PDPE-cache entry, it may use bits MGAW:39 of the input-address to select an entry from the PML4E-cache that is associated with the Domain-ID. It can then use that entry to complete the translation process (locating a PDPE, etc.) as if it had traversed the corresponding PML5E and SS-PML4E.

With 5-level translation, if the hardware does not find a relevant IOTLB entry, a relevant PDE-cache entry, a relevant PDPE-cache entry, or a relevant PML4E-cache entry, it may use bits MGAW:48 of the input-address to select an entry from the PML5E-cache that is associated with the Domain-ID. It can then use that entry to complete the translation process (locating a PML4E, PDPE, etc.) as if it had traversed the corresponding PML5E.

6.2.8 Multiple Cached Entries for a Single Paging-Structure Entry

The IOTLBs and paging-structure caches may contain multiple entries associated with a PASID and/or domain-ID and with information derived from a single paging-structure entry. For illustration, following are some examples for first-stage translation with 4-level paging (similar scenarios are possible with 5-level paging):

- Suppose that two PML4Es contain the same physical address and thus reference the same page-directory-pointer table. Any PDPTE in that table may result in two PDPTE-cache entries, each associated with a different set of input-addresses. Specifically, suppose that the $n1^{th}$ and $n2^{th}$ entries in the PML4 table contain the same physical address. This implies that the physical address in the m^{th} PDPTE in the page-directory-pointer table would appear in the PDPTE-cache entries associated with both p1 and p2, where (p1 » 9) = n1, (p2 » 9) = n2, and (p1 & 1FFH) = (p2 & 1FFH) = m. This is because both PDPTE-cache entries use the same PDPTE, one resulting from a reference from the $n1^{th}$ PML4E and one from the $n2^{th}$ PML4E.
- Suppose that the first PML4E (i.e., the one in position 0) contains the physical address X in PASID-table entry (the physical address of the PML4 table). This implies the following:
  - Any PML4-cache entry associated with input-address with 0 in bits 47:39 contains address X.
  - Any PDPTE-cache entry associated with input-addresses with 0 in bits 47:30 contains address X. This is because the translation for a input-address for which the value of bits 47:30 is 0 uses the value of bits 47:39 (0) to locate a page-directory-pointer table at address X (the

address of the PML4 table). It then uses the value of bits 38:30 (also 0) to find address X again and to store that address in the PDPTE-cache entry.

— Any PDE-cache entry associated with input-addresses with 0 in bits 47:21 contains address X for similar reasons.

— Any IOTLB entry for page number 0 (associated with input-addresses with 0 in bits 47:12) translates to page frame X » 12 for similar reasons.

The same PML4E contributes its address X to all these cache entries because the self-referencing nature of the entry causes it to be used as a PML4E, a PDPTE, a PDE, and a PTE.

Similar examples can be constructed with other paging structures (e.g., PDPE, PDE) and with PML5E (with 5-level paging). Multiple cached entries for a single paging-structure entry are also possible with second-stage translation (involving SS-PML5Es (with 5-level translation), SS-PML4Es, SS-PDPEs, SS-PDEs, and SS-PTEs).

6.3 Translation Caching at Endpoint Device

Chapter 4 described support for endpoint devices to request translations from remapping hardware and cache on Device-TLBs that are local to the endpoint device. Device-TLBs may be utilized to improve address-translation performance and/or to support recoverable translation faults (see Chapter 7). Translation requests from endpoint devices are address translated by the remapping hardware using its translation caches as described in previous sections, and the resulting translation is returned to the endpoint device in a Translation Completion. Refer to Section 4.1.2 for attributes returned in the Translation-Completion. The endpoint device may cache the information returned in the Translation-Completion locally in its Device-TLBs.

6.4 Interrupt Entry Cache

Remapping hardware supporting interrupt remapping may cache frequently used interrupt remapping table entries in the interrupt-entry-cache (IEC). Each entry in a interrupt-entry-cache is an individual interrupt-remap-table-entry. Each cached entry is referenced by the interrupt_index number computed from attributes in the interrupt request (see Section 5.1.3). Each interrupt-entry-cache entry architecturally contains the following information (see Section 9.9):

- Attributes of the remapped interrupt from the IRTE:
  - — Interrupt Vector
  - — Destination ID
  - — Delivery Mode
  - — Trigger Mode
  - — Redirection Hint
  - — Interrupt Mode (to determine interrupt is remapped or posted)
  - — Urgent Flag
  - — Posted Descriptor Address
- The Fault Processing Disable (FPD) flag from the IRTE.
- The interrupt source validation attributes (SID, SQ, SVT fields) from the IRTE.

For implementations reporting Caching Mode (CM) as Clear in the Capability Register, if any of the interrupt-remapping fault conditions described in Section 5.1.4.1 is encountered, the resulting entry is not cached in the IEC. For implementations reporting Caching Mode (CM) as Set in the Capability Register, interrupt-remapping fault conditions may cause caching of the corresponding interrupt remapping entries.

Remapping hardware utilize the interrupt-entry cache as follows:

- If the hardware finds an IEC entry that is for the interrupt_index number of the request, it may use the interrupt attributes from the IEC entry (subject to the interrupt-source validation checks as described in Section 9.9).
- If the hardware does not find a matching IEC entry, it uses the interrupt_index computed for the request to fetch the interrupt-remap-table-entry from the interrupt-remap-table.

6.5 Invalidation of Translation Caches

As noted in Section 6.2, the remapping hardware may create entries in the various translation caches when requests are translated, and it may retain these entries even after the translation structures used to create them have been modified by software. To ensure that address translation uses the modified translation structures, software should take action to invalidate any cached entries that may contain information that has since been modified.

For software to invalidate the various caching structures, the architecture supports the following two types of invalidation interfaces:

- **Register-based invalidation interface**: A legacy invalidation interface with limited capabilities. This interface is supported by hardware implementations of this architecture with Major Version 5 or lower (VER_REG). In all other hardware implementations, all requests are treated as invalid requests and will be ignored (for details see the CAIG field in the Context Command Register and the IAIG field in the IOTLB Invalidate Register).
- **Queued invalidation interface**: An expanded invalidation interface with extended capabilities, supported by later implementations of this architecture. Hardware implementations report support for queued invalidation interface through the Extended Capability Register (see Section 11.4.3).

The following sections provides more details on these hardware interfaces.

6.5.1 Register-based Invalidation Interface

The register-based invalidations provides a synchronous hardware interface for invalidations. Software writes to the invalidation command registers to submit invalidation command and may poll on these registers to check for invalidation completion.

Hardware implementations must process commands submitted through the invalidation registers irrespective of the remapping hardware enable status (i.e irrespective of TES and IES status in the Global Status Register. See Section 11.4.4.2).

Register-based invalidation has the following limitations:

- Register-based invalidation can be used only when queued-invalidations are not enabled.
- Register-based invalidations are not supported with scalable mode translation. This mode requires queued-invalidations to be enabled by software for proper operation.
- Register-based invalidation can target only invalidation of second-stage translations. Invalidation of first-stage and nested translations are not supported (which are supported only through queued-invalidations).
- Register-based invalidation cannot invalidate Device-TLBs on endpoint devices.
- Register-based invalidation cannot invalidate the interrupt entry cache.
- Register-based invalidation cannot invalidate the HPT caches.

The following sub-sections describe the register-based invalidation command registers.

6.5.1.1 Context Command Register

Context Command Register (see Section 11.4.6.1) supports invalidating the context-cache. The architecture defines the following types of context-cache invalidation requests. Hardware implementations may perform the actual invalidation at a coarser granularity if the requested invalidation granularity is not supported.

- *Global Invalidation*: All context-cache entries cached at the remapping hardware are invalidated.
- *Domain-Selective Invalidation*: Context-cache entries associated with the specified domain-id are invalidated.
- *Device-Selective Invalidation*: Context-cache entries associated with the specified device source-id and domain-id are invalidated.

When modifying root-entries or context-entries referenced by more than one remapping hardware units in a platform, software is responsible to explicitly invalidate the context-cache at each of these hardware units.

6.5.1.2 IOTLB Registers

IOTLB invalidation is supported through two 64-bit registers; (a) IOTLB Invalidate Register (see Section 11.4.6.3) and (b) Invalidation Address Register (see Section 11.4.6.4).

The architecture defines the following types of IOTLB invalidation requests. Hardware implementations may perform the actual invalidation at a coarser granularity if the requested invalidation granularity is not supported.

- *Global Invalidation*:
  - — All IOTLB entries are invalidated.
  - — All paging-structure-cache entries are invalidated.
- *Domain-Selective Invalidation*:
  - — IOTLB entries caching mappings (first-stage, second-stage, and nested) associated with the specified domain-id are invalidated.
  - — Paging-structure-cache entries caching mappings (first-stage, second-stage, and nested) associated with the specified domain-id are invalidated.
- *Page-Selective-within-Domain Invalidation*:
  - — IOTLB entries caching second-stage mappings associated with the specified domain-id and the second-stage-input-address range are invalidated.
  - — IOTLB entries caching first-stage and nested mappings associated with the specified domain-id are invalidated.
  - — Paging-structure-cache entries caching first-stage and nested mappings associated with the specified domain-id are invalidated.
  - — Paging-structure-cache entries caching second-stage mappings associated with the specified domain-id and the second-stage-input-address range are invalidated, if the Invalidation Hint (IH) field is Clear. Else, the paging-structure-cache entries caching second-stage mappings are preserved.

For any of the above operations, hardware may perform coarser invalidation. The actual invalidation granularity reported by hardware in the IOTLB Invalidate Register is always the granularity at which the invalidation was performed on the IOTLB.

When modifying page-table entries referenced by more than one remapping hardware units in a platform, software is responsible to explicitly invalidate the IOTLB at each of these hardware units.

6.5.2 Queued Invalidation Interface

The queued invalidation provides an advanced interface for software to submit invalidation requests to hardware and to synchronize invalidation completions with hardware. The Invalidation Queue (IQ) is also used by software to submit Page Group Response descriptors, which are described in Section 7.6.1. Hardware implementations report queued invalidation support through the Extended Capability Register.

The queued invalidation interface uses IQ, which is a circular buffer in system memory. Software submits commands by writing Invalidation Descriptors to the IQ. The following registers are defined to configure and manage the IQ:

- *Invalidation Queue Address Register*: Software programs this register to configure the base physical address and size of the contiguous memory region in the system memory hosting the Invalidation Queue. Remapping hardware reporting Scalable Mode Translation Support as Set (ECAP_REG.SMTS=1) or Abort DMA Support as Set (ECAP_REG.ADMS=1), allow software to additionally program the width of the descriptors (128-bits or 256-bits) that will be written into the Queue. Software should set up the Invalidation Queue for 256-bit descriptors before programming remapping hardware for scalable mode translation (RTADDR_REG.TTM=01b) or abort-dma mode (RTADDR_REG.TTM=11b) as 128-bit descriptors are treated as invalid descriptors (see Table 26 in Section 6.5.2.11) in scalable mode and abort-dma mode.
- *Invalidation Queue Head Register*: This register points to the invalidation descriptor in the IQ that hardware will process next. The Invalidation Queue Head register is incremented by hardware after fetching a valid descriptor from the IQ.
- *Invalidation Queue Tail Register*: This register points to the invalidation descriptor in the IQ to be written next by software. Software increments this register after writing one or more invalidation descriptors to the IQ. When the descriptor width is set to be 256-bit, hardware will treat bit 4 of this register as reserved along with bits 3:0.

Hardware interprets the IQ as empty when the head and tail registers are equal. Software interprets the IQ as full when the Tail Register is one behind the Head register (i.e., when all entries but one in the queue are used). This way software will write at most N-1 invalidation descriptors in a N entry IQ.

To enable queued invalidations, software must:

- Ensure all invalidation requests submitted to hardware through the register-based invalidation registers are completed. (i.e. no pending invalidation requests in hardware).
- Initialize the Invalidation Queue Tail Register (see Section 11.4.9.2) to zero.
- Setup the IQ address, size and descriptor width through the Invalidation Queue Address Register (see Section 11.4.9.3).
- Enable the queued invalidation interface through the Global Command Register (see Section 11.4.4.1). When enabled, hardware sets the QIES field in the Global Status Register (see Section 11.4.4.2).

When the queued invalidation is enabled, software must submit invalidation commands only through the IQ (and not through any register-based invalidation command registers).

Hardware fetches descriptors from the IQ in FIFO order starting from the Head Register if all of the following conditions are true. This is independent of the remapping hardware enable status (state of TES and IRES fields in the Global Status Register).

- QIES field in the Global Status Register is Set (indicating queued invalidation is enabled)
- IQ is not empty (i.e. Head and Tail pointer Registers are not equal)
- There is no pending Invalidation Queue Error or Invalidation Time-out Error (IQE and ITE fields in the Fault Status Register are both Clear)

Hardware implementations may fetch one or more descriptors together. However, hardware must increment the Invalidation Queue Head Register only after verifying the fetched descriptor to be valid. Hardware handling of invalidation queue errors are described in Section 6.5.2.11.

Once enabled, to disable the queued invalidation interface, software must:

- *Quiesce the invalidation queue*. The invalidation queue is considered quiesced when the queue is empty (head and tail registers equal) and the last descriptor completed is an Invalidation Wait Descriptor (which indicates no invalidation requests are pending in hardware).
- *Disable queued invalidation.* The queued invalidation interface is disabled through the Global Command Register. When disabled, hardware resets the Invalidation Queue Head Register to zero, and clears the QIES field in the Global Status Register.

The following subsections describe the various Invalidation Descriptors. Some of the descriptors are treated as invalid in certain address translation mode (see Table 26 for list of valid descriptors in each address translation mode). Type field (bits 11:9 and bits 3:0) of each descriptor identifies the descriptor type. Software must program the reserved fields in the descriptors as zero.

6.5.2.1 Context-cache Invalidate Descriptor

The Context-cache Invalidate Descriptor (*cc_inv_dsc*) allows software to invalidate the context-cache, there by forcing hardware to use the entries from root (scalable-mode root) and context (scalable-mode context) tables in system memory. The context-cache invalidate descriptor is a 128-bit descriptor. It must be padded with 128-bits of 0s in the upper bytes to create a 256-bit descriptor when the invalidation queue is configured for 256-bit descriptors (IQA_REG.DW=1). If a 128-bit version of this descriptor is submitted into an IQ that is setup to provide hardware with 256-bit descriptors or vice-versa it will result in an invalid descriptor error.

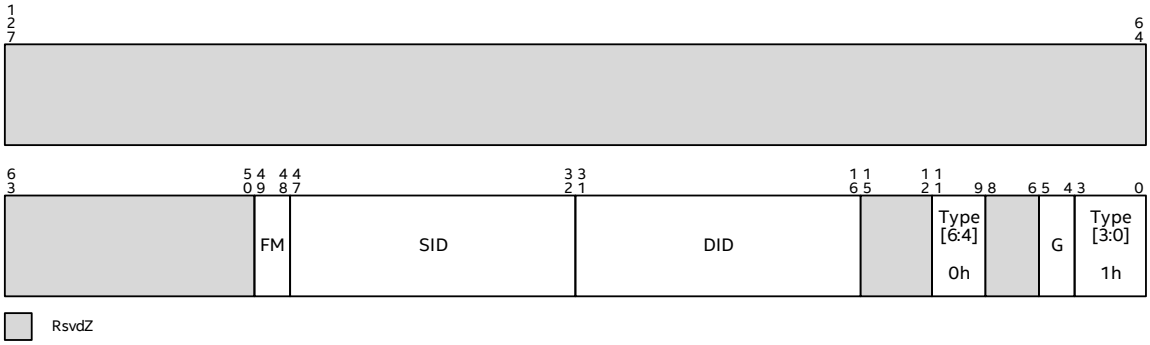


Figure 6-1. Context-cache Invalidate Descriptor (128-bit Version)

The context-cache invalidate descriptor includes the following parameters:

- *Type*: The concatenation of bits[11:9] and bits[3:0] form this 7-bit field. A value of 1h in this field indicates Context-cache Invalidate Descriptor.
- *Granularity (G)*: The G field indicates the requested invalidation granularity. The encoding of the G field is same as the CIRG field in the Context Command Register (described in Section 11.4.6.1). Hardware implementations may perform coarser invalidation than the granularity requested.
  - *Global Invalidation (01b)*: All context-cache entries cached at the remapping hardware are invalidated.
  - *Domain-Selective Invalidation (10b)*: Context-cache entries associated with the specified domain-id are invalidated. Since context-cache is not tagged by domain-id when operating in scalable mode (refer Section 6.2.1), domain-selective context-cache invalidations are processed by hardware as global invalidations when RTADDR_REG.TTM=01b.
  - *Device-Selective Invalidation* (11b): Context-cache entries associated with the specified device source-id and domain-id are invalidated. Since context-cache is not tagged by domain-

id when operating in scalable-mode (refer Section 6.2.1), domain-id field in device-selective context-cache invalidations are ignored by hardware when RTADDR_REG.TTM=01b.

— Reserved (00b): A descriptor with a reserved value for Granularity is treated as an invalid descriptor.

- *Domain-ID (DID)*: For domain-selective and device-selective invalidations, the DID field indicates the target domain-id. This field is ignored by hardware when operating in scalable mode (RTADDR_REG.TTM=01b).
- *Source-ID (SID)*: For device-selective invalidations, the SID field indicates the device source-id.
- *Function Mask (FM)*: Software may use the Function Mask to perform device-selective invalidations on behalf of devices supporting PCI Express Phantom Functions. This field indicates the number of least significant bits of the SID field to be masked for device-selective invalidations. All Context-cache entries associated with both matching domain-id and source-id less the bits indicated by this field, are invalidated. Refer to Table 21 for encodings of this field. The context-entries or scalable-mode context-entries corresponding to the source-ids specified through the SID and FM fields must have the same domain-id specified in the DID field.

Table 21. Invalidate Descriptor Function Mask Encodings

Function Mask Value	Source-ID Bits Masked	Source-IDs Invalidated
0	None	1
1	2	2
2	2:1	4
3	2:0	8

Hardware implementations reporting a write-buffer flushing requirement (RWBF=1 in Capability Register) must implicitly perform a write buffer flushing before invalidating the context-cache. Refer to Section 6.8 for write buffer flushing requirements.

Since information from the context-cache may be used to tag entries in the PASID-cache, IOTLB and paging-structure caches, software must always follow a context-cache invalidation with a PASID-cache invalidation (if operating in scalable mode), followed by an IOTLB invalidation. The granularity of the PASID-cache and IOTLB invalidation must be equal or greater than the preceding context-cache invalidation (e.g., A global context-cache invalidation must be followed by global PASID-cache invalidation and global IOTLB invalidation; A domain/device selective context-cache invalidation must be followed by Domain-selective PASID-cache invalidation and domain-selective or global IOTLB invalidation). Please refer to Section 6.5.3.3 for additional guidance on invalidations.

6.5.2.2 PASID-cache Invalidate Descriptor

The PASID-cache Invalidate Descriptor (*pc_inv_dsc*) allows software to invalidate the PASID-cache, forcing hardware to use entries from the scalable-mode PASID-directory/table in system memory for translating requests. This descriptor is a 256-bit descriptor and will result in an invalid descriptor error if submitted in an IQ that is setup to provide hardware with 128-bit descriptors (IQA_REG.DW=0).

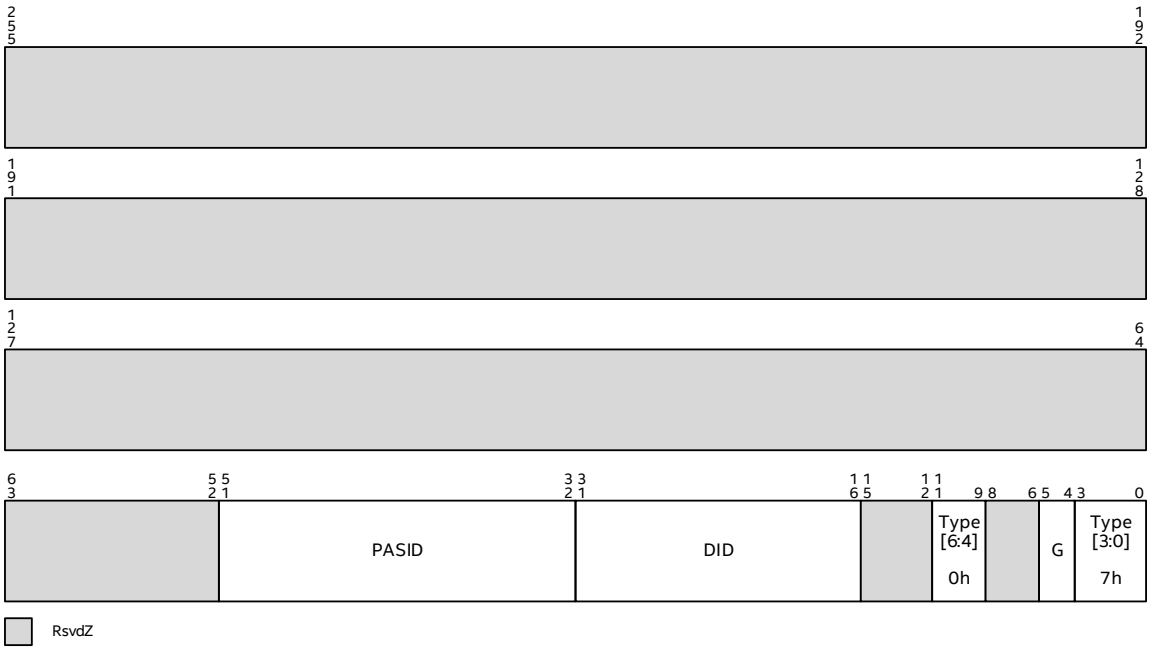


Figure 6-2. PASID-cache Invalidate Descriptor

The PASID-cache invalidate descriptor includes the following parameters:

- *Type*: The concatenation of bits[11:9] and bits[3:0] form this 7-bit field. A value of 7h in this field indicates PASID-cache Invalidate Descriptor.
- *Granularity (G)*: The G field indicates the requested invalidation granularity. Hardware implementations may perform coarser invalidation than the granularity requested. The encoding of the G field is as follows:
  - *Domain-Selective (00b)*: All PASID-cache entries associated with the specified domain-id are invalidated.
  - *PASID-Selective-within-Domain Invalidation* (01b): PASID-cache entries associated with the specified PASID value and the domain-id are invalidated.
  - *Global Invalidation* (11b): All PASID-cache entries are invalidated.
  - Reserved (10b): A descriptor with a reserved value for Granularity is treated as an invalid descriptor.
- *Domain-ID (DID)*: The DID field indicates the target domain-id. Hardware ignores bits 31:(16+N), where N is the domain-id width reported in the Capability Register.
- *PASID*: The PASID value indicates the target process-address-space to be invalidated. This field is ignored by hardware for Domain-selective invalidation granularity.

Since information from the PASID-cache may be used to tag the IOTLB and paging-structure caches, software must always follow a PASID-cache invalidation with an IOTLB invalidation. *Domain-Selective* granularity PASID-cache invalidation must be followed by *Domain-Selective* IOTLB invalidation. A

PASIDs-selective-within-Domain granularity PASID-cache invalidation must be followed by *PASID-selective* P_IOTLB invalidation. A *Global* granularity of PASID-cache invalidation must be followed by *Global* IOTLB invalidation. Please refer to Section 6.5.3.3 for additional guidance on invalidations.

6.5.2.3 IOTLB Invalidate

The IOTLB Invalidate Descriptor (*iotlb_inv_dsc*) allows software to invalidate the IOTLB and paging-structure-caches. This descriptor is expected to be used when software has changed second-stage tables and wants to invalidate affected cache entries. The IOTLB invalidate descriptor is a 128-bit descriptor. It must be padded with 128-bits of 0s in the upper bytes to create a 256-bit descriptor when the invalidation queue is configured for 256-bit descriptors (IQA_REG.DW=1). If a 128-bit version of this descriptor is submitted into an IQ that is setup to provide hardware with 256-bit descriptors or vice-versa it will result in an invalid descriptor error.

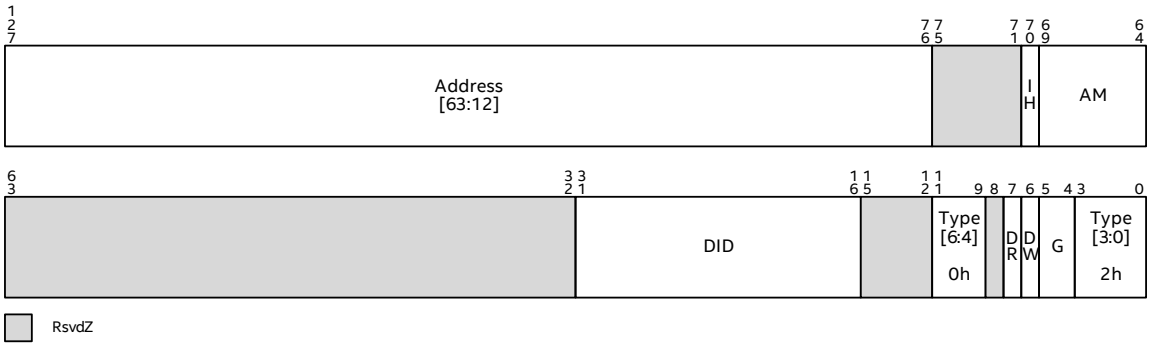


Figure 6-3. IOTLB Invalidate Descriptor (128-bit Version)

The descriptor includes the following parameters:

- *Type*: The concatenation of bits[11:9] and bits[3:0] form this 7-bit field. A value of 2h in this field indicates IOTLB Invalidate Descriptor.
- *Granularity (G)*: The G field indicates the requested invalidation granularity. Hardware implementations may perform coarser invalidation than the granularity requested. The encoding of the G field is same as the IIRG field in the IOTLB Invalidate Register (see Section 11.4.6.2):
  - *Global Invalidation (01b)*:
    - All IOTLB entries are invalidated.
    - All paging-structure-cache entries are invalidated.
  - *Domain-Selective Invalidation (10b)*:
    - IOTLB entries caching mappings associated with the specified domain-id are invalidated.
    - Paging-structure-cache entries caching mappings associated with the specified domain-id are invalidated.
  - *Page-Selective-within-Domain Invalidation (11b)*:
    - Remapping Hardware not supporting Page Selective Invalidation (CAP_REG.PSI=0) treats a descriptor with this Granularity as invalid.
    - IOTLB entries caching second-stage mappings (PGTT=010b) or pass-through (PGTT=100b) mappings associated with the specified domain-id and the input-address range are invalidated.

- IOTLB entries caching first-stage (PGTT=001b) or nested (PGTT=011b) mapping associated with specified domain-id are invalidated.
- When hardware is operating in legacy mode (RTADDR_REG.TTM=00b), IOTLB entries associated with specified domain-id and input-address range are invalidated.
- Paging-structure-cache entries caching mappings other than second-stage-mapping associated with the specified domain-id are invalidated.
- Paging-structure-cache entries caching second-stage mappings associated with the specified domain-id and the second-stage-input-address range are invalidated, if the Invalidation Hint (IH) field has value of 0. If the IH value is 1, the paging-structure-cache entries caching second-stage mappings are preserved.

— *Reserved (00b):*

- A descriptor with a reserved value for Granularity is treated as an invalid descriptor.

- *Drain Reads (DR)*: Software sets this flag to indicate hardware must drain read requests that are already processed by the remapping hardware, but queued within the Root-Complex to be completed. When the value of this flag is 1, hardware must drain the relevant reads before the next Invalidation Wait Descriptor (see Section 6.5.2.9) is completed. Section 6.5.4 describes hardware support for draining. Hardware implementations with Major Version 2 or higher (VER_REG) will ignore this flag and always drain relevant reads before the next Invalidation Wait Descriptor is completed.
- *Drain Writes (DW)*: Software sets this flag to indicate hardware must drain relevant write requests that are already processed by the remapping hardware, but queued within the Root-Complex to be completed. When the value of this flag is 1, hardware must drain the relevant writes before the next Invalidation Wait Descriptor is completed. Section 6.5.4 describes hardware support for draining. Hardware implementations with Major Version 2 or higher (VER_REG) will ignore this flag and always drain relevant writes before the next Invalidation Wait Descriptor is completed.
- *Domain-ID (DID)*: For domain-selective and page-selective-within-domain invalidations, the DID field indicates the target domain-id. Hardware ignores bits 31:(16+N), where N is the domain-id width reported in the Capability Register. This field is ignored by hardware for global invalidations. When RTADDR_REG.TTM=01b, domain-id field has a value in relevant scalable-mode PASID-table entry.
- *Invalidation Hint (IH)*: For page-selective-within-domain invalidations, the Invalidation Hint specifies if the second-stage mappings cached in the paging-structure-caches that controls the specified address/mask range needs to be invalidated or not. For software usages that updates only the leaf SS-PTEs, the second-stage mappings in the paging-structure-caches can be preserved by specifying the Invalidation Hint field value of 1. This field is ignored by hardware for global and domain-selective invalidations.
- *Address (ADDR)*: For page-selective-within-domain invalidations, the Address field in combination with the Address Mask field indicates a size-aligned region of second-stage page address mappings that need to be invalidated. Hardware ignores bits 127:(64+N), where N is the maximum guest address width (MGAW) supported. This field is ignored by hardware for global and domain-selective invalidations.
- *Address Mask (AM)*: For page-selective-within-domain invalidations, the Address Mask specifies the number of low order bits of the ADDR field that must be masked for the invalidation operation. This field enables software to request invalidation of contiguous mappings for size-aligned regions. Refer to Table 22 for encodings of this field. When invalidating a large-page translation, software must use the appropriate Address Mask value (0 for 4KByte page, 9 for 2-MByte page,

and 18 for 1-GByte page). Hardware implementations report the maximum supported address mask value through the Capability register.

Table 22. Invalidate Descriptor Address Mask Encodings

Address Mask Value	ADDR Bits Masked	4K Pages Invalidated
0	None	1
1	12	2
2	13:12	4
3	14:12	8
4	15:12	16
...	...	...

Hardware implementations reporting a write-buffer flushing requirement (RWBF=1 in Capability Register) must implicitly perform a write buffer flushing before invalidating the IOTLB. Refer to Section 6.8 for write buffer flushing requirements.

The table below summarizes what tags are used to find matching entries and invalidate them by various granularity of IOTLB invalidation.

Table 23. IOTLB Invalidation

Granularity	Scalable Mode: IOTLB Entries with PGTT=010b or PGTT=100b Legacy Mode: All IOTLB entries	IOTLB Entries with PGTT=011b or PGTT=001b (N/A for legacy mode)	FS-paging Structure Cache (N/A for legacy mode)	SS-paging Structure Cache
Global (01b)	All	All	All	All
Domain-Selective (10)	DID	DID	DID	DID
Page-Selective-within-Domain (11b)	DID, Address	DID	DID	If IH is 0, DID, Address Otherwise NA

6.5.2.4 PASID-based IOTLB Invalidate Descriptor (P_IOTLB)

The PASID-based-IOTLB Invalidate Descriptor (*p_iotlb_inv_dsc*) allows software to invalidate IOTLB and the paging-structure-caches. This descriptor is expected to be used when software has changed first-stage tables and wants to invalidate affected cache entries. This descriptor is a 256-bit descriptor and will result in an invalid descriptor error if submitted in an IQ that is setup to provide hardware with 128-bit descriptors (IQA_REG.DW=0).

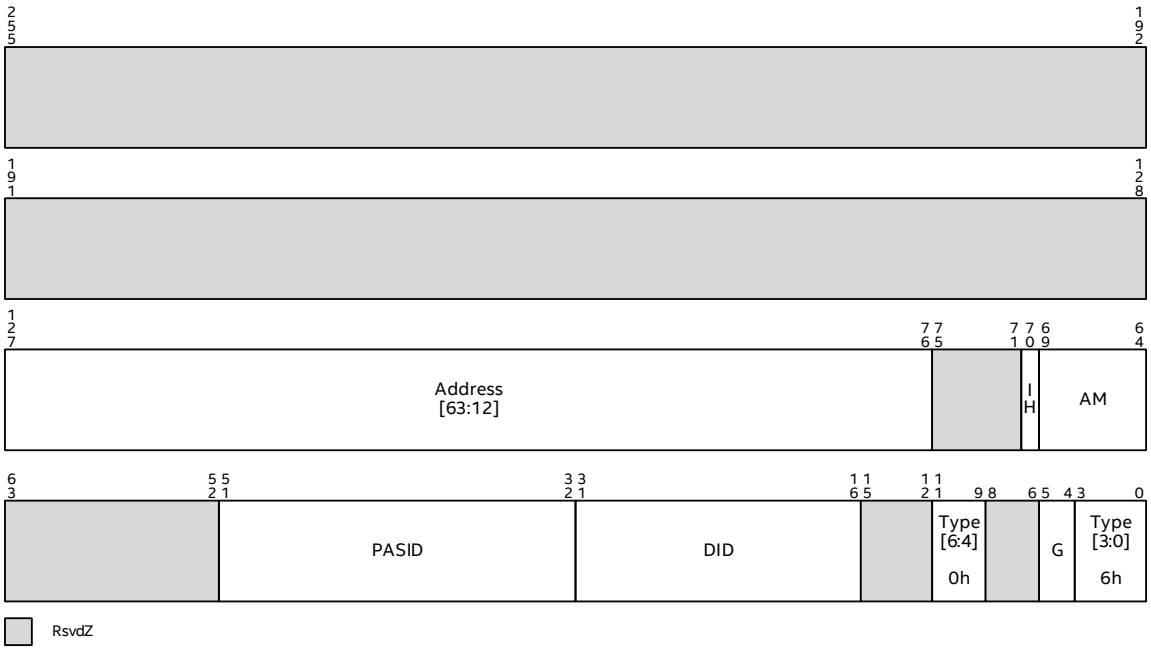


Figure 6-4. PASID-based-IOTLB Invalidate Descriptor

The descriptor includes the following parameters:

- *Type*: The concatenation of bits[11:9] and bits[3:0] form this 7-bit field. A value of 6h in this field indicates PASID-based-IOTLB Invalidate Descriptor.
- *Granularity (G)*: The G field indicates the requested invalidation granularity. Hardware implementations may perform coarser invalidation than the granularity requested. The encoding of the G field is as follows:
  - *PASID-selective (10b)*:
    - IOTLB entries caching mappings that are associated with the specified PASID and domain-id are invalidated.
    - Paging-structure-cache entries caching mappings that are associated with the specified PASID and domain-id are invalidated.
  - *Page-Selective-within-PASID (11b):*
    - IOTLB entries caching mappings that are associated with the specified PASID, domain-id, and the first-stage-input-address range are invalidated.
    - Paging-structure-cache entries caching mappings that are associated with the specified PASID, domain-id, and the first-stage-input-address range are invalidated if the Invalidation Hint (IH) field has value of 0. If the IH value is 1, the paging-structure-cache entries are preserved.
  - *Reserved (00b, 01b):*
    - A descriptor with a reserved value for Granularity is treated as an invalid descriptor.
- *Domain-ID (DID)*: The DID field indicates the target domain-id. Hardware ignores bits 31:(16+N), where N is the domain-id width reported in the Capability Register. When RTADDR_REG.TTM=01b, the domain-id field has a value in the relevant scalable-mode PASID-table entry.

- *PASID*: The PASID value indicates the target process-address-space to be invalidated.
- *Invalidation Hint (IH)*: For *page-selective-within-PASID* invalidations, the Invalidation Hint specifies if the first-stage and nested mappings cached in the paging-structure-caches that controls the specified address/mask range needs to be invalidated or not. For software usages that update only the leaf PTEs, the first-stage and nested mappings in the paging-structure-caches can be preserved by specifying the Invalidation Hint field value of 1. This field is ignored by hardware for other invalidation granularities.
- *Address (ADDR)*: For *page-selective-within-PASID* invalidations, the Address field in combination with the Address Mask field indicates a size-aligned region of first-level page address mappings that need to be invalidated. This field is ignored by hardware for *PASID-selective* invalidations.
- *Address Mask (AM)*: For *page-selective-within-PASID* invalidations, the Address Mask specifies the number of contiguous first-level 4-KByte pages that need to be invalidated. Refer to Table 22 for encodings of this field. When invalidating a large-page translation, software must use the appropriate Address Mask value (0 for 4KByte page, 9 for 2-MByte page, and 18 for 1-GByte page).

PASID-based-IOTLB invalidations are not required by hardware to invalidate PASID-cache entries, and second-stage mappings cached in paging-structure-caches.

PASID-based-IOTLB invalidations must always drain read and write requests that are already processed by the remapping hardware, but queued within the Root-Complex to be completed. Hardware must drain such outstanding read and write requests (to make them globally observable) before the next Invalidation Wait Descriptor (see Section 6.5.2.9) is completed. Section 6.5.4 further describes hardware support for draining.

The table below summarizes what tags are used to find matching entries and invalidate them by various granularity of PASID-based-IOTLB invalidation.

Table 24. PASID-based-IOTLB Invalidation

Granularity (G)	IOTLB Entries with PGTT=010b or PGTT=100b	IOTLB Entries with PGTT=011b or PGTT=001b	FS-paging Structure Cache	SS-paging Structure Cache
PASID-Selective (10)	DID, PASID	DID, PASID	DID, PASID	NA
Page-Selective-within-PASID (11b)	NA	DID, PASID, Address	If IH is 0, (DID, PASID, Address) Otherwise, NA	NA

6.5.2.5 Device-TLB Invalidate Descriptor

The Device-TLB Invalidate Descriptor (*dev_tlb_inv_dsc)* allows software to invalidate cached mappings used by requests-without-PASID from the Device-TLB on an endpoint device. The Device-TLB invalidate descriptor is a 128-bit descriptor. It must be padded with 128-bits of 0s in the upper bytes to create a 256-bit descriptor when the invalidation queue is configured for 256-bit descriptors (IQA_REG.DW=1). If a 128-bit version of this descriptor is submitted into an IQ that is setup to provide hardware with 256-bit descriptors or vice-versa it will result in an invalid descriptor error.

Device-TLB invalidation descriptors are reported as an invalid descriptor type for implementations reporting Device-TLB support as Clear in the Extended Capability Register (ECAP.DT=0).

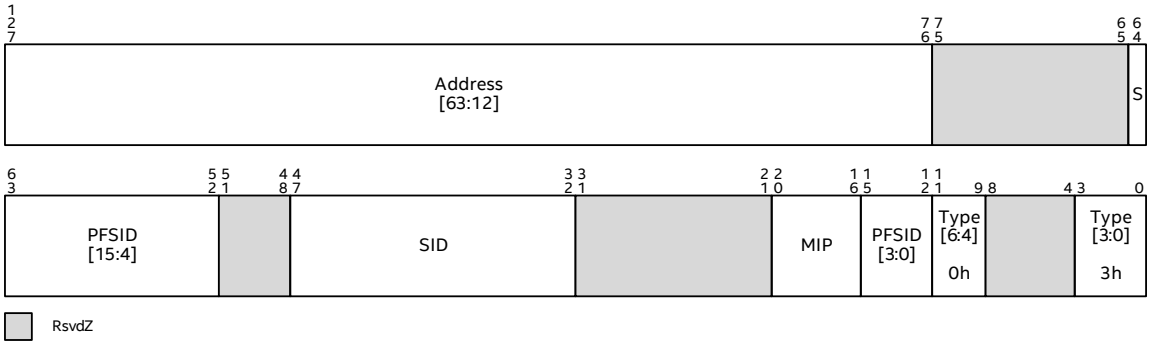


Figure 6-5. Device-TLB Invalidate Descriptor (128-bit Version)

The descriptor includes the following parameters:

- *Type*: The concatenation of bits[11:9] and bits[3:0] form this 7-bit field. A value of 3h in this field indicates Device-TLB Invalidate Descriptor.
- *Source-ID (SID)*: The SID field indicates the source-id of the endpoint device whose Device-TLB needs to be invalidated.
- *Address (ADDR)*: The address field indicates the starting input-address for the mappings that needs to be invalidated. The Address field is qualified by the S field.
- *Size (S)*: When this field is Clear, the target region to invalidate is 4-KByte. When this field is Set, the target region to invalidate is greater than 4-KByte. Refer to the ATS specification within PCI Express Base Specification Revision 4.0 or later for more details.
- *Max Invalidations Pending (MIP)*: This field is a hint to hardware to indicate the maximum number of pending invalidation requests the specified PCI Express endpoint device (Physical Function) can handle optimally. Endpoint devices are required to accept up to 32 pending invalidation requests, but the device may put back pressure on the I/O interconnect (e.g., PCI Express link) for multiple pending invalidations beyond Max Invalidations Pending. A value of 0h in MIP field indicates the device is capable of handling maximum (32) pending invalidation requests without throttling the link. Hardware implementations may utilize this field to throttle the number of pending invalidation requests issued to the specified device. Remapping hardware implementations reporting Pending Invalidation Throttling (DIT=1 in ECAP_REG) utilize this field to throttle the number of pending invalidation requests issued to the physical function specified in PFSID.
- *Physical Function Source-ID (PFSID)*: Remapping hardware implementations reporting Device-TLB Invalidation Throttling as not supported (DIT = 0 in ECAP_REG) treats this field as reserved. For implementations reporting Device-TLB Invalidation Throttling as supported (DIT=1 in ECAP_REG), if the Source-ID (SID) field specifies a Physical Function (PF), PFSID field specifies same value as the SID field; If the Source-ID (SID) field specifies a SR-IOV Virtual Function (VF), PFSID field specifies the Source-ID of the Physical Function (PF) associated with the Virtual Function (VF).

Since translation requests-without-PASID from a device may be serviced by hardware from the IOTLB, software must always request IOTLB invalidation (*iotlb_inv_dsc)* before requesting corresponding Device-TLB (*dev_tlb_inv_dsc)* invalidation.

6.5.2.6 PASID-based-Device-TLB Invalidate Descriptor

The PASID-based-Device-TLB Invalidate Descriptor (*p_dev_tlb_inv_dsc)* allows software to invalidate cached mappings used by requests-with-PASID from the Device-TLB on an endpoint device. This descriptor is a 256-bit descriptor and will result in an invalid descriptor error if submitted in an IQ that is setup to provide hardware with 128-bit descriptors (IQA_REG.DW=0).

PASID-based Device-TLB invalidation descriptors are reported as an invalid descriptor type for implementations reporting Device-TLB support as Clear in the Extended Capability Register (ECAP.DT=0).

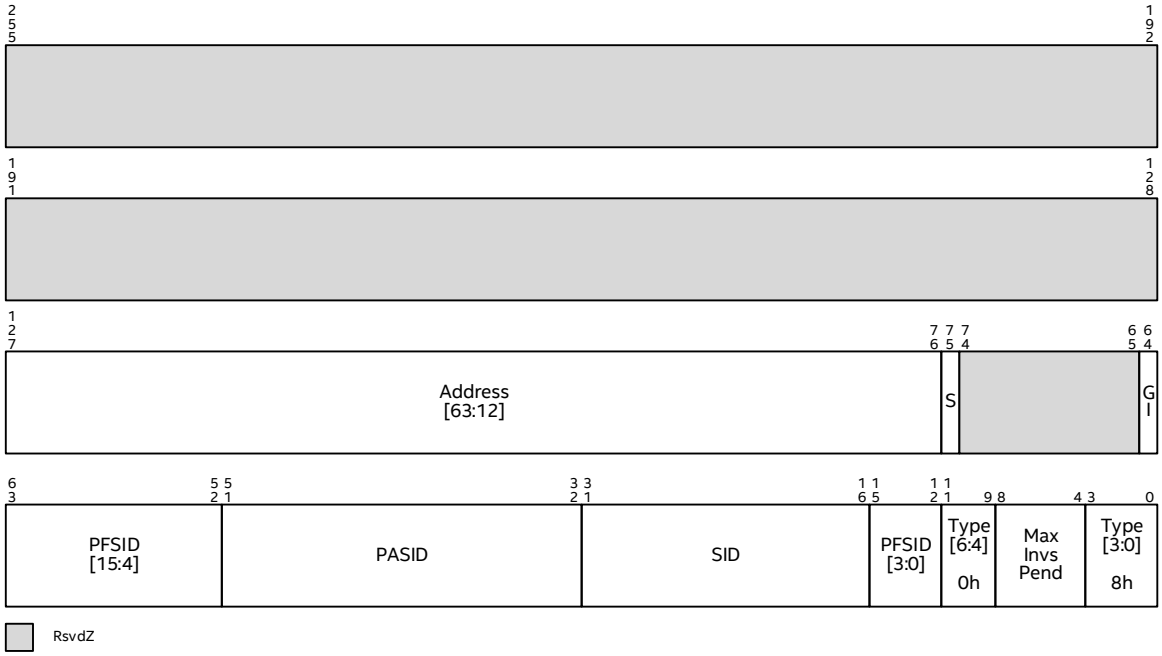


Figure 6-6. PASID-based-Device-TLB Invalidate Descriptor

The descriptor includes the following parameters:

- *Type*: The concatenation of bits[11:9] and bits[3:0] form this 7-bit field. A value of 8h in this field indicates PASID-based-Device-TLB Invalidate Descriptor.
- *Source-ID (SID)*: The SID field indicates the source-id of the endpoint device whose Device-TLB needs to be invalidated.
- *Global Invalidate (GI)*: The value of this field is copied into the "Global Invalidate" bit in the Invalidate Request Message body that is sent to DevTLB (aka ATC).
- *PASID*: The PASID value indicates the target process-address-space to be invalidated. This field is ignored by hardware if the value of G field is 1.
- *Address (ADDR)*: The address field indicates the starting input-address for the mappings that need to be invalidated. The address field is qualified by the S field.
- *Size (S)*: When this field is Clear, the target region to invalidate is 4-KByte. When this field is Set, the target region to invalidate is greater than 4-KByte. Refer to the ATS specification within PCI Express Base Specification Revision 4.0 or later for more details.
- *Max Invalidations Pending (MIP)*: This field is a hint to hardware to indicate the maximum number of pending invalidation requests the specified PCI Express endpoint device (Physical Function) can handle optimally. Endpoint devices are required to accept up to 32 pending invalidation requests, but the device may put back pressure on the I/O interconnect (e.g., PCI Express link) for multiple pending invalidations beyond Max Invalidations Pending. A value of 0h in the MIP field indicates the device is capable of handling maximum (32) pending invalidation requests without throttling the link. Hardware implementations may utilize this field to throttle the number of pending invalidation requests issued to the specified device. Remapping hardware implementations

reporting Pending Invalidation Throttling (DIT=1 in ECAP_REG) utilize this field to throttle the number of pending invalidation requests issued to the physical function specified in PFSID.

- *Physical Function Source-ID (PFSID)*: Remapping hardware implementations reporting Device-TLB Invalidation Throttling as not supported (DIT = 0 in ECAP_REG) treat this field as reserved. For implementations reporting Device-TLB Invalidation Throttling as supported (DIT=1 in ECAP_REG), if the Source-ID (SID) field specifies a Physical Function (PF), PFSID field specifies same value as the SID field; if the Source-ID (SID) field specifies a SR-IOV Virtual Function (VF), PFSID field specifies the Source-ID of the Physical Function (PF) associated with the Virtual Function (VF).

Since translation requests-with-PASID from a device may be serviced by hardware from the IOTLB:

- When operating in scalable mode (RTADDR_REG.TTM=01b), software must always request an appropriate IOTLB invalidation (*p_iotlb_inv_dsc* if the corresponding scalable-mode PASID-table entry is configured for first-level or nested translation, or *iotlb_inv_dsc* if the corresponding scalable-mode PASID-table entry is configured for second-stage only translation) before corresponding PASID-based-device-TLB (*p_dev_tlb_inv_dsc)* invalidation.

6.5.2.7 HPT Cache Invalidate Descriptor

The HPT Cache Invalidate Descriptor (*hpt_inv_desc*) allows software to invalidate one or more levels of the HPT caches, affecting both the HPT leaf caches and HPT paging-structure caches. This descriptor is used when software has altered the HPT table entries resulting in addition or removal of a device's permissions to one or more physical pages. This descriptor is a 256-bit descriptor and will result in an invalid descriptor error if submitted in an IQ that is configured with 128-bit descriptors (IQA_REG.DW=0).

Implementations reporting HPT Support as Clear in the Extended Capability Register treat HPT Cache Invalidate Descriptors as an invalid descriptor type. Refer to Section 11.4.9.9 for more details.

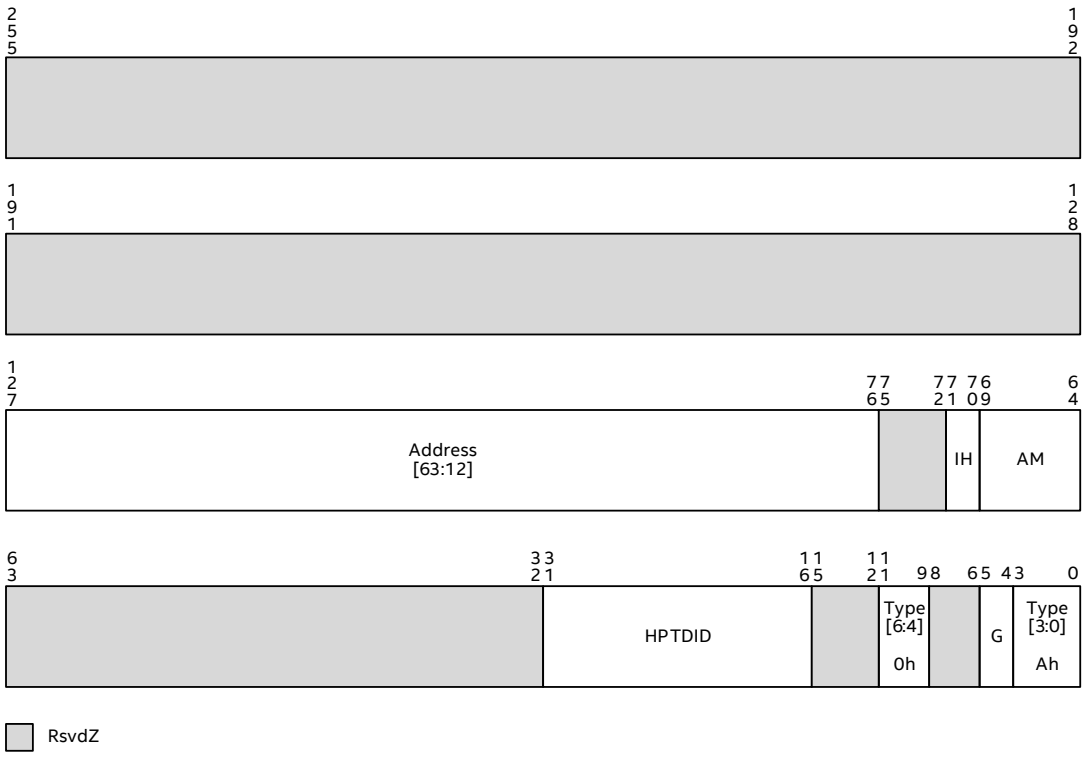


Figure 6-7. HPT Cache Invalidate Descriptor

The descriptor includes the following parameters:

- *Type*: The concatenation of bits[11:9] and bits[3:0] form this 7-bit field. A value of Ah in this field indicates HPT Cache Invalidate Descriptor.
- *Granularity (G)*: The G field indicates the requested invalidation granularity. Hardware implementations may perform coarser invalidation than the granularity requested. The following describe the supported encodings:
  - *Global Invalidation (01b)*:
    - All HPT cache entries are invalidated.
  - *Domain-Selective Invalidation (10b)*:
    - HPT cache entries at the levels indicated by the Invalidation Hint that are associated with the specified HPT Domain-ID are invalidated.
  - *Page-Selective-within-Domain Invalidation (11b)*:
    - HPT cache entries at the levels indicated by the Invalidation Hint, associated with the specified HPT Domain-ID, and within the address range indicated by Address and Address Mask are invalidated.
  - *Reserved (00b):*
    - A descriptor with a reserved value for Granularity is treated as an invalid descriptor.

- *HPT Domain-ID (HPTDID)*: The HPTDID field indicates the target HPT Domain-ID to be invalidated. Hardware ignores bits 31:(16+N), where N is the domain-id width reported in the Capability Register. The value in this field corresponds to the value in the HPT Domain ID field of the Scalable-mode PASID-table Entry used to cache the entry. This field is ignored by hardware when Granularity is Global.
- *Address Mask (AM)*: For page-selective-within-domain invalidations, the Address Mask specifies the number of contiguous 4KB host physical address regions to be invalidated. The encoding for the AM field is documented in Table 22. When invalidating a large-page translation, software must use the appropriate Address Mask value (0 for 4KByte page, 9 for 2-MByte page, and 18 for 1-GByte page).
- *Invalidation Hint (IH)*: Indicates which levels of the HPT caches to invalidate. When Granularity is Global Invalidation the Invalidation Hint field is ignored. The following describes the supported encodings:
  - L1 (00b):
    - Invalidate HPT cache entries corresponding to an HPTL1E.
  - L2 and L1(01b):
    - Invalidate HPT cache entries corresponding to an HPTL1E or HPTL2E.
  - L3,L2, and L1(10b):
    - Invalidate HPT cache entries corresponding to an HPTL3E, HPTL2E, or HPTL1E.
  - All Levels (11b):
    - Invalidate HPT cache entires corresponding to an HPTL4E, HPTL3E, HPTL2E, or HPTL1E.
- *Address (ADDR)*: When Granularity is Page-selective-within-domain, the Address field in combination with the Address Mask field indicates a size-aligned region of host physical addresses to invalidate. Hardware ignores bits 127:(64+N), where N is the Host Address Width (HAW). This field is ignored by hardware when Granularity is Global or Domain-Selective.

6.5.2.8 Interrupt Entry Cache Invalidate Descriptor

The Interrupt Entry Cache Invalidate Descriptor (*iec_inv_dsc*) allows software to invalidate the Interrupt Entry Cache. The Interrupt Entry Cache invalidate descriptor is a 128-bit descriptor. It must be padded with 128-bits of 0s in the upper bytes to create a 256-bit descriptor when the invalidation queue is configured for 256-bit descriptors (IQA_REG.DW=1). If a 128-bit version of this descriptor is submitted into an IQ that is setup to provide hardware with 256-bit descriptors or vice-versa it will result in an invalid descriptor error.

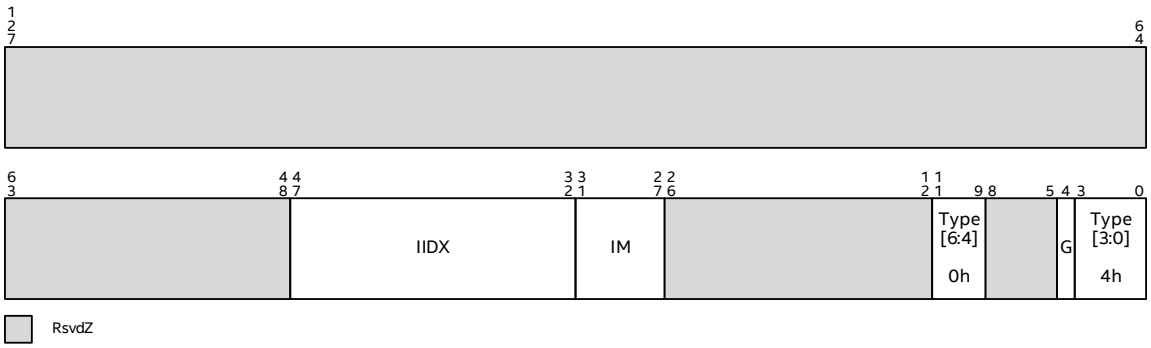


Figure 6-8. Interrupt Entry Cache Invalidate Descriptor (128-bit Version)

The descriptor includes the following parameters:

- *Type*: The concatenation of bits[11:9] and bits[3:0] form this 7-bit field. A value of 4h in this field indicates Interrupt Entry Cache Invalidate Descriptor.
- *Granularity (G):* This field indicates the granularity of the invalidation request. If Clear, a global invalidation of the interrupt-remapping cache is requested. If Set, a index-selective invalidation is requested.
- *Interrupt Index (IIDX):* This field specifies the index of the interrupt remapping entry that needs to be invalidated through a index-selective invalidation.
- *Index Mask (IM):* For index-selective invalidations, the index-mask specifies the number of contiguous interrupt indexes that needs to be invalidated. The encoding for the IM field is described below in Table 25.

Table 25. Index Mask Encodings

Index Mask Value	Index bits Masked	Mappings Invalidated
0	None	1
1	0	2
2	1:0	4
3	2:0	8
4	3:0	16
...	...	...

As part of IEC invalidation, hardware must drain interrupt requests that are already processed by the remapping hardware, but queued within the Root-Complex to be delivered to the processor. Section 6.5.5 describes hardware support for interrupt draining.

Hardware implementations reporting a write-buffer flushing requirement (RWBF=1 in Capability Register) must implicitly perform a write buffer flushing before invalidating the Interrupt Entry Cache. Refer to Section 6.8 for write buffer flushing requirements.

6.5.2.9 Invalidation Wait Descriptor

The Invalidation Wait Descriptor (*inv_wait_dsc*) descriptor allows software to synchronize with hardware for the invalidation request descriptors submitted before the wait descriptor. The invalidation wait descriptor is a 128-bit descriptor. It must be padded with 128-bits of 0s in the upper bytes to create a 256-bit descriptor when the invalidation queue is configured for 256-bit descriptors (IQA_REG.DW=1). If a 128-bit version of this descriptor is submitted into an IQ that is setup to provide hardware with 256-bit descriptors or vice-versa it will result in an invalid descriptor error.

The descriptor includes the following parameters:

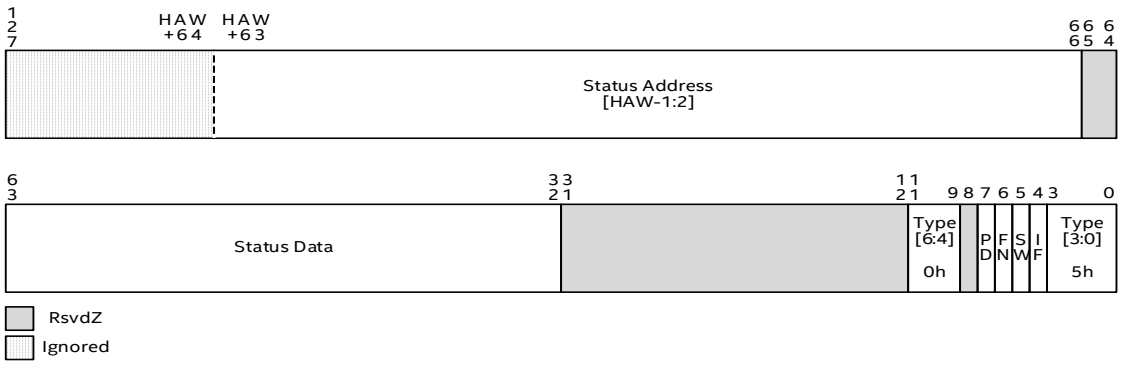


Figure 6-9. Invalidation Wait Descriptor (128-bit Version)

- *Type*: The concatenation of bits[11:9] and bits[3:0] form this 7-bit field. A value of 5h in this field indicates Invalidation Wait Descriptor.
- *Status Write (SW):* Indicate the invalidation wait descriptor completion by performing a coherent DWORD write of the value in the Status Data field to the address specified in the Status Address field.
- *Status Address and Data:* Status address and data is used by hardware to perform wait descriptor completion status write when the SW field is Set. Hardware ignores bits [127:HAW+64] of the Status Address field. Hardware behavior is undefined if the Status Address specified is not an address route-able to memory (such as peer address, interrupt address range of FEEx_xxxxh etc.). The Status Address and Data fields are ignored by hardware when the Status Write (SW) field is Clear.
- *Interrupt Flag (IF):* Indicate the invalidation wait descriptor completion by generating an invalidation completion event per the programming of the Invalidation Completion Event Registers. Section 6.5.2.10 describes details on invalidation event generation.
- *Fence Flag (FN):* When Set, indicates descriptors following the invalidation wait descriptor must be processed by hardware only after the invalidation wait descriptor completes.
- *Page-request Drain (PD):* Remapping hardware implementations reporting Page-request draining as not supported (PDS = 0 in ECAP_REG) treats this field as reserved. For implementations reporting Page-request draining as supported (PDS=1 in ECAP_REG), value of 1 in this field specifies the Invalidation wait completion status write (if SW=1) and Invalidation wait completion interrupt (if IF=1) must be ordered (visible to software) behind page-request descriptor (*page_req_dsc*) writes for all page requests received by remapping hardware before invalidation wait descriptor completion. For optimal performance, software must Set this field only if Page Request draining is required. Refer to Section 7.7 for remapping hardware behavior for page request draining.

Section 6.5.2.12 describes queued invalidation ordering considerations. Hardware completes an invalidation wait command as follows:

- If a status write is specified in the wait descriptor (SW=1), hardware performs a coherent write of the status data to the status address.
- If an interrupt is requested in the wait descriptor (IF=1), hardware sets the IWC field in the Invalidation Completion Status Register. An invalidation completion interrupt may be generated as described in the following section.

6.5.2.10 Hardware Generation of Invalidation Completion Events

The invalidation event interrupt generation logic functions as follows:

- At the time hardware sets the IWC field, it checks if the IWC field is already Set to determine if there is a previously reported invalidation completion interrupt condition that is yet to be serviced by software. If IWC field is already Set, the invalidation event interrupt is not generated.
- If the IWC field is not already Set, the Interrupt Pending (IP) field in the Invalidation Event Control Register is Set. The Interrupt Mask (IM) field is then checked and one of the following conditions is applied:
  - If IM field is Clear, invalidation completion event interrupt is generated along with clearing the IP field.
  - If IM field is Set, the invalidation completion event interrupt is not generated.

The following logic applies for interrupts held pending by hardware in the IP field:

- If IP field was Set when software clears the IM field, the invalidation completion event interrupt is generated along with clearing the IP field.
- If IP field was Set when software services the pending interrupt condition (indicated by IWC field in the Invalidation Completion Status Register being Clear), the IP field is cleared.

At the time an invalidation wait descriptor is completed by remapping hardware, if PD=1 in the wait descriptor, the invalidation completion status write (and/or invalidation completion event interrupt) that signal wait descriptor completion to software must push *page_req_desc* writes for all page requests already received by the remapping hardware. Refer to Section 7.7 for details on page request draining.

The invalidation completion event interrupt must push any in-flight invalidation completion status writes, including status writes that may have originated from the same *inv_wait_dsc* for which the interrupt was generated. Similarly, read completions due to software reading the Invalidation Completion Status Register (ICS_REG) or Invalidation Event Control Register (IECTL_REG) must push (commit) any in-flight invalidation completion event interrupts and status writes generated by the respective hardware unit.

The invalidation completion event interrupts are never subject to interrupt remapping.

6.5.2.11 Hardware Handling of Queued Invalidation Interface Errors

Hardware handles the various queued invalidation interface error conditions as follows:

- *Invalidation Queue Errors*: Hardware sets the IQE (Invalidation Queue Error) field in the Fault Status Register. Hardware provides additional information about the error in the Invalidation Queue Error Record Register fields if the register is supported. Refer to the description of the IQEI field in Section 11.4.9.9 for all possible conditions resulting in a Invalidation Queue Error.

Table 26. List of Valid Descriptor Types for Each Mode

Translation Table Mode (RTADDR_REG.TTM Value)	Descriptor Width = 0 (128-bit Descriptors)	Descriptor Width = 1 (256-bit Descriptors)
legacy mode (00b)	0x1...0x5	0x1...0x5
scalable mode (01b)	none	0x1...0xA
reserved mode (10b)	none	none
abort-dma mode (11b)	none	0x1...0xA

  A fault event may be generated based on the programming of the Fault Event Control Register. The Head pointer Register is not incremented, and references the descriptor associated with the queue error. No new descriptors are fetched from the Invalidation Queue until software clears the IQE field in the Fault Status Register. Software is expected to correct any error conditions that generated the fault event prior to clearing the IQE field. Tail pointer Register updates by software while the IQE field is Set does not cause descriptor fetches by hardware. Any invalidation commands ahead of the invalid descriptor that are already fetched and pending in hardware at the time of detecting the invalid descriptor error are completed by hardware as normal.

- *Invalid Device-TLB Invalidation Response*: If hardware receives an invalid Device-TLB invalidation response, hardware sets the Invalidation Completion Error (ICE) field in the Fault Status Register. A fault event may be generated based on the programming of the Fault Event Control Register. Hardware continues with processing of descriptors from the Invalidation Queue as normal.
- *Device-TLB Invalidation Response Time-out*: If hardware detects a Device-TLB invalidation response time-out, hardware frees the corresponding ITag and sets the ITE (Invalidation Time-out Error) field in the Fault Status Register. A fault event may be generated based on the programming of the Fault Event Control Register. No new descriptors are fetched from the Invalidation Queue until software clears the ITE field in the Fault Status Register. Tail pointer Register updates by software while the ITE field is Set does not cause descriptor fetches by hardware. At the time ITE field is Set, hardware aborts any *inv_wait_dsc* commands pending in hardware and does not increment the Invalidation Queue Head register. When software clears the ITE field in the Fault Status Register, hardware fetches descriptor pointed by the Invalidation Queue Head register. Any invalidation responses received while ITE field is Set are processed as normal (as described in Section 4.3). Since the time-out could be for any (one or more) of the

pending *dev_tlb_inv_dsc* commands, execution of all descriptors including and behind the oldest pending *dev_tlb_inv_dsc* is not guaranteed.

6.5.2.12 Queued Invalidation Ordering Considerations

Hardware must support the following ordering considerations when processing descriptors fetched from the Invalidation Queue:

- Hardware must execute an IOTLB invalidation descriptor (*iotlb_inv_dsc*) or PASID-based-IOTLB invalidation descriptor (*p_iotlb_inv_dsc)* only after all Context-cache invalidation descriptors (*cc_inv_dsc*) and PASID-cache invalidation descriptors (*pc_inv_dsc*) ahead of it in the Invalidation Queue are completed.
- Hardware must execute a PASID-cache invalidation descriptors (*pc_inv_dsc*) only after all Context-cache invalidation descriptors (*cc_inv_dsc*) ahead of it in the Invalidation Queue are completed.
- Hardware must execute a Device-TLB invalidation descriptor (*dev_tlb_inv_dsc)* only after all IOTLB invalidation descriptors (*iotlb_inv_dsc)* and Interrupt Entry Cache invalidation descriptors (*iec_inv_dsc)* ahead of it in the Invalidation Queue are completed.
- Hardware must execute an PASID-based-Device-TLB Invalidation descriptor (*p_dev_tlb_inv_dsc)* only after all IOTLB invalidation descriptors (*iotlb_inv_dsc)*, PASID-based-IOTLB invalidation descriptors (*p_iotlb_inv_dsc)* and Interrupt Entry Cache invalidation descriptors (*iec_inv_dsc)* ahead of it in the Invalidation Queue are completed.
- Hardware must report completion of an Invalidation Wait Descriptor (*inv_wait_dsc)* only after at least all the descriptors ahead of it in the Invalidation Queue and behind the previous *inv_wait_dsc* are completed.
- If the Fence (FN) flag is 0 in a *inv_wait_dsc,* hardware may execute descriptors following the *inv_wait_dsc* before the wait command is completed. If the Fence (FN) flag is 1 in a *inv_wait_dsc,* hardware must execute descriptors following the *inv_wait_dsc* only after the wait command is completed.
- When a Device-TLB invalidation or PASID-based-Device-TLB invalidation time-out is detected, hardware must not complete any pending *inv_wait_dsc* commands.

6.5.3 Invalidation Considerations

The following subsections describe additional details and considerations on invalidations.

6.5.3.1 Implicit Invalidation on Page Requests

In addition to the explicit invalidation through invalidation commands (see Section 6.5.1 and Section 6.5.2) identified above, page requests from endpoint devices invalidate entries in the IOTLBs and paging-structure caches.

When operating in scalable mode (RTADDR_REG.TTM=01b), page requests will traverse the translation tables to obtain the value of the PGTT and DID fields in the scalable-mode PASID-table entry. Page requests without PASID will additionally obtain the PASID value from the RID_PASID field of the scalable-mode context-entry. After this, hardware will internally generate appropriate IOTLB invalidation based on the value of the PGTT field as shown in table below.

Table 27. Implicit Invalidation on Page Request

PGTT	Invalidation	Operand Values
first-stage (001b)	Page-selective-within-PASID P_IOTLB invalidation	DID, PASID, Address[63:12], AM=0, IH=0
second-stage (010b)	Page-selective-within-Domain IOTLB invalidation	DID, Address[63:12], AM=0, IH=0, DR=0, DW=0
nested (011b)	Domain-selective IOTLB invalidation	DID, Address[63:12], AM=0, IH=0, DR=0, DW=0
pass-through (100b)	Page-selective-within-Domain IOTLB invalidation	DID, Address[63:12], AM=0, IH=0, DR=0, DW=0

If a page request is unable to traverse the translation tables and obtain necessary information for IOTLB invalidation, hardware will not generate any implicit invalidation and will manufacture a Page Group Response with Status=IR back to the device without writing a page request in the Page Request Queue.

Software is currently not required to issue invalidations when upgrading page permissions from read-only to read-write. It is possible a device may encounter a recoverable fault due to read-only translations in one of the remapping caches. When software services such a recoverable fault, it may not send invalidation, leaving stale read-only permissions in remapping caches. To remove such stale mappings, remapping hardware performs a suitable implicit IOTLB invalidation based on the PGTT field associated with the Page Request. Software issuing an explicit invalidation after a handling Page Request and before submitting a Page Response does not need to depend on implicit invalidations described earlier.

6.5.3.2 Caching Fractured Translations

Some implementations may choose to cache multiple smaller-page IOTLB entries (fractured translations) for a translation specified by the paging structures to use a page larger than 4 KBytes. There is no way for software to be aware that multiple translations for smaller pages have been used for a large page. Since software is required to always specify the appropriate Address Mask value to cover the address range to be invalidated (Address Mask value of 0 for invalidating a 4-KByte page, 9 for invalidating a 2-MByte page, and 18 for invalidating a 1-GByte page) in the IOTLB invalidation commands, these commands naturally invalidate all IOTLB entries corresponding a large-page translation.

6.5.3.3 Guidance to Software for Invalidations

Table 28 below summarizes recommended invalidation for typical software usage model with additional details in the section.

Note: Global Device-TLB invalidation is Device-TLB invalidation with S=1 and Addr[63:12]=0x7FFFFFFF_FFFFF.

Note: Invalidations described in the table are required when the entry being changed is present or when Caching Mode (CM) is reported as 1. The one exception is that the changes to first-stage tables require invalidation only when a present entry is being changed irrespective of the value of CM. See Section 6.1 and Section 6.2.4.1 for more information.

Table 28. Guidance to Software for Invalidations

	Legacy	Scalable
Changes to a scalable-mode root-table entry	NA	• Global context-cache invalidation • Global PASID-cache invalidation • Global IOTLB invalidation • Global Device-TLB invalidation to all affected functions
Changes to a scalable-mode context-table entry		• Device-selective context-cache invalidation • Domain-selective PASID-cache invalidation to affected domains (can be skipped if all PASID entries were not-present and CM=0) • Domain-selective IOTLB invalidation to affected domains • Global Device-TLB invalidation to affected functions.
Changes to a scalable-mode PASID directory entry		• PASID-selective-within-Domain PASID-cache invalidation to affected PASIDs • Domain-selective IOTLB invalidation to affected Domains • Global Device-TLB invalidation to affected functions

Any one of the following changes to a scalable-mode PASID-table entry corresponding to RID_PASID:
• SSADE transition from 0 to 1 in a scalable-mode PASID-table entry with PGTT value of Second-stage or Nested
• Present bit transition from 1 to 0 in a scalable-mode PASID-table entry
		• PASID-selective-within-Domain PASID-cache invalidation
		• Appropriate IOTLB invalidation
		if (PGTT=SS or Nested) {
		Domain-selective IOTLB invalidation
		} else {
		PASID-selective PASID-based IOTLB invalidation }
		• Global Device-TLB invalidation to affected functions

Any one of the following changes to a scalable-mode PASID-table entry **NOT** corresponding to RID_PASID:
• SSADE transition from 0 to 1 in a PASID-table entry with PGTT value of Second-stage or Nested
• Present bit transition from 1 to 0 in a PASID-table entry
		• PASID-selective-within-Domain PASID-cache invalidation
		• Appropriate IOTLB invalidation
		if (PGTT=SS or Nested) {
		Domain-selective IOTLB invalidation
		} else {
		PASID-selective PASID-based IOTLB invalidation }
		• PASID-based Device-TLB invalidation (with S=1 and Addr[63:12]=0x7FFFFFFF_FFFFF) to affected functions

Table 28. Guidance to Software for Invalidations

	Legacy	Scalable
Changes to fields other than SSADE and P in a scalable-mode PASID-table entry corresponding to RID_PASID	NA	• PASID-selective-within-Domain PASID-cache invalidation • PASID-selective P_IOTLB invalidation • Global Device-TLB invalidation to affected functions
Changes to fields other than SSADE and P in a scalable-mode PASID-table entry **NOT** corresponding to RID_PASID		• PASID-selective-within-Domain PASID-cache invalidation • PASID-selective P_IOTLB invalidation • PASID-based Device-TLB invalidation (with S=1 and Addr[63:12]=0x7FFFFFFF_FFFFF) to affected functions
Re-use Domain-ID in a scalable-mode PASID-table entry		• Domain-selective PASID-cache invalidation • Domain-selective IOTLB invalidation • Global Device-TLB invalidation to all functions within incoming domain

Changes to First-stage Page-tables		For every PASID using the modified First-stage table issue Page-selective-within-PASID P_IOTLB invalidation
		
		For all affected functions (i.e, functions using the modified FS-table) {
		If (Function's RID_PASID entry uses the modified FS-table) {
		Issue Device-TLB invalidation to the function
		} else {
		If (Function's other PASID entries use the modified FS-table) {
		Issue PASID-based Device-TLB invalidation to the function for all affected PASIDs
		}
		}
		}

Changes to Second-stage page-tables	For every Domain using the modified Second-stage table issue Page-selective-within-Domain IOTLB invalidation
	For all affected functions (i.e, functions using the modified SS-table) {
	Issue Device-TLB invalidation to the function
	}
		For every Domain using the modified Second-stage table issue Page-selective-within-Domain IOTLB invalidation
		
		For all affected functions (i.e, functions using the modified SS-table) {
		If (Function's RID_PASID entry uses the modified SS-table) {
		Issue Device-TLB invalidation to the function
		} else {
		If (Function's other PASID entries use the modified SS-table) {
		Issue PASID-based Device-TLB invalidation to the function for all affected PASIDs
		}
		}
		}

Table 28. Guidance to Software for Invalidations

	Legacy	Scalable
Changes to a root-table entry	• Global context-cache invalidation • Global IOTLB invalidation • Global Device-TLB invalidation to all affected functions.	NA
Changes to a context-table entry	• Device-selective context-cache invalidation • Domain-selective IOTLB invalidation • Global Device-TLB invalidation to all affected functions	
Re-use Domain-ID in a context-table entry	• Domain-selective context-cache invalidation • Domain-selective IOTLB invalidation • Device-TLB invalidation to all functions previously within the domain (It may be more convenient for software to issue this invalidation when the Domain-ID stops being used rather than when it is re-used.)	

The Table 28 above provides general guidance on various invalidations that software must perform after certain changes to the DMA remapping structures.

The following recommendations provide details on the invalidation that software should perform when modifying first-stage or second-stage paging entries. Software should generally use *page-selective-within-PASID* P_IOTLB invalidation when modifying first-stage table paging entries, and *page-selective-within-domain* IOTLB invalidation when modifying second-stage table paging entries.

- If software modifies a paging-structure entry that identifies the final page frame for a page number (either a PTE or a paging-structure entry in which the PS flag is 1), it should execute the *page-selective* type of IOTLB invalidation command for any address with a page number whose translation uses that paging-structure entry, with an address-mask matching the page frame size. (Address Mask value of 0 for 4-KByte page, 9 for 2-Mbyte page, and 18 for 1-GByte page). If no intermediate paging-structures entries with PS=0 are modified, the invalidation command can specify an Invalidation Hint (IH) as 1.

  If the same paging-structure entry may be used in the translation of different page numbers (see Section 6.2.8), software should perform the *page-selective* type of IOTLB invalidation for addresses with each of those page numbers, with an Invalidation Hint (IH) value of 0. Alternatively, software could use a coarser-grained IOTLB invalidation command (see Invalidation Granularity description in Section 6.5.2.4).
- If software modifies a paging-structure entry that references another paging structure, it may use one of the following approaches depending upon the type and number of translations controlled by the modified entry:
  - Execute *page-selective* type of IOTLB invalidation command for any addresses with each of the page numbers with translations that will use the entry. These invalidations must specify an Invalidation Hint (IH) value of 0 (so that it invalidates the paging-structure caches). However, if no page numbers that will use the entry have translations (e.g., because the P flags are 0 in all entries in the paging structure referenced by the modified entry), it remains necessary to execute the *page-selective* type of IOTLB invalidation command at least once.
  - If software modifies many first-stage page table entries it may be more performant to execute a *PASID-selective* P_IOTLB invalidation command or even a *Domain-selective* IOTLB invalidation command.

— If software modifies many second-stage page table entries it may be more performant to execute a *Domain-selective* IOTLB invalidation command.

- If the nature of the paging structures is such that a single entry may be used for multiple purposes (see Section 6.2.8), software should perform invalidations for all of these purposes. For example, if a single entry might serve as both a PDE and PTE, it may be necessary to execute the *page-selective* type of IOTLB invalidation command with two (or more) input-addresses; one that uses the entry as a PDE, and one that uses it as a PTE. Alternatively, software could use a *PASID-selective* P_IOTLB invalidation or a *Domain-selective* IOTLB invalidation.
- As noted in Section 6.2.4, the IOTLB may subsequently contain multiple translations for the address range if software modifies the paging structures so that the page size used for a 4-KByte range of input-addresses changes. A reference to an input-address in the address range may use any of these translations.

  Software wishing to prevent this uncertainty should not write to a paging structure entry in a way that would change, for any input-address, both the page size and either the page frame, access rights, or other attributes. It can instead use the following algorithm: first clear the P flag in the relevant paging-structure entry (e.g., PDE); then invalidate any translations for the affected input-addresses (see above); and lastly, modify the relevant paging-structure entry to set the P flag and establish modified translation(s) for the new page size.

6.5.3.4 Guidance to Software for Invalidations with HPT Enabled

Table 29 below summarizes additional recommended invalidation for typical software when HPT Enable is Set in scalable-mode context entry. This guidance only applies to scalable mode operation and must be performed after any invalidation recommended in Table 28.

Table 29. Guidance to Software for Invalidations with HPT Enabled

	Scalable
Any one of the following changes to a present scalable-mode context-table entry: • HPTE transition from 1 to 0 in a scalable-mode context-table entry • EPTR Transition from 1 to 0 in a scalable-mode context-table entry	• Domain-selective HPT-cache invalidation for all HPT domains the device can access (can be skipped if all PASID entries were not-present)
Any changes to HPTSZ, HPTPTR, or HPTDID in a present scalable-mode PASID-table entry	• Domain-selective HPT-cache invalidation to All Levels.
Changes to an initialized HPT paging-structure. Refer to Section 6.2.6.2 for HPT initialization requirements.	• Page-selective-within-Domain HPT-cache invalidation

When removing/reducing permissions, software should update and invalidate translation tables before updating HPT table entries. This avoids unintended faults if the device accesses an affected physical address before the invalidations have completed. Below are recommended steps:

1. Change translation tables as needed
2. IOTLB invalidation, if required by translation table changes
3. DevTLB invalidation, if required by translation table changes
4. Change HPT to remove/reduce permissions as needed
5. HPT invalidation

A device issuing translated requests may be permitted to access the data until remapping hardware processes an HPT invalidation is followed by the completion of an invalidation wait descriptor.

When adding permissions, software should update and invalidate the HPT table entries before updating translation tables. This avoids unintended faults if the device immediately accesses the permitted addresses.

1. Change HPT to add permissions as needed
2. HPT invalidation
3. Change translation tables as needed
4. IOTLB invalidation, if required by translation table changes
5. DevTLB invalidation, if required by translation table changes

6.5.3.5 Optional Invalidation

Intel x86 CPU architecture allows software to skip invalidations when "upgrading" page permissions. This is possible as all CPUs support page faults, and stale TLB entries are removed on encountering a page fault. However, not all devices support page faults and explicit invalidations may be required when software modifies page permissions. This section describes the cases where software may choose to skip invalidations. In all cases not listed, invalidation should be performed any time page table permissions are changed.

First Stage Paging Entry Permission Upgrades:

- For paging structure entries used by devices (irrespective of page fault support in device) invalidation may be skipped for the following cases:
  - Paging structure entry is modified to change the P flag from 0 to 1
- For paging structure entries with P flag set to 1 and only used by devices that support page fault, invalidation may be skipped for the following cases:
  - Paging-structure entry is modified to change the R/W flag from 0 to 1
  - Paging-structure entry is modified to change the U/S flag from 0 to 1
  - Paging-structure entry is modified to change the Accessed flag from 0 to 1
  - Paging-structure entry is modified to change the Dirty flag from 0 to 1

Second Stage Paging Entry Permission Upgrades:

- When Caching Mode is 0:
  - For paging structure entries used by devices (irrespective of page fault support in device) invalidation may be skipped for the following cases:
    - Paging structure entry is modified to change the P flag from 0 to 1
  - For paging structure entry with P flag set to 1 and only used by devices that support page fault, invalidation may be skipped for the following cases:
    - Paging-structure entry with R flag as 1, is modified to change the W flag from 0 to 1.
- When Caching Mode is 1, no invalidations of second stage paging entries can be skipped.

6.5.3.6 Delayed Invalidation

Required invalidations may be delayed under some circumstances with first-level paging. Software developers should understand that, between the modification of a paging-structure entry and execution of the IOTLB invalidation command, the hardware may use translations based on either the old value or the new value of the paging-structure entry. The following items describe some of the potential consequences of delayed invalidation:

- If a paging-structure entry is modified to change the P flag from 1 to 0, an access to an input-address whose translation is controlled by this entry may or may not cause a translation fault.

- If a paging-structure entry is modified to change the R/W flag from 0 to 1, write accesses to input-addresses whose translation is controlled by this entry may or may not cause a translation fault.
- If a paging-structure entry is modified to change the U/S flag from 0 to 1, user-mode accesses to input-addresses whose translation is controlled by this entry may or may not cause a translation fault.

In some cases, the consequences of delayed invalidation may not affect software adversely. For example, when freeing a portion of the process address space (by marking paging-structure entries "not present"), IOTLB invalidation command may be delayed if software does not re-allocate that portion of the process address space or the memory that had been associated with it. However, because of speculative execution by devices (or errant software), there may be accesses to the freed portion of the process address space before the invalidations occur. In this case, the following can happen:

- Reads can occur to the freed portion of the process address space. Therefore, invalidation should not be delayed for an address range that has side effects for reads from devices (e.g., mapped to MMIO).
- The hardware may retain entries in the IOTLBs and paging-structure caches for an extended period of time. Software should not assume that the hardware will not use entries associated with a input-address simply because time has passed.
- As noted in Section 6.2.5, the hardware may create an entry in a paging-structure cache even if there are no translations for any input-address that might use that entry. Thus, if software has marked "not present" all entries in the page table, the hardware may subsequently create a PDE-cache entry for the PDE that references that page table (assuming that the PDE itself is marked "present").
- If software attempts to write to the freed portion of the input-address space, the hardware might not generate a translation fault. (Such an attempt would likely be the result of a software error.) For that reason, the page frames previously associated with the freed portion of the process address space should not be reallocated for another purpose until the appropriate invalidations have been performed.

6.5.4 Draining of Requests to Memory

Requests from devices that are already processed by the remapping hardware, but queued within the Root-Complex to be completed to memory are referred as non-committed requests. Draining refers to hardware pushing (committing) these requests to the global ordering point. Hardware implementations report support for draining through the Capability Registers.

A write request to system memory is considered drained when the effects of the write are visible to processor accesses to addresses targeted by the write request. A read request to system memory is considered drained when the Root-Complex has finished fetching all of its read response data from memory.

Requirements for draining are described below:

- Draining applies only to requests to memory and do not guarantee draining of requests to peer destinations.
- Draining applies only for untranslated requests (AT=00b), including those processed as pass-through by the remapping hardware.
- Draining of translated requests (AT=10b) requires issuing a Device-TLB invalidation command to the endpoint device. Endpoint devices supporting Address Translation Services (ATS) are required to wait for pending translated read responses (or keep track of pending translated read requests and discard their read responses when they arrive) before issuing the ATS invalidation completion message. This effectively guarantees draining of translated read requests. The ATS invalidation completion message is issued on the posted channel and pushes all writes from the device (including any translated writes) ahead of it. To ensure proper write draining of translated

requests, remapping hardware must process ATS invalidation completion messages after all preceding writes are considered drained.

- Read and write draining of untranslated requests are required when remapping hardware status changes from disabled to enabled. The draining must be completed before hardware sets the TES field in Global Status Register (which indicates remapping hardware is enabled). Hardware implementations may perform draining of untranslated requests when remapping hardware status changes from enabled to disabled.
- Read and write draining of untranslated requests are performed on IOTLB invalidation requests specifying Drain Read (DR) and Drain Write (DW) flags respectively. For IOTLB invalidations submitted through the IOTLB Invalidate Register (IOTLB_REG), draining must be completed before hardware clears the IVT field in the register (which indicates invalidation completed). For IOTLB invalidations submitted through the queued invalidation interface, draining must be completed before the next Invalidation Wait Descriptor (*inv_wait_dsc*) is completed by hardware.
  - For global IOTLB invalidation requests specifying DMA read/write draining, all non-committed DMA read/write requests queued within the Root-Complex are drained.
  - For domain-selective IOTLB invalidation requests specifying read/write draining, hardware only guarantees draining of non-committed read/write requests to the domain specified in the invalidation request.
  - For page-selective IOTLB invalidation requests specifying read/write draining, hardware only guarantees draining of non-committed read/write requests with untranslated address overlapping the address range specified in the invalidation request and to the specified domain.
- Read and write draining of untranslated requests are performed on all PASID based IOTLB invalidation requests, where draining is completed before the next Invalidation Wait Descriptor (*inv_wait_dsc*) is completed by hardware.

6.5.5 Interrupt Draining

Interrupt requests that are already processed by the remapping hardware, but queued within the Root-Complex to be completed are referred as non-committed interrupt requests. Interrupt draining refers to hardware pushing (committing) these interrupt requests to the appropriate processor's interrupt controller (Local xAPIC). An interrupt request is considered drained when the interrupt is accepted by the processor Local xAPIC (for fixed and lowest priority delivery mode interrupts this means the interrupt is at least recorded in the Local xAPIC Interrupt Request Register (IRR)).

Requirements for interrupt draining are described below:

- Interrupt draining applies to all non-committed interrupt requests, except Compatibility format interrupt requests processed as pass-through on Intel® 64 platforms.
- Interrupt draining is required when interrupt-remapping hardware status changes from disabled to enabled. The draining must be completed before hardware sets the IES field in Global Status Register (indicating interrupt-remapping hardware is enabled). Hardware implementations may perform interrupt draining when interrupt-remapping hardware status changes from enabled to disabled.
- Interrupt draining is performed by remapping hardware after Interrupt Entry Cache (IEC) invalidation requests. For IEC invalidations submitted through the queued invalidation interface, interrupt draining must be completed before the next Invalidation Wait Descriptor is completed by hardware.
  - For global IEC invalidation requests, all non-committed interrupt requests queued within the Root-Complex are drained.
  - For index-selective IEC invalidation requests, hardware only guarantees draining of non-committed interrupt requests referencing interrupt indexes specified in the invalidation request.
- The Root-Complex considers an interrupt request as drained when it receives acknowledgment from the processor complex. Interrupt draining requires processor complex to ensure the

interrupt request received is accepted by the Local xAPIC (for fixed interrupts, at least recorded in the IRR) before acknowledging the request to the Root-Complex.

6.6 Set Root Table Pointer Operation

Software must always perform a Set Root-Table Pointer operation before enabling or re-enabling (after disabling) remapping hardware.

For implementations reporting the Enhanced Set Root Table Pointer Support (ESRTPS) field as Clear, on a 'Set Root Table Pointer' operation, software must perform a global invalidate of the context-cache, PASID-cache (if applicable), IOTLB, and HPT-cache (if applicable), in that order. This is required to ensure hardware references only the remapping structures referenced by the new root table pointer and not stale cached entries.

For implementations reporting the Enhanced Set Root Table Pointer Support (ESRTPS) field as Set, as part of 'Set Root Table Pointer' operation, hardware performs global invalidation on all DMA remapping translation caches and hence software is not required to perform additional invalidations.

If software sets the root-table pointer while remapping hardware is active (TES=1 in Global Status register), software must ensure the structures referenced by the new root-table pointer provide identical remapping results as the structures referenced by the previous root-table pointer so that in-flight requests are properly translated. This is required since hardware may utilize the cached old paging structure entries or the new paging structure entries in memory to translate in-flight requests, until the Context-cache, PASID-cache, IOTLB, and HPT-cache invalidations are completed. For implementations reporting the Enhanced Set Root Table Pointer Support (ESRTPS) field as Clear, software must not modify the Translation Table Mode (TTM) field in the Root-table Address register while remapping hardware is active (TES=1 in Global Status register). For implementations reporting the Enhanced Set Root Table Pointer Support (ESRTPS) field as Set, software may modify the Translation Table Mode (TTM) field in the Root-table Address register while remapping hardware is active (TES=1 in Global Status register).

6.7 Set Interrupt Remapping Table Pointer Operation

Software must always set the interrupt-remapping table pointer before enabling or re-enabling (after disabling) interrupt-remapping hardware.

For implementations reporting the Enhanced Set Interrupt Remap Table Pointer Support (ESIRTP) field as Clear, after a 'Set Interrupt Remap Table Pointer' operation, software must globally invalidate the interrupt entry cache. This is required to ensure hardware uses only the interrupt-remapping entries referenced by the new interrupt remap table pointer, and not stale cached entries.

For implementations reporting the Enhanced Set Interrupt Table Pointer Support (ESIRTP) field as Set, as part of 'Set Interrupt Remap Table Pointer' operation, hardware performs global invalidation on all interrupt remapping translation caches and hence software is not required to perform additional invalidations.

If software updates the interrupt-remapping table pointer while interrupt-remap hardware is active, software must ensure the structures referenced by the new interrupt-remapping table pointer provide identical remapping results as the structures referenced by the previous interrupt-remapping table pointer to ensure any valid in-flight interrupt requests are properly remapped. This is required since hardware may utilize the old structures or the new structures to remap in-flight interrupt requests, until the IEC invalidation is completed.

6.8 Write Buffer Flushing

On remapping hardware page-table walk, earlier implementations of this architecture did not flush or snoop the write buffers at the memory controller that buffers writes to DRAM, and required explicit software flushing of these write buffers on paging structure modifications. These earlier hardware implementations reported this restriction to software by reporting the Required Write Buffer Flushing (RWBF) field in the Capability Register to 1.

For such hardware implementations requiring write buffer flushing (RWBF=1 in the Capability register), software updates to memory-resident remapping structures may be held in Root-Complex internal hardware write-buffers, and not implicitly visible to remapping hardware. For such implementations, software must explicitly make these updates visible to hardware through one of two methods below:

- For updates to remapping hardware structures that require context-cache, PASID-cache, IOTLB or IEC invalidation operations to flush stale entries from the hardware caches, no additional action is required to make the modifications visible to hardware. This is because, hardware performs an implicit write-buffer-flushing as a pre-condition to context-cache, PASID-cache, IOTLB and IEC invalidation operations.
- For updates to remapping hardware structures (such as modifying a currently not-present entry) that do not require context-cache, PASID-cache, IOTLB or IEC invalidations, software must explicitly perform write-buffer-flushing to ensure the updated structures are visible to hardware.

Newer hardware implementations are expected to NOT require explicit software flushing of write buffers and report RWBF=0 in the Capability register.

6.9 Hardware Register Programming Considerations

A register used to submit a command to a remapping unit is owned by hardware while the command is pending in hardware. Software must not update the associated register until hardware indicates the command processing is complete through appropriate status registers.

For each remapping hardware unit, software may submit a command through the Global Command register, Enhanced Command Interface, Context Command register, IOTLB registers or Protected Memory Enable register. After submitting each command, software must wait for remapping hardware to confirm completion of the command (via appropriate status bit) before submitting another command.

For platforms supporting more than one remapping hardware unit, there are no hardware serialization requirements for operations across remapping hardware units.

6.10 Sharing Remapping Structures Across Hardware Units

Software may share[1] (fully or partially) the various remapping structures across multiple remapping hardware units. When the remapping structures are shared across hardware units, software must explicitly perform the invalidation operations on each remapping hardware unit sharing the modified entries. The software requirements described in this chapter must be individually applied for each such invalidation operation.

1. Sharing of scalable-mode root tables and scalable-mode context tables across remapping hardware units are possible only across remapping hardware units that report Scalable Mode Translation Support (SMTS) field as Set in the Extended Capability register.

7 Address Translation Faults

Address translation faults are classified as follows:

- ***Non-recoverable Faults***: Requests that encounter non-recoverable address translation faults are aborted by the remapping hardware, and typically require a reset of the device (such as through a function-level-reset) to recover and re-initialize the device to put it back into service.
- ***Recoverable Faults***: Requests that encounter recoverable address translation faults can be retried by the requesting device after the condition causing the recoverable fault is handled by software. Recoverable translation faults are detected at the Device-TLB on the device and require the device to support Address Translation Services (ATS) capability. Refer to Address Translation Services in PCI Express Base Specification Revision 4.0 or later for details.

7.1 Remapping Hardware Behavior on Faults

7.1.1 Non-Recoverable Address Translation Faults

Non-recoverable address translation faults can be detected by remapping hardware for many different kinds of requests as shown by Table 30. A non-recoverable fault condition is considered "qualified" if software can suppress reporting of the fault by setting one of the Fault Processing Disable (FPD) bits available in one or more of the address translation structures (i.e., the context-entry, scalable-mode context-entry, scalable-mode PASID-directory entry, scalable-mode PASID-table entry). For a request that encounters a "qualified" non-recoverable fault condition, if the remapping hardware encountered any translation structure entry with an FPD field value of 1, the remapping hardware must not report the fault to software. For example, when processing a request that encounters an FPD field with a value of 1 in the scalable-mode context-entry and encounters any "qualified" fault such as SCT.*, SPD.*, SPT.*, SFS.*, SSS.*, or SGN.*, the remapping hardware will not report the fault to software. Memory requests that result in non-recoverable address translation faults are blocked by hardware. The exact method for blocking such requests are implementation-specific. For example:

- Faulting write requests may be handled in much the same way as hardware handles write requests to non-existent memory. For example, the write request is discarded in a manner convenient for implementations (such as by dropping the cycle, completing the write request to memory with all byte enables masked off, re-directing to a catch-all memory location, etc.).
- Faulting read requests may be handled in much the same way as hardware handles read requests to non-existent memory. For example, the request may be redirected to a catch-all memory location, returned as all 0's or 1's in a manner convenient to the implementation, or the request may be completed with an explicit error indication (recommended). For faulting read requests from PCI Express devices, hardware indicates "Unsupported Request" (UR) or "Completer Abort" (CA) in the completion status field of the PCI Express read completion.

7.1.2 Recoverable Address Translation Faults

When remapping hardware detects a recoverable fault on a translation-request from Device-TLB, it is not reported to software as a fault. Instead, remapping hardware sends a successful translation completion with limited or no permission/privileges. When such a translation completion is received by the Device-TLB, a translation fault is detected at the Device-TLB, and handled as recoverable fault if the Device supports recoverable address translation faults. What device accesses can tolerate and recover from Device-TLB detected faults and what device accesses cannot tolerate Device-TLB detected faults is specific to the device. Device-specific software (e.g., driver) is expected to make

sure translations with appropriate permissions and privileges are present before initiating device accesses that cannot tolerate faults. Device operations that can recover from such Device-TLB faults typically involves two steps:

- Report the recoverable fault to host software; This may be done in a device-specific manner (e.g., through the device-specific driver), or if the device supports PCI Express Page Request Services (PRS) Capability, by issuing a page-request message to the remapping hardware. Section 7.4 describe the page-request interface through the remapping hardware.
- After the recoverable fault is serviced by software, the device operation that originally resulted in the recoverable fault may be replayed, in a device-specific manner.

Device-TLB implementations must ensure that a device request, that led to detection of a translation fault in the Device-TLB and reporting of the fault to system software, does not reuse the same faulty translation on retry of the device request after software has informed the device that the reported fault has been handled. However, other device requests may use the same translation in Device-TLB and may succeed or report another fault to system software. One way devices can meet this requirement is by removing the faulty translations from the Device-TLB after receiving confirmation from system software that the fault has been serviced, however there may other device specific methods to achieve this goal. If a recoverable page fault is reported to software in a device-specific manner, rather than using Page Request Services, then software should ensure that stale IOTLB entries in the remapping hardware in root-complex are invalidated.

7.1.3 Fault Conditions and Remapping Hardware Behavior for Various Requests

Table 30. Fault Conditions and Remapping Hardware Behavior for Various Requests

Condition	Condition Code	Fault Reason (If reported)	Qualified	Untranslated Req-without-PASID	Untranslated Req-with-PASID	Translation Req-without-PASID	Translation Req-with-PASID	Translated Req-without-PASID	Translated Req-with-PASID	Page Request that Requires Response	Page Request that does not Require Response
Legacy Root-Table Faults											
A hardware attempt to access a root-entry referenced through the Root-Table Address (RTA) field in the Root-entry Table Address Register resulted in an error.	LRT.1	8h	No	UR	NA	CA	NA	UR	NA	NA	NA
The Present (P) field in root-entry used to process a request is 0.	LRT.2	1h	No	UR	NA	UR	NA	UR	NA	NA	NA
Non-zero reserved field in a root-entry with Present (P) field set.	LRT.3	Ah	No	UR	NA	CA	NA	UR	NA	NA	NA
Legacy Context-Table Faults											
A hardware attempt to access a context-entry referenced through the CTP field in a root-entry resulted in an error.	LCT.1	9h	No	UR	NA	CA	NA	UR	NA	NA	NA
The Present (P) field in context-entry used to process a request is 0.	LCT.2	2h	Yes	UR	NA	UR	NA	UR	NA	NA	NA
Non-zero reserved field in a context-entry with Present (P) field set.	LCT.3	Bh	Yes	UR	NA	CA	NA	UR	NA	NA	NA
Invalid programming of a context-entry used to process a request.	LCT.4.0	3h									
• The Address-Width (AW) field is programmed with a value not supported by hardware.	LCT.4.1	3h	Yes	UR	NA	CA	NA	UR	NA	NA	NA
• The Translation-Type (TT) field is programmed to indicate a translation type not supported by the hardware implementation.	LCT.4.2	3h	Yes	UR	NA	CA	NA	UR	NA	NA	NA
• A hardware attempt to access the second-stage paging entry referenced through the SSPTPTR field of the context-entry resulted in an error.	LCT.4.3	3h	Yes	UR	NA	CA	NA	NA	NA	NA	NA
Translation Type (TT) field in present context-entry, specifies blocking of translation request (without PASID) and translated request.	LCT.5	Dh	Yes	NA	NA	UR	NA	UR	NA	NA	NA
Legacy Second-Stage Table Faults											

Table 30. Fault Conditions and Remapping Hardware Behavior for Various Requests (Contd.)

Condition	Condition Code	Fault Reason (If reported)	Qualified	Untranslated Req-without-PASID	Untranslated Req-with-PASID	Translation Req-without-PASID	Translation Req-with-PASID	Translated Req-without-PASID	Translated Req-with-PASID	Page Request that Requires Response	Page Request that does not Require Response
When legacy mode (RTADDR_REG.TTM=00b) is enabled, a hardware attempt to access a second-stage paging entry (SS-PML4E, SS-PDPE, SS-PDE, or SS-PTE) referenced through the address (ADDR) field in a preceding second-stage paging entry (SS-PML5E, SS-PML4E, SS-PDPE, SS-PDE) resulted in an error.	LSS.1	7h	Yes	UR	NA	CA	NA	NA	NA	NA	NA
When legacy mode (RTADDR_REG.TTM=00b) is enabled, a non-zero reserved field in a second-stage paging entry (SS-PML5E, SS-PML4E, SS-PDPE, SS-PDE, or SS-PTE) with at least one of the Read (R), or Write (W) fields set.	LSS.2	Ch	Yes	UR	NA	CA	NA	NA	NA	NA	NA
Legacy General Faults											
Address overflow in second-stage translation. For example:	LGN.1.0	4h									
• When legacy mode (RTADDR_REG.TTM=00b) is enabled, the address in an Untranslated or Translation request is above (2^X - 1), where X is the minimum of MGAW reported in the capability register and the value in the Address-Width (AW) field of the context-entry used to process a request.	LGN.1.1	4h	Yes	UR	NA	Success with R=W=U=S=0	NA	NA	NA	NA	NA
• When legacy mode (RTADDR_REG.TTM=00b) is enabled, the address in a Translated request is above the Host Address width (HAW) supported by the DMA remapping hardware.	LGN.1.2	4h	Yes	NA	NA	NA	NA	UR	NA	NA	NA
• When legacy mode (RTADDR_REG.TTM=00b) is enabled, the address in an Untranslated request using pass-through translation type (TT=10) is above the Host Address Width (HAW) supported by the DMA remapping hardware.	LGN.1.3	4h	Yes	UR	NA	NA	NA	NA	NA	NA	NA
When legacy mode (RTADDR_REG.TTM=00b) is enabled, a Write or AtomicOp request encountered lack of write permission.	LGN.2	5h	Yes	UR	NA	Success with effective permission	NA	NA	NA	NA	NA
When legacy mode (RTADDR_REG.TTM=00b) is enabled, a Read or AtomicOp request encountered lack of read permission. For implementations reporting the ZLR field as 1 in the capability register, this fault condition is not applicable for zero-length read requests to write-only mapped pages in second-stage translation.	LGN.3	6h	Yes	UR	NA	Success with effective permission	NA	NA	NA	NA	NA
When legacy mode (RTADDR_REG.TTM=00b) is enabled, hardware detected an output address (i.e. address after remapping) in the interrupt address range (FEEx_xxxxh). For Translated requests and requests with pass-through translation type (TT=10), the output address is the same as the address in the request.	LGN.4	Eh	Yes	UR	NA	CA	NA	UR	NA	NA	NA

Table 30. Fault Conditions and Remapping Hardware Behavior for Various Requests (Contd.)

Condition	Condition Code	Fault Reason (If reported)	Qualified	Untranslated Req-without-PASID	Untranslated Req-with-PASID	Translation Req-without-PASID	Translation Req-with-PASID	Translated Req-without-PASID	Translated Req-with-PASID	Page Request that Requires Response	Page Request that does not Require Response
Root-Table Address Register Faults											
Invalid programming of Root Table Address (RTADDR_REG) registers. For example:	RTA.1.0	30h									
• The TTM field is programmed to value of 11b on Hardware Implementations not supporting Abort DMA Mode (ADMS=0 in Extended Capability Register).	RTA.1.1	30h	No	UR	UR	CA	CA	UR	UR	IR	drop
• The TTM field is programmed to value of 10b.	RTA.1.2	30h	No	UR	UR	CA	CA	UR	UR	IR	drop
• The TTM field is programmed to value of 01b for hardware implementation not supporting Scalable Translation Mode (SMTS=0 in Extended Capability Register).	RTA.1.3	30h	No	UR	UR	CA	CA	UR	UR	IR	drop
• The Second Stage I/O Read/Write Enable (RTADDR_REG.SSIRWE) field is set when the TTM field is programmed to legacy mode (TTM=00b).	RTA.1.4	30h	No	UR	NA	CA	NA	UR	NA	NA	NA
Translation Table Mode (TTM) field with value 00b in Root-table Address register (RTADDR_REG) used to process request-with-PASID.	RTA.2	31h	No	NA	UR	NA	UR	NA	UR	NA	NA
For hardware implementations supporting Page Request Service (PRS), Translation Table Mode (TTM) field with value 00b in Root-table Address register (RTADDR_REG) used to process a page request (with or without PASID).	RTA.3	32h	No	NA	NA	NA	NA	NA	NA	IR	drop
For hardware implementations supporting Abort DMA Mode (ADMS=1 in Extended Capability Register), Translation Table Mode (TTM) field with value 11b in Root-table Address register (RTADDR_REG) used to process Untranslated/Translation/Translated request or a page request.	RTA.4	33h	No	UR	UR	UR	UR	UR	UR	IR	drop
Hardware implementations with Major Version 8 and above (VER_REG) detected an untranslated/translation/translated request-with-PASID with Execute-Requested (ER) field set.	RTA.5	34h	No	NA	UR	NA	UR	NA	UR	NA	NA
Scalable-Mode Root-Table Faults											
A hardware attempt to access a scalable-mode root-entry referenced through the Root-Table Address (RTA) field in the Root-entry Table Address Register resulted in an error.	SRT.1	38h	No	UR	UR	CA	CA	UR	UR	IR	drop
The Present (P) field in UP/LP fields in scalable-mode root-entry used to process a request is 0.	SRT.2	39h	No	UR	UR	UR	UR	UR	UR	IR	drop

Table 30. Fault Conditions and Remapping Hardware Behavior for Various Requests (Contd.)

Condition	Condition Code	Fault Reason (If reported)	Qualified	Untranslated Req-without-PASID	Untranslated Req-with-PASID	Translation Req-without-PASID	Translation Req-with-PASID	Translated Req-without-PASID	Translated Req-with-PASID	Page Request that Requires Response	Page Request that does not Require Response
Non-zero reserved field in lower 64-bits of the scalable-mode root-entry with LP field set, or non-zero reserved fields in the upper 64-bits of the scalable-mode root-entry with UP field set.	SRT.3	3Ah	No	UR	UR	CA	CA	UR	UR	IR	drop
Scalable-Mode Context-Table Faults											
A hardware attempt to access a scalable-mode context-entry referenced through UCTP/LCTP field in the scalable-mode root-entry resulted in an error.	SCT.1	40h	No	UR	UR	CA	CA	UR	UR	IR	drop
The Present (P) field in the scalable-mode context-entry used to process a request is 0.	SCT.2	41h	Yes	UR	UR	UR	UR	UR	UR	IR	drop
Non-zero reserved field in a scalable-mode context-entry with Present (P) field set.	SCT.3	42h	Yes	UR	UR	CA	CA	UR	UR	IR	drop
Invalid programming of a scalable-mode context-entry used to process a request.	SCT.4.0	43h									
• For scalable-mode context-entry, the Device-TLB Enable (DTE) field and Page Request Enable (PRE) field are inconsistently programmed. (DTE=0 and PRE=1 is an illegal combination.)	SCT.4.1	43h	Yes	UR	UR	CA	CA	UR	UR	IR	drop
• The value in the RID_PASID field of a scalable-mode context-entry (with P=1) is larger than the maximum PASID-value supported by PDTS field in the scalable-mode context-entry.	SCT.4.2	43h	Yes	UR	UR	CA	CA	UR	UR	IR	drop
• The Enable PASID in Translated Requests (EPTR) field and PASID Enable (PASIDE) field are inconsistently programmed. (EPTR=1 and PASIDE=0 is an illegal combination.)	SCT.4.3	43h	Yes	UR	NA	CA	NA	UR	UR	IR	drop
The Device-TLB Enable (DTE) field in a scalable-mode context-entry used to process the translation request (with or without PASID) or translated request is 0.	SCT.5	44h	Yes	NA	NA	UR	UR	UR	UR	NA	NA
The PASID Enable (PASIDE) field in a present scalable-mode context-entry used to process the untranslated request with PASID, translation request with PASID or page request with PASID is 0.	SCT.6	45h	Yes	NA	UR	NA	UR	NA	NA	IR	drop
The PASID value in the untranslated/translation/translated request with PASID or page request with PASID is larger than the maximum PASID-value supported by the PDTS field in the scalable-mode context-entry used to process the request.	SCT.7	46h	Yes	NA	UR	NA	UR	NA	UR	IR	drop

Table 30. Fault Conditions and Remapping Hardware Behavior for Various Requests (Contd.)

Condition	Condition Code	Fault Reason (If reported)	Qualified	Untranslated Req-without-PASID	Untranslated Req-with-PASID	Translation Req-without-PASID	Translation Req-with-PASID	Translated Req-without-PASID	Translated Req-with-PASID	Page Request that Requires Response	Page Request that does not Require Response
The Page Request Enable (PRE) field in a present scalable-mode context-entry used to process a page request is 0.	SCT.8	47h	Yes	NA	NA	NA	NA	NA	NA	IR	drop
The PASID value in an untranslated request with PASID, a translation request with PASID, a translated request with PASID, or a page request with PASID matches the RID_PASID value in the scalable-mode context entry used to process the request.	SCT.9	48h	Yes	NA	UR	NA	UR	NA	UR	IR	drop
The Enable PASID in Translated Requests (EPTR) field is Clear in the scalable-mode context entry used to process a translated request with PASID.	SCT.10	49h	Yes	NA	NA	NA	NA	NA	UR	NA	NA
Scalable-Mode PASID-Directory Faults											
A hardware attempt to access the scalable-mode PASID-directory entry referenced through the PASIDDIRPTR field in scalable-mode context-entry resulted in an error.	SPD.1	50h	No	UR	UR	CA	CA	CA	CA	IR	drop
The Present (P) field in the scalable-mode PASID-directory entry used to process the untranslated/translation/translated request (with or without PASID) is 0.	SPD.2	51h	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	UR	UR	IR	drop
Non-zero reserved field in a present scalable-mode PASID-directory entry used to process the untranslated/translation/translated request (with or without PASID).	SPD.3	52h	Yes	UR	UR	CA	CA	CA	CA	IR	drop
Scalable-Mode PASID-Table Faults											
A hardware attempt to access a scalable-mode PASID-table entry referenced through the SMPTBLPTR field in a scalable-mode PASID-directory entry resulted in an error.	SPT.1	58h	No	UR	UR	CA	CA	CA	CA	IR	drop
The Present (P) field in the scalable-mode PASID-table entry used to process the untranslated/translation/translated request (with or without PASID) or a page request is 0.	SPT.2	59h	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	UR	UR	IR	drop
Non-zero reserved field in a present scalable-mode PASID-table entry used to process the untranslated/translation/translated request (with or without PASID) or a page request.	SPT.3	5Ah	Yes	UR	UR	CA	CA	CA	CA	IR	drop
Invalid programming of the scalable-mode PASID-table entry used to process a request. For example:	SPT.4.0	5Bh									

Table 30. Fault Conditions and Remapping Hardware Behavior for Various Requests (Contd.)

Condition	Condition Code	Fault Reason (If reported)	Qualified	Untranslated Req-without-PASID	Untranslated Req-with-PASID	Translation Req-without-PASID	Translation Req-with-PASID	Translated Req-without-PASID	Translated Req-with-PASID	Page Request that Requires Response	Page Request that does not Require Response
• The Address-Width (AW) field is programmed with a value not supported by hardware.	SPT.4.1	5Bh	Yes	UR	UR	CA	CA	CA	CA	IR	drop

• The PASID Granular Translation-Type (PGTT) field is programmed to indicate a translation type not supported by the hardware implementation.
  • PGTT= 000b, 101b, 110b, 111b
  • PGTT=001b, when ECAP_REG.FSTS is Clear
  • PGTT=010b, when ECAP_REG.SSTS is Clear
  • PGTT=011b, when ECAP_REG.NEST is Clear
  • PGTT=100b, when ECAP_REG.PT is Clear
	SPT.4.2	5Bh	Yes	UR	UR	CA	CA	CA	CA	IR	drop

• The First Stage Paging Mode (FSPM) field is programmed with a value not supported by hardware.	SPT.4.3	5Bh	Yes	UR	UR	CA	CA	CA	CA	IR	drop
• For nested translation(PGTT=011b), The FSPTPTR field in the present scalable-mode PASID-table entry represents an address above (2^X - 1), where X is the minimum of MGAW reported in the capability register and value in the Address-Width (AW) field of a scalable-mode PASID-table entry used to process a request.	SPT.4.4	5Bh	Yes	UR	UR	CA	CA	CA	CA	IR	drop
Hardware implementations with Major Version 7 and below (VER_REG) attempted to process an untranslated/translation request (with PASID) with Execute-Requested (ER) field set and the Execute Requests Support (ERS) field is 0 in the Extended Capability Register.	SPT.5	5Ch	Yes	NA	UR	NA	Success with R=W=U=S=0	NA	NA	NA	NA
The Supervisor Requests Enable (SRE) field is 0 in the present scalable-mode PASID-table entry used to process an untranslated/translation request (with or without PASID) with Privileged-mode-Requested (PR) field Set. (PR value may come from RID_PRIV field in scalable-mode context entry.)	SPT.6	5Dh	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	NA	NA	NA	NA
The address in a Translated Request exceeded the size of the HPT governed by HPT Size field.	SPT.9	60h	Yes	NA	NA	NA	NA	UR	UR	NA	NA
Scalable-Mode First-Stage Table Faults											

Table 30. Fault Conditions and Remapping Hardware Behavior for Various Requests (Contd.)

Condition	Condition Code	Fault Reason (If reported)	Qualified	Untranslated Req-without-PASID	Untranslated Req-with-PASID	Translation Req-without-PASID	Translation Req-with-PASID	Translated Req-without-PASID	Translated Req-with-PASID	Page Request that Requires Response	Page Request that does not Require Response
A hardware attempt to access a first-stage paging entry (FS-PML4E with 5-level paging, FS-PDPE, FS-PDE, or FS-PTE) referenced through the Address (ADDR) field in a preceding first-stage paging entry (FS-PML5E with 5-level paging, FS-PML4E, FS-PDPE, or FS-PDE) resulted in an error. (For nested translations, second-stage nested translation faults encountered when accessing first-stage paging entries are treated as fault conditions SSS.1-SSS.5, SGN.5, SGN.6 or SGN.7. See description of these fault conditions in Table 30.)	SFS.1	70h	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	NA	NA	NA	NA
The Present (P) field in first-stage paging entry (FS-PML5E, FS-PML4E, FS-PDPE, FS-PDE, or FS-PTE) is 0.	SFS.2	71h	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	NA	NA	NA	NA
Non-zero reserved field in first-stage paging entry (FS-PML5E, FS-PML4E, FS-PDPE, FS-PDE, or FS-PTE) with Present (P) field set.	SFS.3	72h	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	NA	NA	NA	NA
A hardware attempt to access the FS-PML4 (FS-PML5 with 5-level paging) entry referenced through the FSPTPTR field in the scalable-mode PASID-table entry resulted in an error.	SFS.4	73h	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	NA	NA	NA	NA
For nested translation, the ADDR field in a first-stage paging entry (FS-PML5E, FS-PML4E, FS-PDPE, FS-PDE, or FS-PTE) represents an address above (2^X - 1), where X is the minimum of MGAW reported in the capability register and value in the Address-Width (AW) field of a scalable-mode PASID-table entry used to process a request.	SFS.5	74h	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	NA	NA	NA	NA
For nested translation, read of the FS-PML4 (FS-PML5 with 5-level paging) entry referenced through FSPTPTR field in the scalable-mode PASID-table entry was blocked due to lack of read permissions in the nested second-stage translation.	SFS.6	75h	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	NA	NA	NA	NA
For nested translations, read of first-stage paging entry (FS-PML4E with 5-level paging, FS-PDPE, FS-PDE, or FS-PTE) was blocked due to lack of read permissions in the nested second-stage translation.	SFS.7	76h	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	NA	NA	NA	NA
For nested translations, Access/Dirty/Extended-Access flag update to first-stage paging entry (FS-PML5E, FS-PML4E, FS-PDPE, FS-PDE, or FS-PTE) was blocked due to lack of write permissions in the nested second-stage translation.	SFS.8	77h	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	NA	NA	NA	NA

Table 30. Fault Conditions and Remapping Hardware Behavior for Various Requests (Contd.)

Condition	Condition Code	Fault Reason (If reported)	Qualified	Untranslated Req-without-PASID	Untranslated Req-with-PASID	Translation Req-without-PASID	Translation Req-with-PASID	Translated Req-without-PASID	Translated Req-with-PASID	Page Request that Requires Response	Page Request that does not Require Response
When remapping hardware is setup to not snoop processor caches on access to first-stage paging structure (ECAP_REG.SMPWC=0 or PWSNP field in PASID-table entry is 0), hardware encountered a need to update the A/D bit in a first-stage paging structure entry.	SFS.9	90h	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	NA	NA	NA	NA
Remapping hardware failed to update Access/Dirty/Extended-Access flag in first-stage paging structure entry.	SFS.10	91h	Yes	UR	UR	Success with R=W=U=S=0. Also reported as non-recoverable fault	Success with R=W=U=S=0. Also reported as non-recoverable fault	NA	NA	NA	NA
Scalable-Mode Second-Stage Table Faults											
When operating in scalable mode (RTADDR_REG.TTM=01b), a hardware attempt to access a second-stage paging entry (SS-PML4E, SS-PDPE, SS-PDE, or SS-PTE) referenced through the address (ADDR) field in a preceding second-stage paging entry (SS-PML5E, SS-PML4E, SS-PDPE, SS-PDE) resulted in an error.	SSS.1	78h	Yes	UR	UR	CA	CA	NA	NA	NA	NA
When operating in scalable mode (RTADDR_REG.TTM=01b), hardware encountered a second-stage paging entry (SS-PML5E, SS-PML4E, SS-PDPE, SS-PDE, or SS-PTE) with R=W=0 or hardware detected that the logical-AND of the read permission bits and logical-AND of write permission bits from the result of the second-stage page-walk for which both cumulative permissions are not granted.	SSS.2	79h	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	NA	NA	NA	NA
When operating in scalable mode (RTADDR_REG.TTM=01b), a non-zero reserved field in a second-stage paging entry (SS-PML5E, SS-PML4E, SS-PDPE, SS-PDE, or SS-PTE) with at least one of the read or write permission bits set.	SSS.3	7Ah	Yes	UR	UR	CA	CA	NA	NA	NA	NA
When operating in scalable mode (RTADDR_REG.TTM=01b), a hardware attempt to access the second-stage paging entry referenced through the SSPTPTR field in scalable-mode PASID-table entry resulted in an error.	SSS.4	7Bh	Yes	UR	UR	CA	CA	NA	NA	NA	NA

Table 30. Fault Conditions and Remapping Hardware Behavior for Various Requests (Contd.)

Condition	Condition Code	Fault Reason (If reported)	Qualified	Untranslated Req-without-PASID	Untranslated Req-with-PASID	Translation Req-without-PASID	Translation Req-with-PASID	Translated Req-without-PASID	Translated Req-with-PASID	Page Request that Requires Response	Page Request that does not Require Response
When remapping hardware is setup to not snoop processor caches on access to second-stage paging structure (ECAP_REG.SMPWC=0 or PWSNP field in PASID-table entry is 0), hardware encountered a need to update the A/D bit (SSADE field in PASID-table entry is 1 and A/D bit in a second-stage paging structure entry is 0) in a second-stage paging structure entry.	SSS.5	7Ch	Yes	UR	UR	CA	CA	NA	NA	NA	NA
Remapping hardware failed to update Access/Dirty bit in second-stage tables.	SSS.6	7Dh	Yes	UR	UR	CA	CA	NA	NA	NA	NA
Scalable-Mode General Faults											
Input-address in the request subjected to first-stage translation is not canonical (i.e., address bits 63:N are not same value as address bits [N-1], where N is 48 bits with 4-level paging and 57 bits with 5-level paging).	SGN.1	80h	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	NA	NA	NA	NA
When performing first-stage translation for request with user privilege (value of 0 in the Privilege-mode-requested (PR) field), hardware encountered a present first-stage-paging-entry with U/S field value of 0 (supervisor), causing a privilege violation.	SGN.2	81h	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	NA	NA	NA	NA
The input-address in the request is a Host Address and is greater than the Host Address Width (HAW) supported by DMA remapping hardware. For example:	SGN.4.0	83h									
• When operating in scalable mode (RTADDR_REG.TTM=01b), the address in a Translated request is above the Host Address Width (HAW) supported by the DMA remapping hardware.	SGN.4.1	83h	Yes	NA	NA	NA	NA	UR	UR	NA	NA
• When operating in scalable mode (RTADDR_REG.TTM=01b), the address in an Untranslated/Translation request (with or without PASID) using pass-through translation type (PGTT=100) is above the Host Address Width (HAW) supported by the DMA remapping hardware.	SGN.4.2	83h	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	NA	NA	NA	NA
When operating in scalable mode (RTADDR_REG.TTM=01b), the address in a untranslated or translation request (with or without PASID) using second-stage only translation (PGTT=010) is above (2^X - 1), where X is the minimum of MGAW reported in the capability register and the value in the Address-Width (AW) field of the context-entry or scalable-mode PASID-table entry used to process the request.	SGN.5	84h	Yes	UR	UR	Success with R=W=U=S=0	Success with R=W=U=S=0	NA	NA	NA	NA

Table 30. Fault Conditions and Remapping Hardware Behavior for Various Requests (Contd.)

Condition	Condition Code	Fault Reason (If reported)	Qualified	Untranslated Req-without-PASID	Untranslated Req-with-PASID	Translation Req-without-PASID	Translation Req-with-PASID	Translated Req-without-PASID	Translated Req-with-PASID	Page Request that Requires Response	Page Request that does not Require Response
When operating in scalable mode (RTADDR_REG.TTM=01b), a Write or AtomicOp request encountered lack of write permission. Pass-through translations do not encounter this condition. For nested translations, the lack of write permission can be at first-stage translation or at the nested second-stage translation. Refer to Section 3.6.1 for access rights checking with first-stage translation, Section 3.7.1 for access rights checking with second-stage translation, and Section 3.8.1 for access rights checking with nested translation.	SGN.6	85h	Yes	UR	UR	Success with effective permission	Success with effective permission	NA	NA	NA	NA
When operating in scalable mode (RTADDR_REG.TTM=01b), a Read or AtomicOp request encountered lack of read permission. First-stage or pass-through translations do not encounter this condition. Nested-translation can encounter this condition only when translating leaf entry (FS-PTE for 4K, FS-PDE for 2M, FS-PDPE for 1G) through second-stage mappings. Refer to Section 3.6.1 for access rights checking with first-stage translation, Section 3.7.1 for access rights checking with second-stage translation, and Section 3.8.1 for access rights checking with nested translation. For implementations reporting the ZLR field as 1 in the capability register, this fault condition is not applicable for zero-length read requests to write-only mapped pages in second-stage translation and nested translation.	SGN.7	86h	Yes	UR	UR	Success with effective permission	Success with effective permission	NA	NA	NA	NA
When operating in scalable mode (RTADDR_REG.TTM=01b), hardware detected an output address (i.e., address after remapping) in the interrupt address range (FEEx_xxxxh).	SGN.8	87h									
• Untranslated request with an output address that falls within the interrupt address range (FEEx_xxxxh).	SGN.8.1	87h	Yes	UR	UR	NA	NA	NA	NA	NA	NA
• Translation request using a PASID Table entry with the PGTT field programmed to a value other than pass-through (PGTT=100b) and resulting in an output address range overlapping with the interrupt address range (FEEx_xxxxh). Translation requests using a PASID Table entry with the PGTT field programmed to pass-through (PGTT=100b) do not encounter this condition as it is handled by condition S.1, S.2, and S.3 in this table.	SGN.8.2	87h	Yes	NA	NA	CA	CA	NA	NA	NA	NA
• Translated requests with the output address that falls within the interrupt address range (FEEx_xxxxh).	SGN.8.3	87h	Yes	NA	NA	NA	NA	UR	UR	NA	NA
The PASID value in an untranslated/translation/translated request-with-PASID is greater than the value reported by PASID Size Supported (PSS) field in the extended capability register.	SGN.10	89h	Yes	NA	UR	NA	UR	NA	UR	NA	NA

Table 30. Fault Conditions and Remapping Hardware Behavior for Various Requests (Contd.)

Condition	Condition Code	Fault Reason (If reported)	Qualified	Untranslated Req-without-PASID	Untranslated Req-with-PASID	Translation Req-without-PASID	Translation Req-with-PASID	Translated Req-without-PASID	Translated Req-with-PASID	Page Request that Requires Response	Page Request that does not Require Response
Special Conditions											
Hardware implementations detect the following special conditions:											
• Hardware detected an address in a translation request without PASID in the interrupt address range (FEEx_xxxxh) with NW flag Clear. The special handling to interrupt address range is to comprehend potential endpoint device behavior of issuing translation requests to all of its memory transactions including its message signaled interrupt (MSI) posted writes.	S.1	Not a fault	NA	NA	NA	Success with R=S=0, W=U=1	NA	NA	NA	NA	NA
• Hardware detected an address in a translation request without PASID in the interrupt address range (FEEx_xxxxh) with NW flag Set.	S.2	Not a fault	NA	NA	NA	Success with R=W=U =S=0	NA	NA	NA	NA	NA
HPT Faults											
A hardware attempt to access an HPT entry referenced through the Address field in a preceding HPT entry resulted in an error.	HPT.1	A0h	Yes	NA	NA	NA	NA	CA	CA	NA	NA
Hardware detected a non-zero reserved field in an HPT entry.	HPT.2	A1h	Yes	NA	NA	NA	NA	CA	CA	NA	NA
A hardware attempt to access the HPT entry referenced through the HPT Root field in a PASID table entry resulted in an error.	HPT.3	A2h	Yes	NA	NA	NA	NA	CA	CA	NA	NA
For implementations reporting the ZLR field as 1, when processing a translated zero length read request, hardware terminated the HPT walk before a non-zero PP*i* could be found. Example conditions: • HPTL4E with AV=0. • HPTL3E or HPTL2E with AV=0 and PPi=0. • HPTL1E with PP*i*=0.	HPT.4	A3h	Yes	NA	NA	NA	NA	UR	UR	NA	NA
When operating in scalable mode (RTADDR_REG.TTM=01b) and a translated request (with and without PASID) used a scalable-mode context entry with HPT Enable (HPTE) field set. A Write or AtomicOp request encountered lack of write permission in the HPT tables.	HPT.5	A4h	Yes	NA	NA	NA	NA	UR	UR	NA	NA

Table 30. Fault Conditions and Remapping Hardware Behavior for Various Requests (Contd.)

Condition	Condition Code	Fault Reason (If reported)	Qualified	Untranslated Req-without-PASID	Untranslated Req-with-PASID	Translation Req-without-PASID	Translation Req-with-PASID	Translated Req-without-PASID	Translated Req-with-PASID	Page Request that Requires Response	Page Request that does not Require Response
When operating in scalable mode (RTADDR_REG.TTM=01b), and a translated request (with and without PASID) used a scalable-mode context entry with HPT Enable (HPTE) field set. A Read or AtomicOp request encountered lack of read permission in the HPT tables. For implementations reporting the ZLR field as 1 in the capability register, this fault condition is not applicable for zero-length read requests.	HPT.6	A5h	Yes	NA	NA	NA	NA	UR	UR	NA	NA

Note: Effective permission for Translation requests (with or without PASID) in the table above is defined as output of DMA remapping unit after combining information from DMA remapping tables and requirements from Address Translation Service chapter from PCI Express Base specification.

Note: Translation requests that are blocked by valid page tables (e.g., no reserved violation) get a response of 'Unsupported Requested' (UR) which results in disabling of ATS at the function. This is reasonable because for an SRIOV device, each PF/VF can independently enable/disable ATS, and disabling of ATS on a VF does not affect other VF/PF.

Note: Page requests that encounter a fault prior to successfully updating the Page Request Queue get a response of 'Invalid Request' (IR). Such page requests report a non-recoverable fault and don't get a 'Response Failure' (RF) because a response of RF will result in a device disabling the Page Request Interface (PRI) for the entire device. For an SR-IOV device, PRI cannot be enabled independently for PF/VF and an RF response on a page request from a VF will result in disabling PRI for all VF/PF. For details on IR and RF please refer to the PCI Express Base specification.

Note: Page Requests that do not require a response and encounter a fault condition will result in a 'drop' by remapping hardware and the Page Request Queue will remain unaffected. Such page requests will report the event as a non-recoverable fault.

7.2 Non-Recoverable Fault Reporting

Processing of non-recoverable address translation faults (and interrupt translation faults) involves logging the fault information and reporting to software through a fault event (interrupt). The remapping architecture defines Primary Fault Logging as the default fault logging method that must be supported by all implementations of this architecture.

7.2.1 Primary Fault Logging

The primary method for logging non-recoverable faults is through Fault Recording Registers. The number of Fault Recording Registers supported is reported through the Capability Register (see Section 11.4.2). Section 11.4.7.6 describes the Fault Recording Registers.

Hardware maintains an internal index to reference the Fault Recording Register in which the next non-recoverable fault can be recorded. The index is reset to zero when both address and interrupt translations are disabled (i.e., TES and IES fields Clear in Global Status Register), and increments whenever a fault is recorded in a Fault Recording Register. The index wraps around from N-1 to 0, where N is the number of fault recording registers supported by the remapping hardware unit.

Hardware maintains the Primary Pending Fault (PPF) field in the Fault Status Register as the logical "OR" of the Fault (F) fields across all the Fault Recording Registers. The PPF field is re-computed by hardware whenever hardware or software updates the F field in any of the Fault Recording Registers.

When primary fault recording is active, hardware functions as follows upon detecting a non-recoverable address translation or interrupt translation fault:

- Hardware checks the current value of the Primary Fault Overflow (PFO) field in the Fault Status Register. If it is already Set, the new fault is not recorded.
- If hardware supports compression[1] of multiple faults from the same requester, it compares the source-id (SID) field of each Fault Recording Register with Fault (F) field Set, to the source-id of the currently faulted request. If the check yields a match, the fault information is not recorded.

1. Hardware implementations supporting only a limited number of fault recording registers are recommended to collapse multiple pending faults from the same requester.

- If the above check does not yield a match (or if hardware does not support compression of faults), hardware checks the Fault (F) field of the Fault Recording Register referenced by the internal index. If that field is already Set, hardware sets the Primary Fault Overflow (PFO) field in the Fault Status Register, and the fault information is not recorded.
- If the above check indicates there is no overflow condition, hardware records the current fault information in the Fault Recording Register referenced by the internal index. Depending on the current value of the PPF field in the Fault Status Register, hardware performs one of the following steps:
  - If the PPF field is currently Set (implying there are one or more pending faults), hardware sets the F field of the current Fault Recording Register and increments the internal index.
  - Else, hardware records the internal index in the Fault Register Index (FRI) field of the Fault Status Register and sets the F field of the current Fault Recording Register (causing the PPF field also to be Set). Hardware increments the internal index, and an interrupt may be generated based on the hardware interrupt generation logic described in Section 7.3.

Software is expected to process the non-recoverable faults reported through the Fault Recording Registers in a circular FIFO fashion starting from the Fault Recording Register referenced by the Fault Recording Index (FRI) field, until it finds a Fault Recording Register with no faults (F field Clear).

To recover from a primary fault overflow condition, software must first process the pending faults in each of the Fault Recording Registers, Clear the Fault (F) field in all those registers, and Clear the overflow status by writing a 1 to the Primary Fault Overflow (PFO) field. Once the PFO field is cleared by software, hardware continues to record new faults starting from the Fault Recording Register referenced by the current internal index.

7.3 Non-Recoverable Fault Event

Non-recoverable faults are reported to software using a message-signaled interrupt controlled through the Fault Event Control Register. The non-recoverable fault event information (such as interrupt vector, delivery mode, address, etc.) is programmed through the Fault Event Data and Fault Event Address Registers.

A Fault Event may be generated under the following conditions:

- When primary fault logging is active, recording a non-recoverable fault to a Fault Recording Register causing the Primary Pending Fault (PPF) field in Fault Status Register to be Set.
- When queued invalidation interface is active, an invalidation queue error causing the Invalidation Queue Error (IQE) field in the Fault Status Register to be Set.
- Invalid Device-TLB invalidation completion response received causing the Invalidation Completion Error (ICE) field in the Fault Status Register to be Set.
- Device-TLB invalidation completion time-out detected causing the Invalidation Time-out Error (ITE) field in the Fault Status Register to be Set.

For these conditions, the Fault Event interrupt generation hardware logic functions as follows:

- Hardware checks if there are any previously reported interrupt conditions that are yet to be serviced by software. Hardware performs this check by evaluating if any of the PPF[1], PFO, IQE, ICE and ITE fields in the Fault Status Register is Set. If hardware detects any interrupt condition yet to be serviced by software, the Fault Event interrupt is not generated.
- If the above check indicates no interrupt condition yet to be serviced by software, the Interrupt Pending (IP) field in the Fault Event Control Register is Set. The Interrupt Mask (IM) field is then checked and one of the following conditions is applied:
  - If IM field is Clear, the fault event is generated along with clearing the IP field.

1. The PPF field is computed by hardware as the logical OR of Fault (F) fields across all the Fault Recording Registers of a hardware unit.

— If IM field is Set, the interrupt is not generated.

The following logic applies for interrupts held pending by hardware in the IP field:

- If IP field was Set when software clears the IM field, the fault event interrupt is generated along with clearing the IP field.
- If IP field was Set when software services all the pending interrupt conditions (indicated by all status fields in the Fault Status Register being Clear), the IP field is cleared.

Read completions due to software reading the Fault Status Register (FSTS_REG) or Fault Event Control Register (FECTL_REG) must push (commit) any in-flight Fault Event interrupt messages generated by the respective hardware unit.

The fault event interrupts are never subject to interrupt remapping.

7.4 Recoverable Fault Reporting

Recoverable faults are detected at the Device-TLB on the endpoint device. Devices supporting Page Request Services (PRS) Capability report the recoverable faults as Page Request messages to software through the remapping hardware. Page requests are supported only when DMA remapping hardware is in scalable mode. A page request received by DMA remapping hardware in legacy mode results in a non-recoverable fault (see condition RTA.3). Software informs the servicing of the page requests by sending Page Group Response messages to the device through the remapping hardware. Refer to Address Translation Services (ATS) in PCI Express Base Specification Revision 4.0 or later for details on the Page Request and Page Group Response messages.

The following sections describe the remapping hardware processing of page requests from endpoint devices and page group response from software. Remapping hardware indicates support for page requests through the Extended Capability Register (see Section 11.4.3).

7.4.1 Handling of Page Requests

When PRS Capability is enabled at an endpoint device, recoverable faults detected at its Device-TLB cause the device to issue page-request messages to the remapping hardware.

Remapping hardware supports a Page Request Queue, as a circular buffer in system memory to record Page Request messages received. The following registers are defined to configure and manage the Page Request Queue (PRQ):

- *Page Request Queue Address Register*: Software programs this register to configure the base physical address and size of the contiguous memory region in system memory hosting the Page Request Queue.
- *Page Request Queue Head Register*: This register points to the Page Request Descriptor in the Page Request Queue that software will process next. Software increments this register after processing one or more Page Request Descriptors in the PRQ.
- *Page Request Queue Tail Register*: This register points to the Page Request Descriptor in the Page Request Queue to be written next by hardware. This register is incremented by hardware after it gets confirmation that write of the Page Request Queue Descriptor to PRQ is visible to software.

Hardware interprets the PRQ as empty when the Head and Tail Registers are equal. Hardware interprets the PRQ as full when the Tail Register is one behind the Head Register (i.e., when all entries but one in the queue are used). This way, hardware will write at most N-1 page requests in a N entry Page Request Queue.

To enable page requests from an endpoint device, software must:

- Initialize the Page Request Queue Head and Tail Registers (see Section 11.4.11.1 and Section 11.4.11.2) to zero.

- Set up the PRQ address and size through the Page Request Queue Address Register (see Section 11.4.11.3).
- Configure the scalable-mode context-entry used to process requests from the device, such that the Present (P), Device-TLB Enable (DTE), and Page Request Enable (PRE) fields are Set.
- Configure and enable page requests at the device through the PRS Capability Registers. (Refer to Address Translation Services (ATS) in PCI Express Base Specification Revision 4.0 or later for PRS Capability Register details).

A page request may encounter one of the three error conditions:

- Remapping Table Error: A remapping table error that prevents a page request from successfully processing the scalable-mode PASID-table entry. This includes conditions such as the Page Request Enable (PRE) field in the scalable-mode context-entry used to process the request is 0. For a complete list of conditions that affect page requests, see Table 30. For each of the conditions affecting page requests, hardware will report a non-recoverable fault with an associated fault code (see Table 30) and if needed, provide a response as described in Table 30.
- Page Request Queue Overflow (PRQ Overflow): The Page Request Overflow (PRO) field in the Page Request Status Register is 1. No action is taken by hardware to report a fault or generate an event.
- Page Request Queue Full (PRQ full): The Page Request Queue is full (i.e., the current value of the Tail Register is one behind the value of the Head Register), causing hardware to set the Page Request Overflow (PRO) field in the Page Request Status Register (see Section 11.4.11.4). Setting the PRO field can cause a fault event to be generated depending on the programming of the Page Request Event Registers (see Section 7.5).

When a page request encounters one of the error conditions described above, it is not written into Page Request Queue. Additionally, hardware will generate a page group response for page requests that have the Last Page In Group (LPIG) field set as described by Table 31 below.

Table 31. Page Request Error Conditions

LPIG	Remapping Table Fault	PRQ Overflow	PRQ Full
0	Hardware does not generate any Page Group Response for the dropped Page Request.		
1	Hardware generates Page Group Response with code of Invalid Request	Hardware generates Page Group Response with code of Success	Hardware generates Page Group Response with code of Success

If a page request does not encounter any of the error conditions described above, the remapping hardware:

- Performs an implicit invalidation (see Section 6.5.3.1) to invalidate any translations cached in the IOTLB and paging structure caches that control the address specified in the page request.
- Writes a Page Request Descriptor to the Page Request Queue entry at the offset specified by the Tail Register, and increments the value in the Tail Register. Depending on the type of the Page Request Descriptor written to the Page Request Queue and programming of the Page Request Event Registers, a recoverable fault event may be generated (see Section 7.5).

The following sections describe the Page Request Descriptor types written by hardware to the Page Request Queue. All descriptors are 256-bit in size. The Type field (bits 7:0) of each Page Request Descriptor identifies the descriptor type.

7.4.1.1 Page Request Descriptor

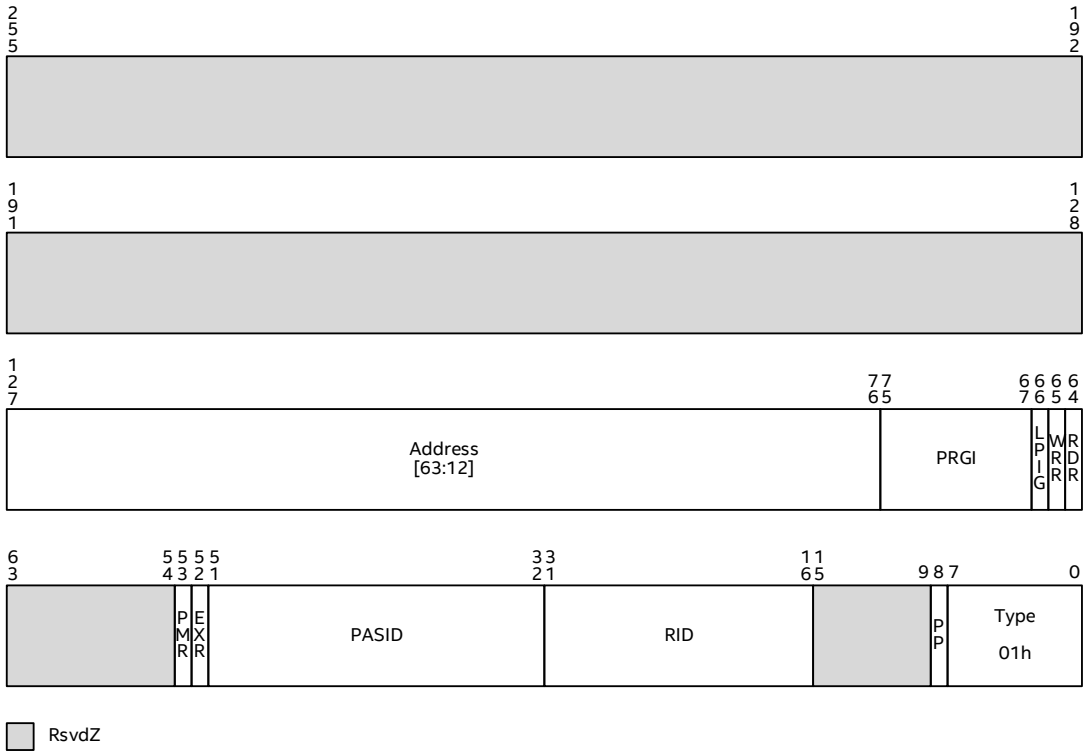


Figure 7-1. Page Request Descriptor

A Page Request Descriptor (*page_req_dsc*) is used to report page request messages received by the remapping hardware.

Page Request Messages: Page request messages[1] are sent by endpoint devices to report one or more page requests that are part of a page group (i.e., with same value in Page Request Group Index field). A page group is composed of one or more individual page request. Page requests with a PASID Present field value of 1 are considered to be page-requests-with-PASID. Page requests with a PASID Present field value of 0 are considered to be page-requests-without-PASID.

The Page Request Descriptor (*page_req_dsc*) includes the following fields:

- *Type*: A value of 01h in this field indicates Page Request Descriptor.
- *Requester-ID (RID):* The Requester-ID field identifies the endpoint device function targeted by the Page Request Group Response. The upper 8 bits of the Requester-ID field specify the bus number and the lower 8 bits specify the device number and function number. Software copies the bus number, device number, and function number fields from the respective Page Request Descriptor to form the Requester-ID field in the Page Group Response Descriptor. Refer to Section 3.4.1 for the format of this field.

1. Refer to the PCI Express Address Translation Services (ATS) specification for details on page request and response messages.

- *PASID Present (PP):* If the PASID Present field is 1, the page request is due to a recoverable fault by a request-with-PASID. If the PASID Present field is 0, the page request is due to a recoverable fault by a request-without-PASID.
- *PASID:* If the PASID Present field is 1, this field provides the PASID value of the request-with-PASID that encountered the recoverable fault that resulted in this page request. If the PASID Present field is 0, this field is undefined.
- *Address (ADDR)*: This field indicates the page address received in the page request.
- *Page Request Group Index (PRGI):* The 9-bit Page Request Group Index field identifies the page group to which this request is part of. Software is expected to return the Page Request Group Index in the respective page group response. This field is undefined if both the Read Requested and Write Requested fields are 0.
- *Last Page in Group (LPIG):* If the Last Page in Group field is 1, this is the last request in the page group identified by the value in the Page Request Group Index field.
- *Read Requested (RDR):* If the Read Requested field is 1, the request that encountered the recoverable fault (that resulted in this page request), requires read access to the page.
- *Write Requested (WRR):* If the Write Requested field is 1, the request that encountered the recoverable fault (that resulted in this page request), requires write access to the page.
- *Execute Requested (EXR):* If the PASID Present, Read Requested and Execute Requested fields are all 1, the request-with-PASID that encountered the recoverable fault that resulted in this page request requires execute access to the page.
- *Privilege Mode Requested (PMR):* The Privilege Mode Requested field indicates the privilege of the request-with-PASID that encountered the recoverable fault (that resulted in this page request). A value of 1 for this field indicates supervisor privilege, and a value of 0 indicates user privilege.

See Section 7.6 for how software is expected to service page requests and respond with page group responses which are described in Section 7.6.1.

7.5 Recoverable Fault Event

Remapping hardware supports notifying pending recoverable faults to software through a Page Request Event interrupt.

- When a Page Request Descriptor (*page_req_dsc*) is written to the Page Request Queue, hardware Sets the Pending Page Request (PPR) field in the Page Request Status Register (see Section 11.4.11.4).
- When attempting to write a page request into the Page Request Queue, hardware encounters PRQ full condition, causing hardware to Set the Page Request Overflow (PRO) field in the Page Request Status Register

The Page Request Event generation hardware logic functions as follows:

- Hardware evaluates PPR and PRO fields in the Page Request Status Register and if any of them is set, Page Request Event is not generated.
- If above evaluation indicates that none of the bits are set, the Interrupt Pending (IP) field in the Page Request Event Control Register (see Section 11.4.11.5) is Set. The Interrupt Mask (IM) field in this register is then checked and one of the following conditions is applied:
  - — If IM field is Clear, the fault event is generated along with clearing the IP field.
  - — If IM field is Set, the interrupt is not generated.

The following logic applies for interrupts held pending by hardware in the IP field in the Page Request Event Control Register:

- If IP field was 1 when software clears the IM field, the Page Request Event interrupt is generated along with clearing the IP field.

- If IP field was 1 when software services all the pending interrupt conditions (indicated by PPR and PRO fields being Clear) in the Page Request Status Register, the IP field is cleared

A page request from an endpoint is considered 'received' by remapping hardware when it arrives at the remapping hardware ingress. A received page request is considered 'accepted' to the page request queue by remapping hardware when the corresponding page request descriptor write (*page_req_dsc)* is issued. An 'accepted' page request is considered 'delivered' by remapping hardware when the respective Page Request Descriptor write (*page_req_dsc*) to Page Request Queue becomes visible to software followed by increment of the Page Request Queue Tail register.

For producer consumer ordering of page request processing, the following ordering requirements must be met by remapping hardware:

- A Page Request Event interrupt must ensure all 'accepted' page requests (including the accepted page request that led to generation of this interrupt) are 'delivered' (become software visible), before the Page Request Event interrupt is delivered to software.
- Read completions due to software reading Page Request Queue Tail Register (PQT_REG) must ensure all 'accepted' page requests are 'delivered'.
- Read completions due to software reading Page Request Status Register (PRS_REG), Page Request Queue Tail Register (PQT_REG) or Page Request Event Control Register (PECTL_REG) must push (commit) any in-flight Page Request Event interrupt generated by the respective remapping hardware unit.

The Page Request Event interrupts are never subject to interrupt remapping.

7.6 Servicing Recoverable Faults

Software processes Page Request Descriptors written to the Page Request Queue by remapping hardware. Processing the descriptor involves resolving the page-fault condition, creating the translation with appropriate permission and privilege (if the page requested is legitimate), and issuing a response back to the device through the remapping hardware when required.

For a page request indicating last request in group (LPIG = 1), software must respond with a page group response after servicing all page requests that are part of that page group. For a page request not indicating last request in group (LPIG = 0), software must not send any page group response. The response is sent by software to the remapping hardware by submitting Page Group Response Descriptor through the Invalidation Queue (IQ). The remapping hardware processes each Page Group Response Descriptor by formatting and sending an appropriate Page Request Group Response message to the endpoint device. Refer to Section 6.5.2 for details on Invalidation Queue operation.

While servicing page request, software may determine that the request is spurious. i.e., the page reported in the page request already has a translation with the requested permissions and privilege in the page tables. Spurious page requests can result if software upgraded a paging entry (e.g., not present to present, read-only to read-write, etc.), and the faulting request used the translation before the upgrade that was cached in the IOTLB or Device-TLB. Irrespective of how a page request was serviced by software (i.e., successfully processed by creating the translation, identified as a spurious page request that did not require any update to translation, identified as invalid request due to invalid page/permission/privilege requested), software must send a page group response with appropriate Response Code if a response is required for the page request.

The following sections describe the Page Group Response Descriptor written by software to the Invalidation Queue. The Type field (bits 11:9 and bits 3:0) of each page group response descriptor identifies the descriptor type (similar to other invalidation descriptors submitted through the Invalidation Queue). This descriptor is a 256-bit descriptor and will result in an invalid descriptor error if submitted in an IQ that is setup to provide hardware with 128-bit descriptors (IQA_REG.DW=0).

7.6.1 Page Group Response Descriptor

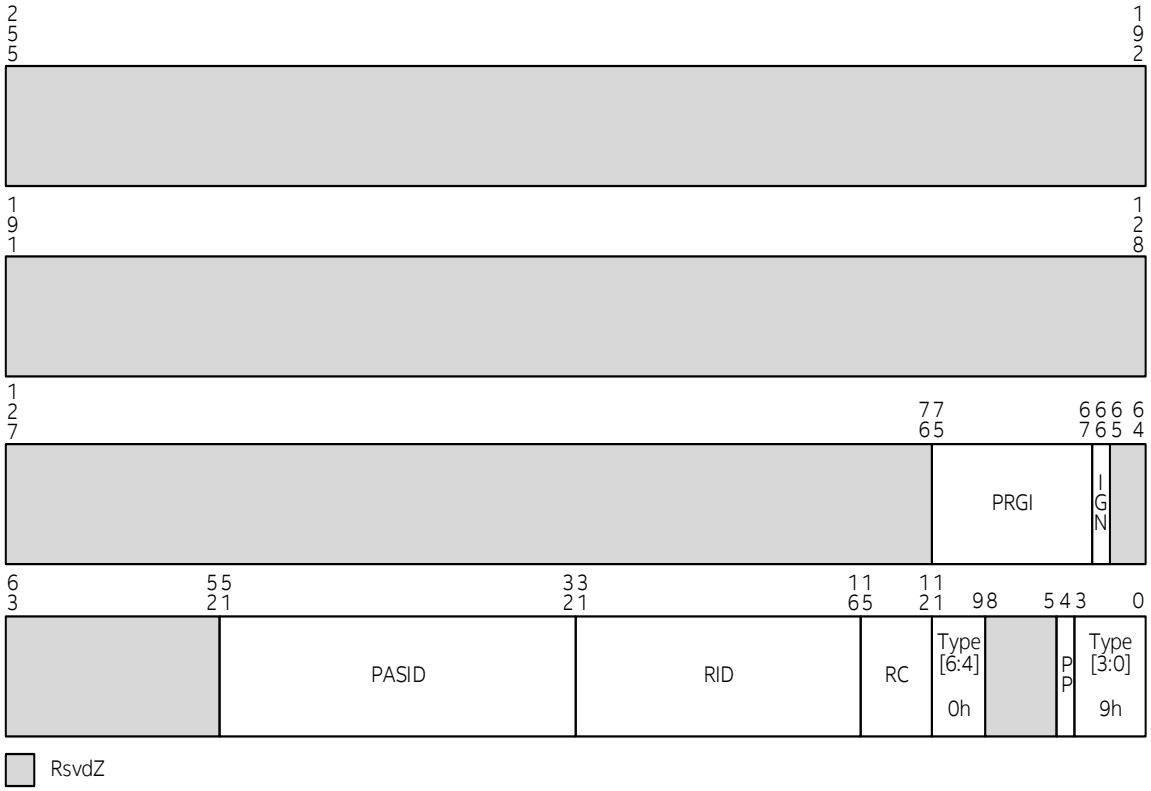


Figure 7-2. Page Group Response Descriptor

A Page Group Response Descriptor is issued by software in response to a page request that indicated that it was the last request in a group. The page group response for last request in a group must be issued after servicing all page requests with the same Page Request Group Index value.

Implementations reporting PRS Support as Clear in Extended Capability (ECAP_REG.PRS) Register treat Page Group Response Descriptors as an invalid descriptor. Refer to Section 11.4.3 for more details.

The Page Group Response Descriptor (*page_grp_resp_dsc*) includes the following fields:

- *Type*: The concatenation of bits[11:9] and bits[3:0] form this 7-bit field. A value of 9h in this field indicates Page Group Response Descriptor.
- *Requester-ID (RID):* The Requester-ID field identifies the endpoint device function targeted by the Page Group Response message. The upper 8 bits of the Requester-ID field specify the bus number and the lower 8 bits specify the device number and function number. Software copies the bus number, device number, and function number fields from the respective Page Request Descriptor to form the Requester-ID field in the Page Group Response Descriptor. Refer to Section 3.4.1 for the format of this field.
- *PASID Present (PP):* If the PASID Present field is 1, the page group response carries a PASID. The value in this field must match the value in the PASID Present field of the respective Page Request Descriptor.

- *PASID:* If the PASID Present field is 1, this field provides the PASID value for the page group response. The value in this field must match the value in the PASID field of the respective Page Request Descriptor. If the PASID Present field is 0, this field is ignored by hardware.
- IGN - Ignored: Hardware must ignore programming of this field to ensure backward compatibility with older software.
- *Page Request Group Index (PRGI):* The Page Request Group Index identifies the page group of this page group response. The value in this field must match the value in the Page Request Group Index field of the respective Page Request Descriptor.
- *Response Code (RC):* The Response Code indicates the page group response status. The field follows the Response Code (see Table 32) in the Page Group Response message as specified in Address Translation Services (ATS) in the PCI Express Base Specification, Revision 4.0 or later. Refer to the PCI Express specification for endpoint device behavior with these Response Codes. If all page requests that are part of a Page Group were serviced successfully, a Response Status code of 'Success' is returned.

Table 32. Response Codes

Value	Status	Description
0h	Success	All Page Requests in the Page Request Group were successfully serviced.
1h	Invalid Request	One or more Page Requests within the Page Request Group was not successfully serviced.
2h - Eh	Reserved	Not used.
Fh	Response Failure	Servicing of one or more Page Requests within the Page Request Group encountered a non-recoverable error.

7.7 Page Request Ordering and Draining

This section describes the expected endpoint device behavior and remapping hardware behavior on ordering and draining of page-requests.

- A recoverable fault encountered by an endpoint device is considered 'dispatched' when the corresponding Page Request message is posted to its egress to the interconnect. On a Device-TLB invalidation, the endpoint device must ensure the respective Device-TLB invalidation completion message is posted to its egress to the interconnect, ordered behind 'dispatched' page requests.
- Page requests and Device-TLB invalidation completion messages follow strict posted ordering on the interconnect fabric and arrive at the ingress of remapping hardware in strict posted ordering.
- A Page request is considered 'received' by remapping hardware when it arrives at the remapping hardware ingress. A 'received' page request is considered 'accepted' to the Page Request Queue by remapping hardware when the corresponding Page Request Descriptor write (*page_req_dsc)* is issued. An 'accepted' page request is considered 'delivered' by remapping hardware when the respective Page Request Descriptor write (*page_req_dsc*) to Page Request Queue becomes visible to software followed by increment of the Page Request Queue Tail register.
- If remapping hardware processing of a Device-TLB invalidation completion message results in a pending Invalidation Wait Descriptor (*inv_wait_dsc*) to complete, and if the Page-request Drain (PD=1) flag is Set in the *inv_wait_dsc*, the respective invalidation wait completion status write (if SW=1) and invalidation wait completion interrupt (if IF=1) must be ordered (made visible to software) behind page-request descriptor (*page_req_dsc*) writes for all page requests 'received' ahead of the Device-TLB invalidation completion message.

With above ordering, to drain in-flight page requests from an endpoint device, software can issue a *dev_tlb_inv_dsc* (or *p_dev_tlb_inv_dsc*) of any invalidation granularity targeting the device, followed by an *inv_wait_dsc* with PD flag Set and SW/IF flag Set, and wait for the *inv_wait_dsc* to complete.

7.8 Page Group Response Ordering and Draining

This section describes the remapping hardware behavior and expected endpoint device behavior on ordering and draining of page-request responses.

- Remapping hardware must ensure that page group responses (*page_grp_resp_dsc)* and Device-TLB invalidation requests (*dev_tlb_inv_dsc* or *p_dev_tlb_inv_dsc*) submitted by software through the Invalidation Queue are processed in order and respective messages are posted in order to the interconnect.
- Page Group Response messages and Device-TLB invalidation requests follow strict posted ordering on the interconnect fabric.
- An endpoint device is expected to 'accept' valid page group responses and Device-TLB invalidation requests in the order they are received. This implies that, before a valid Device-TLB invalidation request is acted on by the endpoint device (leading to posting an invalidation response message), the endpoint device has 'accepted' all valid page group responses received ahead of the Device-TLB invalidation request.

With above ordering, to drain in-flight page group responses issued by software to an endpoint device, software can issue a *dev_tlb_inv_dsc* (or *p_dev_tlb_inv_dsc*) of any invalidation granularity targeting the device, followed by an *inv_wait_dsc* with SW/IF flag Set, and wait for the *inv_wait_dsc* to complete.

7.9 Pending Page Request Handling on Terminal Conditions

This section describes the expected endpoint device behavior for handling pending page requests on terminal conditions. A page request is considered pending at an endpoint device, if the devices has issued the page request and have not yet received the respective page group response.

Terminal conditions are defined as software initiated conditions that can result in an endpoint device resetting its internal state used to match page group responses to pending page requests. Examples of terminal conditions encountered by a device while it has pending page requests may include:

- Device specific software stack/driver detecting a unresponsive device and performing any form of partial device reset (E.g., GPU Engine Reset) leading to termination of one or more work-items (that may have pending page requests).
- Device specific software stack detecting a unresponsive device and performing a full device reset (E.g., GPU Timeout Detection and Recovery Reset) leading to termination of all work-items (some or all with pending page requests) active on that device.
- System software performing Function Level Reset (FLR) of a Physical Function or SR-IOV Virtual Function (or reset of a Intel® Scalable IOV Assignable Device Interface (ADI)), leading to termination of all work-items (some or all with pending page requests) submitted through respective function/interface.

On above terminal conditions[1], the endpoint device is expected to handle pending page requests as follows:

- Ensure that recoverable faults encountered by the device before the terminal event are 'dispatched' (i.e., corresponding page requests are posted to device egress to the interconnect), so that any subsequent Device-TLB invalidation request completions from the device are ordered behind such page requests (i.e., follow same ordering described in Section 7.7 as with normal operation if there was no terminal condition).

1. For devices that may be subject to device specific software stack/driver initiated terminal conditions while in operation, such devices are expected to normally receive, process and respond to Device-TLB invalidation requests during such terminal conditions. This is required as these device specific terminal conditions may be initiated transparent to system software operation that can be issuing Device-TLB invalidations as part of TLB shootdown operations.

- Cancel tracking of pending page requests affected by the terminal condition and replenish page request credits consumed by such page requests.
- Continue normal behavior of discarding any page group responses (without adverse side effects) received that has no matching pending page request.

To handle in-flight page requests affected by terminal conditions, software initiating the terminal condition must follow below steps:

- After completing the terminal condition and before putting the device function back in service, request system software to drain (and discard) in-flight page requests/responses from the endpoint device (as described in Section 7.10), and only after completion of such page request/response draining resume normal operation of the device. For terminal conditions (such as partial device reset) where only a subset of PASIDs active on the device are affected, the drain and discard request may specify the affected PASID(s), in which case, only page requests from the device with specified PASID(s) are discarded after draining (and page requests from other PASIDs are handled normally with page group responses).

The following section describes the steps system software may follow to drain all in-flight and pending page requests and page group responses from/to an endpoint device.

7.10 Software Steps to Drain Page Requests & Responses

System Software may follow below steps to drain in-flight page requests and page group responses between remapping hardware queues (Page Request Queue for page requests and Invalidation Queue for page group responses) and an endpoint device.

a. Submit Invalidation Wait Descriptor (*inv_wait_dsc*) with Fence flag (FN=1) Set to Invalidation Queue. This ensures that all requests submitted to the Invalidation Queue ahead of this wait descriptor are processed and completed by remapping hardware before processing requests after the Invalidation Wait Descriptor. It is not required to specify SW flag (or IF flag) in this descriptor or for software to wait on its completion, as its function is to only act as a barrier.

b. Submit an IOTLB invalidate descriptor (*iotlb_inv_dsc* or *p_iotlb_inv_dsc*) followed by Device-TLB invalidation descriptor (*dev_tlb_inv_dsc* or *p_dev_tlb_inv_dsc*) targeting the endpoint device. These invalidation requests can be of any granularity. Per the ordering requirements described in Section 7.8, older page group responses issued by software to the endpoint device before step (a) are guaranteed to be received by the endpoint before the endpoint receives this Device-TLB invalidation request.

c. Submit Invalidation Wait Descriptor (*inv_wait_dsc*) with Page-request Drain (PD=1) flag Set, along with Invalidation Wait Completion status write flag (SW=1), and wait on Invalidation Wait Descriptor completion. Per the ordering requirements described in Section 7.7, the Device-TLB invalidation completion from the device is guaranteed to be ordered behind already issued page requests from the device. Also, per the ordering requirements in Section 7.7, the remapping hardware ensures that the Invalidation Wait Descriptor status write (that signals completion of invalidation descriptors submitted in step (b)) is ordered (with respect to software visibility) behind the Page Request Descriptor (*page_req_dsc*) writes for page requests received before the Device-TLB invalidation completion.

d. If there are no Page Request Queue overflow condition encountered by remapping hardware during above steps, software can be guaranteed that all page requests and page group responses are drained between the remapping hardware and the target endpoint device. However, if a page-request queue full condition was detected by remapping hardware when processing a page request with Last Page In Group (LPIG) field Set during steps (a) through (c) above, the remapping hardware generates a successful auto page group response (see Section 7.4.1 for remapping hardware auto page group response behavior). To drain such potential auto page group responses generated by remapping hardware, software must repeat steps (b) and (c).

7.11 Revoking PASIDs with Pending Page Faults

At any time of operation, system software resource management actions (e.g., on host application process termination) can result in system software requesting the endpoint device specific driver to revoke the PASID that it has previously allocated and is actively being used by the endpoint device. To service such system software request, it is the responsibility of the endpoint device and the driver to revoke use of this PASID by the device and ensure all outstanding page-requests for this PASID are serviced by system software and page-request responses received, before returning success to system software for the PASID revocation request.

After de-allocating a PASID, system software may follow the same steps for in-flight page request/response draining described in Section 7.10 to ensure any in-flight page requests/responses for the de-allocated PASID are drained before re-allocating that PASID to a new client.

8 BIOS Considerations

The system BIOS is responsible for detecting the remapping hardware functions in the platform and for locating the memory-mapped remapping hardware registers in the host system address space. The BIOS reports the remapping hardware units in a platform to system software through the DMA Remapping Reporting (DMAR) ACPI table described below[1].

8.1 DMA Remapping Reporting Structure

Field	Byte Length	Byte Offset	Description
Signature	4	0	"DMAR". Signature for the DMA Remapping Description table.
Length	4	4	Length, in bytes, of the description table including the length of the associated remapping structures.
Revision	1	8	1
Checksum	1	9	Entire table must sum to zero.
OEMID	6	10	OEM ID
OEM Table ID	8	16	For DMAR description table, the Table ID is the manufacturer model ID.
OEM Revision	4	24	OEM Revision of DMAR Table for OEM Table ID.
Creator ID	4	28	Vendor ID of utility that created the table.
Creator Revision	4	32	Revision of utility that created the table.
Host Address Width	1	36	This field indicates the maximum DMA physical addressability supported by this platform. The system address map reported by the BIOS indicates what portions of this addresses are populated. The Host Address Width (HAW) of the platform is computed as (N+1), where N is the value reported in this field. For example, for a platform supporting 40 bits of physical addressability, the value of 100111b is reported in this field.

1. All Reserved fields in DMAR remapping structures must be initialized to 0.

Field	Byte Length	Byte Offset	Description
Flags	1	37	• Bit 0: INTR_REMAP - If Clear, the platform does not support interrupt remapping. If Set, the platform supports interrupt remapping. INTR_REMAP must be Set for implementations reporting Interrupt Remapping Required (IRREQ) as Set in the Extended Capability Register. • Bit 1: X2APIC_OPT_OUT - For firmware compatibility reasons, platform firmware may Set this field to request system software to opt out of enabling Extended xAPIC (X2APIC) mode. This field is valid only when the INTR_REMAP field (bit 0) is Set. Since firmware is permitted to hand off platform to system software in legacy xAPIC mode, system software is required to check this field as Clear as part of detecting X2APIC mode support in the platform. X2APIC_OPT_OUT must be Clear for implementations reporting Extended Interrupt Mode Enable Required (EIMER) as Set in the Extended Capability Register • Bit 2: DMA_CTRL_PLATFORM_OPT_IN_FLAG: Platform firmware is recommended to Set this field to report any platform initiated DMA is restricted to only reserved memory regions (reported in RMRR structures) when transferring control to system software such as on *ExitBootServices()*. System software may program DMA remapping hardware to block DMA outside of RMRR, except for memory explicitly registered by device drivers with system software. • Bits 3-7: Reserved (0).
Reserved	10	38	Reserved (0).
Remapping Structures[]	-	48	A list of structures. The list will contain one or more DMA Remapping Hardware Unit Definition (DRHD) structures, and zero or more Reserved Memory Region Reporting (RMRR) and Root Port ATS Capability Reporting (ATSR) structures. These structures are described below.

8.2 Remapping Structure Types

The following types of remapping structures are defined. All remapping structures start with a 'Type' field (two bytes) followed by a 'Length' field (two bytes) indicating the size in bytes of the structure (including sub-structures).

Value	Description
0	DMA Remapping Hardware Unit Definition (DRHD) Structure
1	Reserved Memory Region Reporting (RMRR) Structure
2	Root Port ATS Capability Reporting (ATSR) Structure
3	Remapping Hardware Static Affinity (RHSA) Structure
4	ACPI Name-space Device Declaration (ANDD) Structure
5	SoC Integrated Address Translation Cache (SATC) Reporting Structure
6	SoC Integrated Device Property (SIDP) Reporting Structure
>6	Reserved for future use. For forward compatibility, software skips structures it does not comprehend by skipping the appropriate number of bytes indicated by the Length field.

BIOS implementations must report these remapping structure types in numerical order. i.e., All remapping structures of type 0 (DRHD) enumerated before remapping structures of type 1 (RMRR), and so forth.

8.3 DMA Remapping Hardware Unit Definition Structure

A DMA-remapping hardware unit definition (DRHD) structure uniquely represents a remapping hardware unit present in the platform. There must be at least one instance of this structure for each PCI segment in the platform.

Field	Byte Length	Byte Offset	Description
Type	2	0	0 - DMA Remapping Hardware Unit Definition (DRHD) structure
Length	2	2	Varies (16 + size of Device Scope Structure)
Flags	1	4	Bit 0: INCLUDE_PCI_ALL • If Clear, this remapping hardware unit has under its scope only devices in the specified Segment that are explicitly identified through the 'Device Scope' field. The device can be of any type as described by the 'Type' field in the Device Scope Structure including (but not limited to) I/OxAPIC and HPET. • If Set, this remapping hardware unit has under its scope all PCI compatible devices in the specified Segment, except devices reported under the scope of other remapping hardware units for the same Segment. If a DRHD structure with INCLUDE_PCI_ALL flag Set is reported for a Segment, it must be enumerated by BIOS after all other DRHD structures for the same Segment[1]. A DRHD structure with INCLUDE_PCI_ALL flag Set may use the 'Device Scope' field to enumerate I/OxAPIC and HPET devices under its scope. Bits 1-7: Reserved.
Size	1	5	Bits 3:0: Indicates the size of the remapping hardware register set for this remapping unit. If the value in this field is N, the size of the register set is 2^N 4 KB pages (2^{N+12} bytes). Bits 4–7: Reserved.
Segment Number	2	6	The PCI Segment associated with this unit.
Register Base Address	8	8	Base address of remapping hardware register-set for this unit. This address must be aligned according to the size of the register set size reported in the Size field of this structure.
Device Scope []	-	16	The Device Scope structure contains zero or more Device Scope Entries that identify devices in the specified segment and under the scope of this remapping hardware unit. The Device Scope structure is described in Section 8.3.1. For SoC integrated SR-IOV RCIEPs, only the SR-IOV Physical Function (PF) is enumerated in this table. (Refer to Section 8.3.3 regarding VFs)

1. On platforms with multiple PCI segments, any of the segments can have a DRHD structure with INCLUDE_PCI_ALL flag Set.

8.3.1 Device Scope Structure

The Device Scope Structure is made up of Device Scope Entries. Each Device Scope Entry may be used to indicate a PCI endpoint device, a PCI sub-hierarchy, or devices such as I/OxAPICs or HPET (High Precision Event Timer).

In this section, the generic term 'PCI' is used to describe conventional PCI, PCI-X, and PCI Express devices. Similarly, the term 'PCI-PCI bridge' is used to refer to conventional PCI bridges, PCI-X bridges, PCI Express root ports, or downstream ports of a PCI Express switch.

A PCI sub-hierarchy is defined as the collection of PCI controllers that are downstream to a specific PCI-PCI bridge. To identify a PCI sub-hierarchy, the Device Scope Entry needs to identify only the parent PCI-PCI bridge of the sub-hierarchy.

Field	Byte Length	Byte Offset	Description
Type	1	0	The following values are defined for this field. • 0x01: PCI Endpoint Device - The device identified by the 'Path' field is a PCI endpoint device. This type must not be used in Device Scope of DRHD structures with INCLUDE_PCI_ALL flag Set. • 0x02: PCI Sub-hierarchy - The device identified by the 'Path' field is a PCI-PCI bridge. In this case, the specified bridge device and all its downstream devices are included in the scope. This type must not be in Device Scope of DRHD structures with INCLUDE_PCI_ALL flag Set. • 0x03: IOAPIC - The device identified by the 'Path' field is an I/O APIC (or I/O SAPIC) device, enumerated through the ACPI MADT I/O APIC (or I/O SAPIC) structure. • 0x04: MSI_CAPABLE_HPET[1] - The device identified by the 'Path' field is an HPET Timer Block capable of generating MSI (Message Signaled interrupts). HPET hardware is reported through ACPI HPET structure. • 0x05: ACPI_NAMESPACE_DEVICE - The device identified by the 'Path' field is an ACPI name-space enumerated device capable of generating DMA and/or MSI requests. Other values for this field are reserved for future use.
Length	1	1	Length of this Entry in Bytes. (6 + X), where X is the size in bytes of the "Path" field.
Flags	1	2	• This field is reserved for future use when this device scope entry appears outside of an SIDP structure. • This field is reserved for future use when the 'Type' field has a value other than 01h or 05h. • Bit 0: REQ_WO_PASID_NESTED_NOTALLOWED: For this Source ID, it is recommended that system software not program the PGTT field in the PASID Table entry, associated with RID_PASID, with a value of 011b (Nested Translation). • Bit 1: REQ_WO_PASID_PWSNP_NOTALLOWED: For this Source ID, system software must not program the PWSNP field in the PASID Table entry, associated with RID_PASID, with a value of 1b. • Bit 2: REQ_WO_PASID_PGSNP_NOTALLOWED: For this Source ID, system software must not program the PGSNP field in the PASID Table entry, associated with RID_PASID, with a value of 1b. Additionally, system software must not Set the SNP field in any leaf second-stage paging structure entries used by RID_PASID from this Source ID. • Bit 3: ATC_HARDENED: The device with this Source ID has ATC that is validated per requirements described in Section 4.4. It is recommended that system software enable ATC for this device. • Bit 4: ATC_REQUIRED: The device with this Source ID has a functional requirement to enable its ATC (via the ATS capability) for device operation. • Bits 5-7: Reserved.
Reserved	1	3	Reserved for future use.

Field	Byte Length	Byte Offset	Description
Enumeration ID	1	4	When the 'Type' field indicates 'IOAPIC', this field provides the I/O APICID as provided in the I/O APIC (or I/O SAPIC) structure in the ACPI MADT (Multiple APIC Descriptor Table). When the 'Type' field indicates 'MSI_CAPABLE_HPET', this field provides the 'HPET Number' as provided in the ACPI HPET structure for the corresponding Timer Block. When the 'Type' field indicates 'ACPI_NAMESPACE_DEVICE', this field provides the "ACPI Device Number" as provided in the ACPI Name-space Device Declaration (ANDD) structure for the corresponding ACPI device. This field is reserved for future use for all other 'Type' fields.
Start Bus Number	1	5	This field describes the bus number (bus number of the first PCI Bus produced by the PCI Host Bridge) under which the device identified by this Device Scope resides. For Device Scope Entries with Type value of 0x4 (HPET) this field describes the upper 8 bits (Bus) of the unique 16-bit source-id allocated by the platform for the MSI-capable HPET Timer Block. For Device Scope Entries with Type value of 0x5 (ACPI_NAMESPACE_DEVICE) this field describes the upper 8 bits (Bus) of the unique 16-bit source-id allocated by the platform for the ACPI name-space device.
Path	2 * N	6	For Device Scope Entries with Type value of 0x1, 0x2 or 0x3 this field describes the hierarchical path from the Host Bridge to the device specified by the Device Scope Entry. For example, a device in a N-deep hierarchy is identified by N {PCI Device Number, PCI Function Number} pairs, where N is a positive integer. Even offsets contain the Device numbers, and odd offsets contain the Function numbers. The first {Device, Function} pair resides on the bus identified by the 'Start Bus Number' field. Each subsequent pair resides on the bus directly behind the bus of the device identified by the previous pair. The identity (Bus, Device, Function) of the target device is obtained by recursively walking down these N {Device, Function} pairs. If the 'Path' field length is 2 bytes (N=1), the Device Scope Entry identifies a 'Root-Complex Integrated Device'. The requester-id of 'Root-Complex Integrated Devices' are static and not impacted by system software bus rebalancing actions. If the 'Path' field length is more than 2 bytes (N > 1), the Device Scope Entry identifies a device behind one or more system software visible PCI-PCI bridges. Bus rebalancing actions by system software modifying bus assignments of the device's parent bridge impacts the bus number portion of device's requester-id. For Device Scope Entries with Type value of 0x4 (HPET) this field describes the lower 8 bits {Device, Function} of the unique 16-bit source-id allocated by the platform for the MSI-capable HPET Timer Block. For Device Scope Entries with Type value of 0x5 (ACPI_NAMESPACE_DEVICE) this field describes the lower 8 bits {Device, Function} of the unique 16-bit source-id allocated by the platform for the ACPI name-space device.

1. An HPTE Timer Block is capable of MSI interrupt generation if any of the Timers in the Timer Block reports FSB_INTERRUPT_DELIVERY capability in the Timer Configuration and Capability Registers. HPET Timer Blocks not capable of MSI interrupt generation (and instead have their interrupts routed through I/OxAPIC) are not reported in the Device Scope.

The following pseudocode describes how to identify the device specified through a Device Scope structure:

```
n = (DevScope.Length - 6) / 2;                  // number of entries in the 'Path' field
type = DevScope.Type;                           // type of device
bus = DevScope.StartBusNum;                     // starting bus number
dev = DevScope.Path[0].Device;                  // starting device number
func = DevScope.Path[0].Function;               // starting function number
i = 1;
while (--n) {
      bus = read_secondary_bus_reg(bus, dev, func);// secondary bus# from config reg.
      dev = DevScope.Path[i].Device;                // read next device number
      func = DevScope.Path[i].Function;             // read next function number
      i++;
}
source_id = {bus, dev, func};
target_device = {type, source_id};        // if 'type' indicates 'IOAPIC', DevScope.EnumID
                                          // provides the I/O APICID as reported in the ACPI MADT
```

8.3.1.1 Reporting Scope for I/OxAPICs

Interrupts from devices that only support (or are only enabled for) legacy interrupts are routed through the I/OxAPICs in the platform. Each I/OxAPIC in the platform is reported to system software through ACPI MADT (Multiple APIC Descriptor Tables). Some platforms may also expose I/OxAPICs as PCI-discoverable devices.

For platforms reporting interrupt remapping capability (INTR_REMAP flag Set in the DMAR structure), each I/OxAPIC in the platform reported through ACPI MADT must be explicitly enumerated under the Device Scope of the appropriate remapping hardware units (even for remapping hardware unit reported with INCLUDE_PCI_ALL flag Set in DRHD structure).

- For I/OxAPICs that are PCI-discoverable, the source-id for such I/OxAPICs (computed using the above pseudocode from its Device Scope structure) must match its PCI requester-id effective at the time of boot.
- For I/OxAPICs that are not PCI-discoverable:
  - If the 'Path' field in Device Scope has a size of 2 bytes, the corresponding I/OxAPIC is a Root-Complex integrated device. The 'Start Bus Number' and 'Path' field in the Device Scope structure together provides the unique 16-bit source-id allocated by the platform for the I/OxAPIC. Examples are I/OxAPICs integrated to the IOH and south bridge (ICH) components.
  - If the 'Path' field in Device Scope has a size greater than 2 bytes, the corresponding I/OxAPIC is behind some software visible PCI-PCI bridge. In this case, the 'Start Bus Number' and 'Path' field in the Device Scope structure together identifies the PCI-path to the I/OxAPIC device. Bus rebalancing actions by system software modifying bus assignments of the device's parent bridge impacts the bus number portion of device's source-id. Examples are I/OxAPICs in PCI Express-to-PCI-X bridge components in the platform.

8.3.1.2 Reporting Scope for MSI Capable HPET Timer Block

High Precision Event Timer (HPET) Timer Block supporting Message Signaled Interrupt (MSI) interrupts may generate interrupt requests directly to the Root-Complex (instead of routing through I/OxAPIC). Platforms supporting interrupt remapping must explicitly enumerate any MSI-capable HPET Timer Block in the platform through the Device Scope of the appropriate remapping hardware unit. In this case, the 'Start Bus Number' and 'Path' field in the Device Scope structure together provides the unique 16-bit source-id allocated by the platform for the MSI-capable HPET Timer Block.

8.3.1.3 Reporting Scope for ACPI Name-space Devices

Some platforms may support ACPI name-space enumerated devices that are capable of generating DMA and/or MSI requests. Platforms supporting DMA and/or interrupt remapping must explicitly declare any such ACPI name-space devices in the platform through ACPI Name-space Device Declaration (ANDD) structure and enumerate them through the Device Scope of the appropriate remapping hardware unit. In this case, the 'Start Bus Number' and 'Path' field in the Device Scope structure together provides the unique 16-bit source-id allocated by the platform for the ACPI name-space device. Multiple ACPI name-space devices that share common bus-mastering hardware resources may share a common source-id. For example, some Intel® SoC platforms supports a Low Power Sub System (LPSS) in the south-bridge, that shares a common DMA resource across multiple ACPI name-space devices such as I2C, SPI, UART, and SDIO.

8.3.1.4 Device Scope Example

This section provides an example platform configuration with multiple remapping hardware units. The configurations described are hypothetical examples, only intended to illustrate the Device Scope structures.

Figure 8-1 illustrates a platform configuration with a single PCI segment and host bridge (with a starting bus number of 0), and supporting three remapping hardware units as follows:

1. Remapping hardware unit #1 has under its scope an integrated device at (dev:func) of (4:0) which does not support ATS and an integrated device at (dev:func) at (5:0) which does support ATS.
2. Remapping hardware unit #2 has under its scope all devices downstream to the PCI Express root port located at (dev:func) of (7:0).
3. Remapping hardware unit #3 has under its scope all other PCI compatible devices in the platform not explicitly under the scope of the other remapping hardware units. In this example, this includes the integrated device at (dev:func) of (30:0), and all the devices attached to the south bridge component. The I/OxAPIC in the platform (I/O APICID = x) is under the scope of this remapping hardware unit, and has a BIOS assigned bus/dev/function number of (0,31,7). The HPET in the platform is under the scope of this remapping hardware unit, and has a BIOS assigned bus/dev/function number of (0,31,6).

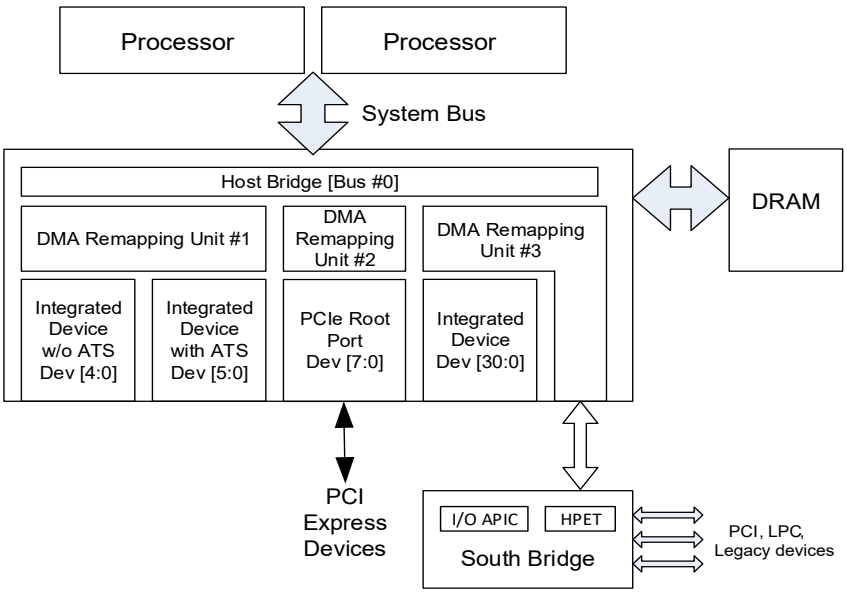


Figure 8-1. Hypothetical Platform Configuration

To describe the platform configuration in Figure 8-1, BIOS creates DMAR structures in memory as shown in the Table 33.

Table 33. DMAR Structure for Platform Shown in Figure 8-1

Offset	0	1	2	3	4	5	6	7	
	DMA Remapping Reporting Structure								
0	Signature				Length (152)				
8	Revision (1)	Check-sum	OEMID						
16	OEM Table ID								
24	OEM Revision				Creator ID				
32	Creator Revision				HAW	Flags (1)	Reserved		
40	Reserved								
	DMA Remapping Hardware Unit Definition (DRHD) Structure #1								DRHD #1
48	Type (0)		Length (32)		Flags (0)	Reserved (0)	Segment Number (0)		
56	Register Base Address								
	Device Scope Structure (Dev4)								
64	Type (1)	Length (8)	Reserved (0)		Enumera-tionID (0)	StartBus-Num (0)	Path.Dev (4)	Path.Func (0)	
	Device Scope Structure (Dev5)								
72	Type (1)	Length (8)	Reserved (0)		Enumera-tionID (0)	StartBus-Num (0)	Path.Dev (5)	Path.Func (0)	
	DMA Remapping Hardware Unit Definition (DRHD) Structure #2								DRHD #2
80	Type (0)		Length (24)		Flags (0)	Reserved (0)	Segment Number (0)		
88	Register Base Address								
	Device Scope Structure (Dev7)								
96	Type (2)	Length (8)	Reserved (0)		Enumera-tionID (0)	StartBus-Num (0)	Path.Dev (7)	Path.Func (0)	

Table 33. DMAR Structure for Platform Shown in Figure 8-1

Offset	0	1	2	3	4	5	6	7	
	DMA Remapping Hardware Unit Definition (DRHD) Structure #3								DRHD #3
104	Type (0)		Length (32)		Flags (1)	Reserved (0)	Segment Number (0)		
112	Register Base Address								
	Device Scope Structure (IO APIC)								
120	Type (3)	Length (8)	Reserved (0)		Enumera-tionID (x)	StartBus-Num (0)	Path.Dev (31)	Path.Func (7)	
	Device Scope Structure (HPET)								
128	Type (4)	Length (8)	Reserved (0)		Enumera-tionID (y)	StartBus-Num (0)	Path.Dev (31)	Path.Func (6)	
	SoC Integrated Address Translation Cache (SATC) Reporting Structure								SATC - Dev#5
136	Type (5)		Length (16)		Flags (1)	Reserved (0)	Segment Number (0)		
	Device Scope Structure (Dev5)								
144	Type (1)	Length (8)	Reserved (0)		Enumera-tionID (0)	StartBus-Num (0)	Path.Dev (5)	Path.Func (0)	
152									

This platform requires 3 DRHD structures. The Device Scope fields in each DRHD structure are described as below.

- Device Scope for remapping hardware unit #1 contains two Device Scope Entries.
  - Device Scope Entry at offset 64 [1, 8, 0, 0, 0, 4, 0] with type field value of 0x1 is used to conclude that remapping unit #1 includes an endpoint device at PCI Segment 0, Bus 0, Device 4 and function 0.
  - Device Scope Entry at offset 72 [1, 8, 0, 0, 0, 5, 0] with type field value of 0x1 is used to conclude that remapping unit #1 includes an endpoint device at PCI Segment 0, Bus 0, Device 5 and function 0.
- Device Scope for remapping hardware unit #2 contains only one Device Scope Entry, identified as [2, 8, 0, 0, 0, 7, 0] (at offset 96).
  - System Software uses the Entry Type field value of 0x02 to conclude that all devices downstream of the PCI-PCI bridge device at PCI Segment 0, Bus 0, Device 7, and Function 0 are within the scope of this remapping hardware unit.
- Device Scope for remapping hardware unit #3 contains two Device Scope Entries. Also, the DHRD structure for remapping hardware unit #3 indicates the INCLUDE_PCI_ALL flag. This hardware unit must be the last in the list of hardware unit definition structures reported.
  - System software uses the INCLUDE_PCI_ALL flag to conclude that all PCI compatible devices that are not explicitly enumerated under other remapping hardware units are in the scope of remapping unit #3.
  - Device Scope Entry at offset 120 [3, 8, 0, x, 0, 31, 7] with Type field value of 0x3 is used to conclude that the I/OxAPIC (with I/O APICID=x and source-id of [0, 31, 7]) is under the scope of remapping hardware unit #3.
  - Device Scope Entry at offset 128 [4, 8, 0, y, 0, 31, 6] with Type field value of 0x4 is used to conclude that the HPET (with HPET_ID =y and source-id of [0, 31, 6]) is under the scope of remapping hardware unit #3.

8.3.2 Implications for ARI

The PCI Express Alternate Routing-ID Interpretation (ARI) Extended Capability enables endpoint devices behind ARI-capable PCI Express Root/Switch ports to support 'Extended Functions', beyond the limit of 8 'Traditional Functions'. When ARI is enabled, 'Extended Functions' on an endpoint are under the scope of the same remapping unit as the 'Traditional Functions' on the endpoint.

8.3.3 Implications for SR-IOV

The PCI Express Single-Root I/O Virtualization (SR-IOV) Capability enables a Physical Function on an endpoint device to support multiple Virtual Functions (VFs). A Physical Function can be a Traditional Function or an ARI Extended Function. When SR-IOV is enabled, Virtual Functions of a Physical Function are under the scope of the same remapping unit as the Physical Function. Properties described by SATC and SIDP entries for a Physical Function also apply to its Virtual Functions.

8.3.4 Implications for PCI/PCI Express* Hot Plug

Conventional PCI and PCI Express defines support for hot plug. Devices hot plugged behind a parent device (PCI* bridge or PCI Express root/switch port) are under the scope of the same remapping unit as the parent device.

8.3.5 Implications with PCI Resource Rebalancing

System software may perform PCI resource rebalancing to dynamically reconfigure the PCI sub-system (such as on PCI or PCI Express hot-plug). Resource rebalancing can result in system software changing the bus number allocated for a device. Such rebalancing only changes the device's identity (Source-ID). The device will continue to be under the scope of the same remapping unit as it was before rebalancing. System software is responsible for tracking device identity changes and resultant impact to Device Scope.

8.3.6 Implications with Provisioning PCI BAR Resources

System BIOS typically provisions the initial PCI BAR resources for devices present at time of boot. To conserve physical address space (especially below 4GB) consumed by PCI BAR resources, BIOS implementations traditionally use compact allocation policies resulting in BARs of multiple devices/functions residing within the same system-base-page-sized region (4KB for Intel® 64 platforms). However, allocating BARs of multiple devices in the same system-page-size region imposes challenges to system software using remapping hardware to assign these devices to isolated domains.

For platforms supporting remapping hardware, BIOS implementations should avoid allocating BARs of otherwise independent devices/functions in the same system-base-page-sized region.

8.4 Reserved Memory Region Reporting Structure

Section 3.16 describes the details of BIOS allocated reserved memory ranges that may be DMA targets. BIOS may report each such reserved memory region through the RMRR structures, along with the devices that requires access to the specified reserved memory region. Reserved memory ranges that are either not DMA targets, or memory ranges that may be target of BIOS initiated DMA only during pre-boot phase (such as from a boot disk drive) must not be included in the reserved memory region reporting. The base address of each RMRR region must be 4KB aligned and the size must be an integer multiple of 4KB.

BIOS must report the RMRR reported memory addresses as reserved (or as EFI runtime) in the system memory map returned through methods such as INT15, EFI GetMemoryMap etc. The reserved memory region reporting structures are optional. If there are no RMRR structures, the system software concludes that the platform does not have any reserved memory ranges that are DMA targets.

The RMRR regions are expected to be used for legacy usages (such as USB, UMA Graphics, etc.) requiring reserved memory. Platform designers should avoid or limit use of reserved memory regions since these require system software to create holes in the DMA virtual address range available to system software and its drivers.

Field	Byte Length	Byte Offset	Description
Type	2	0	1 - Reserved Memory Region Reporting Structure
Length	2	2	Varies (24 + size of Device Scope structure)
Reserved	2	4	Reserved (0).
Segment Number	2	6	PCI Segment Number associated with devices identified through the Device Scope field.
Reserved Memory Region Base Address	8	8	Base address of 4KB-aligned reserved memory region.
Reserved Memory Region Limit Address	8	16	Last address of the reserved memory region. Value in this field must be greater than the value in Reserved Memory Region Base Address field. The reserved memory region size (Limit - Base + 1) must be an integer multiple of 4KB.
Device Scope[]	-	24	The Device Scope structure contains one or more Device Scope entries that identify devices requiring access to the specified reserved memory region. The devices identified in this structure must be devices under the scope of one of the remapping hardware units reported in DRHD.

8.5 Root Port ATS Capability Reporting Structure

This structure is applicable only for platforms supporting Device-TLBs as reported through the Extended Capability Register. For each PCI Segment in the platform that supports Device-TLBs, BIOS provides an ATSR structure. The ATSR structures identifies PCI Express Root-Ports supporting Address Translation Services (ATS) transactions. Software must enable ATS on endpoint devices behind a Root Port only if the Root Port is reported as supporting ATS transactions.

Field	Byte Length	Byte Offset	Description
Type	2	0	2 - Root Port ATS Capability Reporting Structure
Length	2	2	Varies (8 + size of Device Scope Structure)
Flags	1	4	• Bit 0: ALL_PORTS: If Set, indicates all PCI Express Root Ports in the specified PCI Segment supports ATS transactions. If Clear, indicates ATS transactions are supported only on Root Ports identified through the Device Scope field. • Bits 1-7: Reserved.
Reserved	1	5	Reserved (0).
Segment Number	2	6	The PCI Segment associated with this ATSR structure.
Device Scope []	-	8	If the ALL_PORTS flag is Set, the Device Scope structure is omitted. If ALL_PORTS flag is Clear, the Device Scope structure contains Device Scope Entries that identifies Root Ports supporting ATS transactions. The Device Scope structure is described in Section 8.3.1. All Device Scope Entries in this structure must have a Device Scope Entry Type of 02h.

8.6 Remapping Hardware Static Affinity Structure

Remapping Hardware Status Affinity (RHSA) structure is applicable for platforms supporting non-uniform memory (NUMA), where Remapping hardware units spans across nodes. This optional structure provides the association between each Remapping hardware unit (identified by its respective Base Address) and the proximity domain to which that hardware unit belongs. Such platforms, report the proximity of processor and memory resources using ACPI Static Resource Affinity (SRAT) structure. To optimize remapping hardware performance, software may allocate translation structures referenced by a remapping hardware unit from memory in the same proximity domain. Similar to SRAT, the information in the RHSA structure is expected to be used by system software during early initialization, when evaluation of objects in the ACPI name-space is not yet possible.

Field	Byte Length	Byte Offset	Description
Type	2	0	3 - Remapping Hardware Static Affinity Structure. This is an optional structure and intended to be used only on NUMA platforms with Remapping hardware units and memory spanned across multiple nodes. When used, there must be a Remapping Hardware Static Affinity structure for each Remapping hardware unit reported through DRHD structure.
Length	2	2	Length is 20 bytes
Reserved	4	4	Reserved (0).
Register Base Address	8	8	Register Base Address of this Remap hardware unit reported in the corresponding DRHD structure.
Proximity Domain [31:0]	4	16	Proximity Domain to which the Remap hardware unit identified by the Register Base Address field belongs.

8.7 ACPI Name-space Device Declaration Structure

An ACPI Name-space Device Declaration (ANDD) structure uniquely represents an ACPI name-space enumerated device capable of issuing DMA requests in the platform. ANDD structures are used in conjunction with Device-Scope entries of type 'ACPI_NAMESPACE_DEVICE'. Refer to Section 8.3.1 for details on Device-Scope entries.

Field	Byte Length	Byte Offset	Description
Type	2	0	4 - ACPI Name-space Device Declaration (ANDD) structure
Length	2	2	Length of this Entry in Bytes. (8 + N), where N is the size in bytes of the "ACPI Object Name" field.
Reserved	3	4	Reserved (0).
ACPI Device Number	1	7	Each ACPI device enumerated through an ANDD structure must have a unique value for this field. To report an ACPI device with 'ACPI Device Number' value of X, under the scope of a DRHD unit, a Device-Scope entry of type 'ACPI_NAMESPACE_DEVICE' is used with value of X in the Enumeration ID field. The 'Start Bus Number' and 'Path' fields in the Device-Scope together provides the 16-bit source-id allocated by the platform for the ACPI device.
ACPI Object Name	N	8	ASCII, null terminated, string that contains a fully qualified reference to the ACPI name-space object that is this device. (For example, "_SB.I2C0" represents the ACPI object name for an embedded I2C controller in southbridge; Quotes are omitted in the data field). Refer to ACPI specification for fully qualified references for ACPI name-space objects.

8.8 SoC Integrated Address Translation Cache Reporting Structure

The SoC Integrated Address Translation Cache (SATC) reporting structure identifies devices that have address translation cache (ATC), as defined by the PCI Express Base Specification, and that is validated per requirements described in Section 4.4. It is recommended that system software enable ATC for this device.

Field	Byte Length	Byte Offset	Description
Type	2	0	5 - SoC Integrated Address Translation Cache (SATC) Reporting Structure
Length	2	2	Varies (8 + size of Device Scope Structure)
Flags	1	4	• Bit 0: ATC_REQUIRED: If Set, indicates that every SoC integrated device enumerated in this table has a functional requirement to enable its ATC (via the ATS capability) for device operation. If Clear, any device enumerated in this table can operate when its respective ATC is not enabled (albeit with reduced performance or functionality). • Bits 1-7: Reserved.
Reserved	1	5	Reserved (0).
Segment Number	2	6	The PCI Segment associated with this SATC structure. All SoC integrated devices within a PCI segment with the same value for the Flags field must be enumerated in the same SATC structure.
Device Scope []	-	8	The Device Scope structure contains Device Scope Entries that identify SoC integrated devices (in the specified PCI segment) with address translation caches (ATC). Such ATCs are validated per requirements described in Section 4.4 in this specification. The Device Scope structure is described in Section 8.3.1 of this specification. All Device Scope Entries in this structure must have a Device Scope Entry Type of 01h (PCI Endpoint Device) with path field length of 2 bytes (indicating it is an RCIEP). Such devices must also enumerate PCISIG defined ATS capability. For SoC integrated SR-IOV RCIEPs with ATC, only the SR-IOV Physical Function (PF) is enumerated in this table. (Refer to Section 8.3.3 regarding VFs.)

An example of the SATC structure for platform configuration in Figure 8-1 is shown in Table 33. The SATC structure is at offset 128 and identified by type field value of 0x5.

- System software uses the SATC structure to conclude that the devices listed underneath are validated per the requirements described in Section 4.4 and that software is recommended to enable ATC on such devices.
- Because there is only one SoC integrated device with address translation cache in our example platform configuration, the SATC structure has only one Device Scope Entry identified by [1, 8, 0, 0, 0, 5, 0].
- System software uses the ATC_REQUIRED flag to conclude that it must enable ATC on the device via the ATS capability for the device to be functional.

8.9 SoC Integrated Device Property Reporting Structure

The SoC Integrated Device Property (SIDP) reporting structure identifies devices that have special properties and that may put restrictions on how system software must configure remapping structures that govern such devices in a platform where remapping hardware is enabled. SIDP uses the Flags field in the Device Scope Entry to indicate the special properties of each device.

System software that uses the SIDP structure must ignore SATC structures that may also be present.

Field	Byte Length	Byte Offset	Description
Type	2	0	6 - SoC Integrated Device Property (SIDP) Reporting Structure
Length	2	2	Varies (8 + size of Device Scope Structure)
Reserved	2	4	Reserved for future use.
Segment Number	2	6	The PCI Segment associated with this SIDP structure.
Device Scope []	-	8	The Device Scope structure contains Device Scope Entries that identify SoC integrated devices (in the specified PCI segment) with special properties/restrictions that system software needs to be aware of. The Device Scope structure is described in Section 8.3.1 of this specification. All Device Scope Entries in this structure must have a Device Scope Entry Type of 01h (PCI Endpoint Device) or Type 05h (ACPI Device) with path field length of 2 bytes (indicating it is an RCIEP). For SoC integrated SR-IOV RCIEPs with special properties/restriction, only the SR-IOV Physical Function (PF) is enumerated in this table. (Refer to Section 8.3.3 regarding VFs).

8.10 Remapping Hardware Unit Hot Plug

Remapping hardware units are implemented in Root-Complex components such as the I/O Hub (IOH). Such Root-Complex components may support hot-plug capabilities within the context of the interconnect technology supported by the platform. These hot-pluggable entities consist of an I/O subsystem rooted in a ACPI host bridge. The I/O subsystem may include Remapping hardware units, in addition to I/O devices directly attached to the host bridge, PCI/PCI Express sub-hierarchies, and I/OxAPICs.

The ACPI DMAR static tables and sub-tables defined in previous sections enumerate the remapping hardware units present at platform boot-time. Following sections illustrates the ACPI methods for dynamic updates to remapping hardware resources, such as on I/O hub hot-plug. Following sections assume familiarity with ACPI 3.0 specification and system software support for host-bridge hot-plug.

8.10.1 ACPI Name Space Mapping

ACPI defines Device Specific Method (_DSM) as a method that enables ACPI devices to provide device specific functions without name-space conflicts. A Device Specific Method (_DSM) with the following GUID is used for dynamic enumeration of remapping hardware units.

GUID
D8C1A3A6-BE9B-4C9B-91BF-C3CB81FC5DAF

The _DSM method would be located under the ACPI device scope where the platform wants to expose the remapping hardware units. For example, ACPI name-space includes representation for hot-pluggable I/O hubs in the system as a ACPI host bridges. For Remapping hardware units implemented in I/O hub component, the _DSM method would be under the respective ACPI host bridge device.

The _DSM method supports the following function indexes.

Function Index	Description
0	Query function as specified in ACPI 3.0 specification. Returns which of the below function indexes are supported.
1	Return DMA Remapping Hardware Definition (DRHD) Structures[1]
2	Return Root Port ATS Capability Reporting (ATSR) Structure
3	Return Remapping Hardware Static Affinity (RHSA) Structure

1. Reserved Memory Region Reporting (RMRR) structures are not reported via _DSM, since use of reserved memory regions are limited to legacy devices (USB, iGFX etc.) that are not applicable for hot-plug.

8.10.2 ACPI Sample Code

This section illustrates sample ASL code for enumerating remapping resources in an I/O hub.

```
Scope \_SB {
    ....
    Device (IOHn) {                          // host bridge representation for I/O Hub n
        Name (_HID, EISAID("PNP0A08"))
        Name (_CID, EISAID("PNP0A03"))
        ...
        Method (_DSM, 0, NotSerialized) {    // Device specific method
            Switch(Arg0) {
                case (ToUUID("D8C1A3A6-BE9B-4C9B-91BF-C3CB81FC5DAF")) {
                    Switch (Arg2) {          // No switch for Arg1, since only one version of this method is supported
                        case(0): {Return (Buffer() {0x1F})}     // function indexes 1-4 supported
                        case(1): {Return DRHDT} // DRHDT is a buffer containing relavent DRHD structures for I/O Hub n
                        case(2): {Return ATSRT}  // ATSRT is a buffer containing relavent ATSR structure for I/O Hub n
                        case(3): {Return RHSAT} // RHSAT is a buffer containing relavent RHSAT structure for I/O Hub n
                    }
                }
            }
        }
        ...
    }
} // end of Scope SB
```

8.10.3 Example Remapping Hardware Reporting Sequence

The following sequence may be practiced for enumerating remapping hardware resources at boot time.

- Platform prepares name space and populates the ACPI DMAR static reporting tables to be reported to system software. These DMAR static tables report only the remapping hardware units that are present at time of boot, and accessible by system software.

The following sequence may be practiced on I/O hub hot-add:

- Platform notifies system software via ACPI the presence of new resources.
- System software evaluates the handle to identify the object of the notify as ACPI host bridge (I/O hub)
- If System software is able to support the hot-add of host bridge, it calls _OST to indicate success.
- System software evaluates _DSM method to obtain the remapping hardware resources associated with this host bridge (I/O hub)[1].
- System software initializes and prepares the remapping hardware for use.
- System software continues with host-bridge hot-add processing, including discovery and configuration of I/O hierarchy below the hot-added host-bridge.

1. Invoking the _DSM method does not modify the static DMAR tables. System software must maintain the effective DMAR information comprehending the initial DMAR table reported by the platform, and any remapping hardware units added or removed via _DSM upon host bridge hot-add or hot-remove.

9 Translation Structure Formats

This chapter describes the memory-resident structures for DMA and interrupt remapping. Hardware must access structure entries that are 64-bit or 128-bit atomically. Hardware must update a 512-bit Posted Interrupt Descriptor (see Section 9.11 for details) atomically. Other than the Posted Interrupt Descriptor (PID), hardware is allowed to break access to larger than 128-bit entries into multiple aligned 128-bit accesses.

9.1 Root Entry

The following figure and table describe the root-entry. The Root Table Address Register points to a table of root-entries, when the Translation Table Mode (TTM) field in the register is 00b.

.

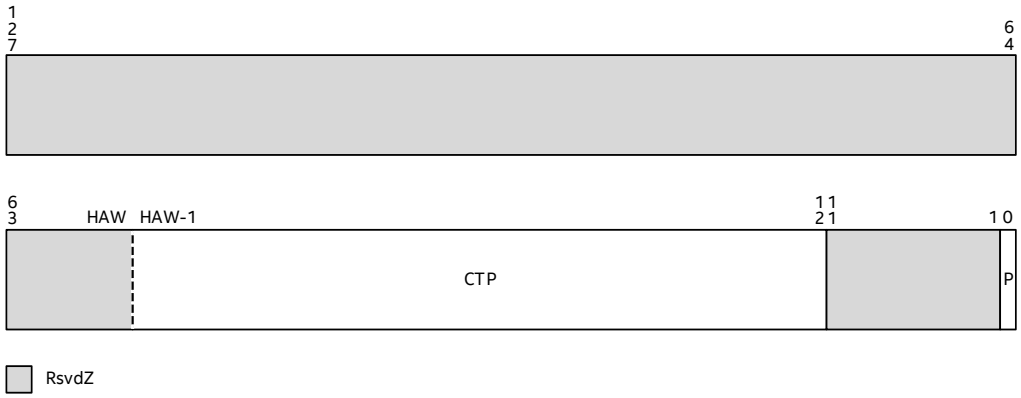


Figure 9-1. Root-Entry Format

Bits	Field	Description
127:64	R: Reserved	Reserved. Must be 0.
63:12	CTP: Context-table Pointer	Pointer to Context-table for this bus. The Context-table is 4KB in size and size-aligned. Hardware treats bits 63:HAW as reserved (0), where HAW is the host address width of the platform.
11:1	R: Reserved	Reserved. Must be 0.
0	P: Present	This field indicates whether the root-entry is present. • 0: Indicates the root-entry is not present. All other fields are ignored by hardware. • 1: Indicates the root-entry is present.

9.2 Scalable-mode Root Entry

The following figure and table describe the scalable-mode root-entry. The Root Table Address Register points to a table of scalable-mode root-entries, when the Translation Table Mode (TTM) field in the register is programmed with a value of 01b.

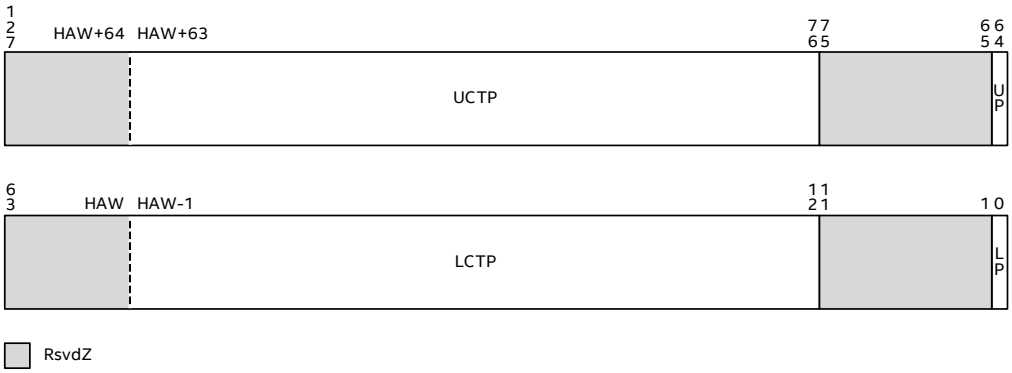


Figure 9-2. Scalable-mode Root-Entry Format

Bits	Field	Description
127:76	UCTP: Upper Context Table Pointer	This field specifies pointer to upper scalable-mode context-table for this bus. The upper scalable-mode context-table is 4KB in size and size-aligned. Hardware treats bits 127:(HAW+64) as reserved (0), where HAW is the host address width of the platform.
75:65	R: Reserved	Reserved. Must be 0.
64	UP: Upper Present	This field indicates whether the upper-half of the scalable-mode-root-entry is present. • 0: Indicates upper half of the scalable-mode root-entry is not present. Bits 127:65 are ignored by hardware. • 1: Indicates the upper-half of the scalable-mode root-entry is present.
63:12	LCTP: Lower Context Table Pointer	This field specifies pointer to lower scalable-mode context-table for this bus. The lower scalable-mode context-table is 4KB in size and size-aligned. Hardware treats bits 63:HAW as reserved (0), where HAW is the host address width of the platform.
11:1	R: Reserved	Reserved. Must be 0.
0	LP: Lower Present	This field indicates whether the lower-half of the scalable-mode root-entry is present. • 0: Indicates lower half of the scalable-mode root-entry is not present. Bits 63:1 are ignored by hardware. • 1: Indicates the lower-half of the scalable-mode root-entry is present.

9.3 Context Entry

The following figure and table describe the context-entry. Context-entries support translation of requests-without-PASID. Context-entries are referenced through root-entries described in Section 9.1.

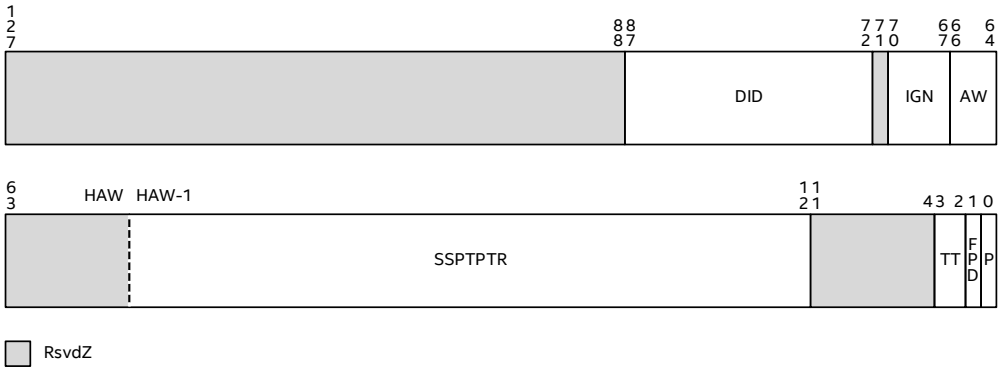


Figure 9-3. Context-Entry Format

Bits	Field	Description
127:88	R: Reserved	Reserved. Must be 0.
87:72	DID: Domain Identifier	Identifier for the domain to which this context-entry maps. Hardware may use the domain identifier to tag its internal caches. The Capability Register reports the domain-id width supported by hardware. For implementations supporting less than 16-bit domain-ids, unused bits of this field are treated as reserved by hardware. For example, for implementation supporting 8-bit domain-ids, bits 87:80 of this field are treated as reserved. Context-entries programmed with the same domain identifier must always reference same address translation (SSPTPTR field). Context-entries referencing same address translation are recommended to be programmed with same domain id for hardware efficiency. When Caching Mode (CM) field in Capability Register is reported as Set, the domain-id value of zero is architecturally reserved. Software must not use domain-id value of zero when CM is Set.
71	R: Reserved	Reserved. Must be 0.
70:67	IGN: Ignored	Hardware ignores the programming of this field.

Bits	Field	Description
66:64	AW: Address Width	When the Translation-type (TT) field is 00b or 01b, this field indicates the adjusted guest-address-width (AGAW) to be used by hardware for the second-stage page-table walk. The following encodings are defined for this field: • 000b: Reserved • 001b: 39-bit AGAW (3-level page table) • 010b: 48-bit AGAW (4-level page table) • 011b: 57-bit AGAW (5-level page table) • 100b-111b: Reserved The value specified in this field must match an AGAW value supported by hardware (as reported in the SAGAW field in the Capability Register). When the Translation-type (TT) field indicates pass-through processing (10b), this field must be programmed to indicate the largest AGAW value supported by hardware. Untranslated requests-without-PASID processed through this context-entry and accessing addresses above 2^X-1 (where X is the AGAW value indicated by this field) are blocked and treated as translation faults.
63:12	SSPTPTR: Second Stage Page Translation Pointer	When the Translation-Type (TT) field is 00b or 01b, this field points to the base of second-stage paging entries (described in Section 9.8). Hardware treats bits 63:HAW as reserved (0), where HAW is the host address width of the platform. This field is ignored by hardware when Translation-Type (TT) field is 10b (pass-through).
11:4	R: Reserved	Reserved. Must be 0.
3:2	TT: Translation Type	This field is applicable only for requests-without-PASID, as hardware blocks all requests-with-PASID in legacy mode before they can use context table. • 00b: Untranslated requests are translated using second-stage paging structures referenced through the SSPTPTR field. Translated requests and Translation Requests are blocked. • 01b: Untranslated, Translated and Translation Requests are supported. This encoding is treated as reserved by hardware implementations not supporting Device-TLBs (DT=0 in Extended Capability Register). • 10b: Untranslated requests are processed as pass-through. The SSPTPTR field is ignored by hardware. Translated and Translation Requests are blocked. This encoding is treated by hardware as reserved for hardware implementations not supporting Pass Through (PT=0 in Extended Capability Register). • 11b: Reserved.
1	FPD: Fault Processing Disable	Enables or disables recording/reporting of qualified non-recoverable faults. • 0: Qualified non-recoverable faults are recorded/reported for requests processed through this context-entry. • 1: Qualified non-recoverable faults are not recorded/reported for requests processed through this context-entry. This field is evaluated by hardware irrespective of the setting of the present (P) field.
0	P: Present	• 0: Indicates the context-entry is not present. All other fields except Fault Processing Disable (FPD) field are ignored by hardware. • 1: Indicates the context-entry is present.

9.4 Scalable-Mode Context-Entry

The following figure and table describe the scalable-mode context-entry. Scalable-mode context-entries are referenced through scalable-mode root-entries described in Section 9.2.

Scalable-mode context-entries support translation of both requests-without-PASID and requests-with-PASID, through a two-level scalable-mode PASID-table structure referenced by the PASIDDIRPTR (PASID Directory Pointer) field. Requests-without-PASID are processed as if they are Requests-with-PASID with PASID value as specified by the RID_PASID field of the relevant scalable-mode context-entry, PrivilegeModeRequested value as specified by RID_PRIV field of the relevant scalable-mode context-entry.

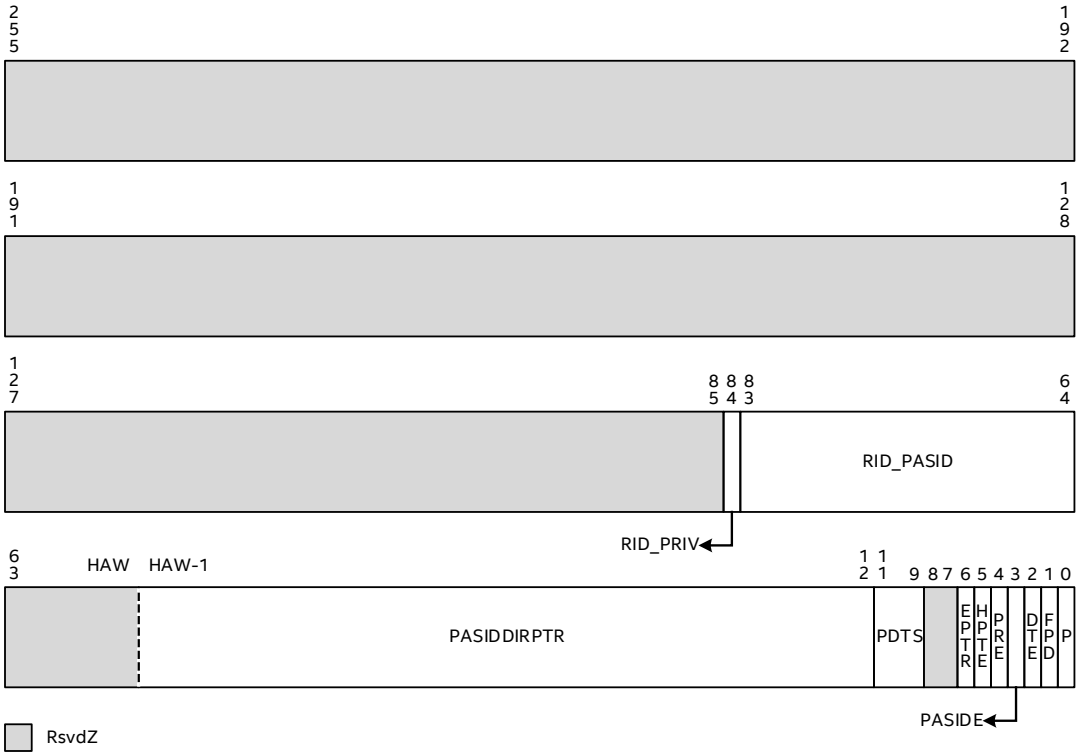


Figure 9-4. Scalable-Mode Context-Entry Format

Bits	Field	Description
255:85	R: Reserved	Reserved. Must be 0.
84	RID_PRIV: Requester ID to PrivilegeModeRequested Assignment	This field is treated as Reserved(0) for implementations not supporting RID_PRIV (RPRIVS=0 in Extended Capability Register). Requests-without-PASID processed through this scalable-mode context entry use the value specified in this field as their PrivilegeModeRequested value to determine R and W permissions for untranslated and translation requests without PASID. Completion for translation request-without-PASID that observed RID_PRIV as Set carry R and W permissions associated with a Privileged Mode Entity. Thus, it is possible for DevTLB to have an entry that is without PASID but has R and W permissions for a Privileged Entity. Implementations reporting the RPRIVS field in the Extended Capability Register as Clear use a PrivilegeModeRequested value of 0 to perform address translation for requests-without-PASID.
83:64	RID_PASID: Requester ID to PASID Assignment	This field is treated as Reserved(0) for implementations not supporting RID_PASID (RPS=0 in Extended Capability Register). Requests-without-PASID processed through this scalable-mode context entry are treated as Requests-with-PASID with PASID value specified in this field. ExecuteRequested field is treated as 0 for such requests. Implementations reporting RPS field in Extended Capability Register as Clear use a PASID value of 0 to perform address translation for requests without PASID.
63:12	PASIDDIRPTR: PASID Directory Pointer	This field points to the base of the scalable-mode PASID-directory. Hardware treats bits 63:(HAW) bits in this field as reserved (0), where HAW is the host address width of the platform. This field is always treated as Host Physical Address (HPA). Section 9.5 describes the format of entries in the scalable-mode PASID-directory.
11:9	PDTS: PASID Directory Size	Value of X in this field indicates that the PASID-directory has $2^{(X+7)}$ entries.
8:7	R: Reserved	Reserved. Must be 0.
6	EPTR: Enable PASID in Translated Requests	This field is treated as Reserved(0) for implementations not supporting PASID in Translated Requests. (PTRS=0 in Extended Capability Register) • 0: Translated requests with PASID are treated as Unsupported Request (UR). • 1: Translated requests with PASID are allowed.
5	HPTE: HPT Enable	This field is treated as Reserved(0) for implementations not supporting Host Permission Tables (HPTS=0 in Extended Capability Register) • 0: Translated requests are processed without checking the request address in the host permission tables. • 1: Processing of Translated requests include the Host Permission Table as described in Section 4.2.4.
4	PRE: Page Request Enable	This field is treated as Reserved(0) for implementations not supporting Page Requests (PRS=0 in Extended Capability Register). • 0: Page Requests to report recoverable address translation faults are blocked. • 1: Page Requests to report recoverable address translation faults are allowed.
3	PASIDE: PASID Enable	This field is treated as Reserved(0) for implementations not supporting PASID (PASID=0 in Extended Capability Register). • 0: Requests-with-PASID received and processed through this scalable-mode context-entry are blocked. • 1: Requests-with-PASID received and processed through this scalable-mode context-entry are processed per the programming of the scalable-mode PASID structures referenced via PASIDDIRPTR. Programming of this field is not applicable for processing of Requests-without-PASID.

Bits	Field	Description
2	DTE: Device-TLB Enable	This field is treated as Reserved(0) for implementations not supporting Device-TLBs (DT=0 in Extended Capability Register). • 0: Translation Requests (with or without PASID) and Translated Requests received and processed through this scalable-mode context-entry are blocked. • 1: Translation Requests (with or without PASID) received and processed through this scalable-mode context-entry are processed per the programming of the scalable-mode PASID structures referenced via PASIDDIRPTR. Translated Requests bypass address translation.
1	FPD: Fault Processing Disable	Enables or disables recording/reporting of qualified non-recoverable faults. • 0: Qualified non-recoverable faults are recorded/reported for requests processed through this scalable-mode context-entry. • 1: Qualified non-recoverable faults are not recorded/reported for requests processed through this scalable-mode context-entry. This field is evaluated by hardware irrespective of the setting of the present (P) field.
0	P: Present	• 0: Indicates the scalable-mode context-entry is not present. All other fields except Fault Processing Disable (FPD) field are ignored by hardware. • 1: Indicates the scalable-mode context-entry is present.

9.5 Scalable-Mode PASID Directory Entry

The following figure and table describe the scalable-mode PASID-directory entry used in scalable-mode to translate requests-with-PASID and requests-without-PASID (using the implied PASID value programmed in RID_PASID field in the scalable-mode context-entry).

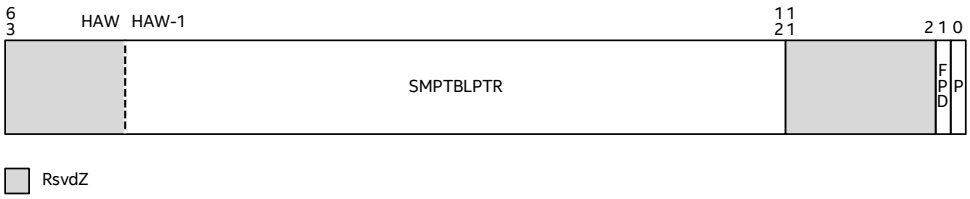


Figure 9-5. Scalable-Mode PASID Directory Entry Format

Bits	Field	Description
63:12	SMPTBLPTR: Scalable-Mode PASID Table Pointer	This field points to the base of a 4KB sized scalable-mode PASID-Table. Hardware treats bits 63:HAW in this field as Reserved(0), where HAW is the host address width of the platform. This field is always treated as Host Physical Address (HPA). Section 9.6 describes format of entries in scalable-mode PASID Table Entry.
11: 2	R: Reserved	Reserved (0).
1	FPD: Fault Processing Disable	Enables or disables recording/reporting of qualified non-recoverable faults. If this scalable-mode PASID directory entry is referenced through a scalable-mode context entry with the FPD field value set to 1, then this field has no effect. • 0: Qualified non-recoverable faults are recorded/reported for requests processed through this scalable-mode PASID-directory entry. • 1: Qualified non-recoverable faults are not recorded/reported for requests processed through this scalable-mode PASID-directory entry. This field is evaluated by hardware irrespective of the setting of the present (P) field.
0	P: Present	• 0: Indicates the scalable-mode PASID-directory entry is not present. All other fields except Fault Processing Disable (FPD) field are ignored by hardware. • 1: Indicates the scalable-mode PASID-directory entry is present.

9.6 Scalable-Mode PASID Table Entry

The following figure and table describe scalable-mode PASID-Table entry used in scalable-mode to translate requests-with-PASID and requests-without-PASID (using the implied PASID value programmed in RID_PASID field in the scalable-mode context-entry).

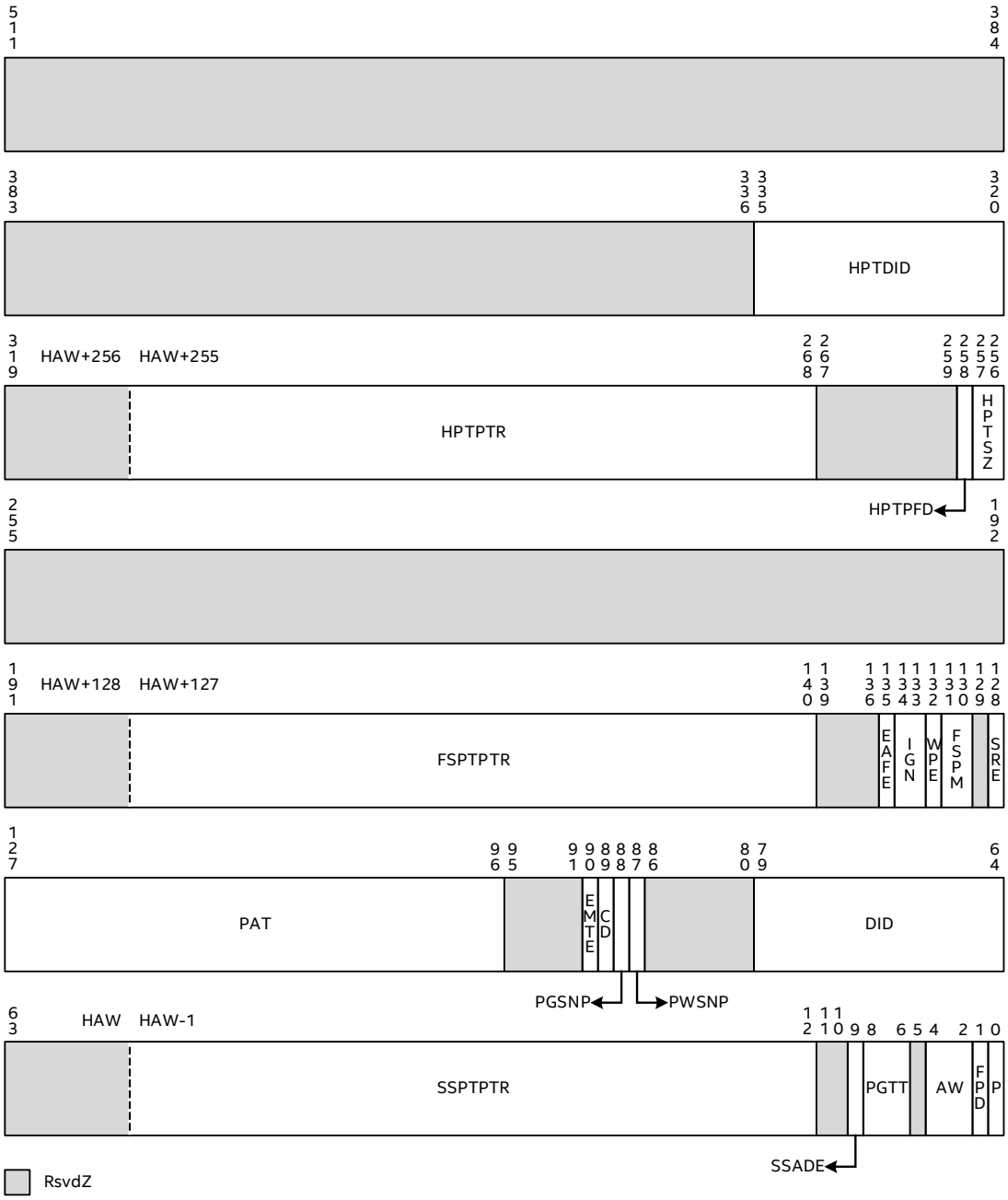


Figure 9-6. Scalable-Mode PASID Table Entry Format

<table>
<tr><th>Bits</th><th>Field</th><th>Description</th></tr>
<tr><td>511:336</td><td>R: Reserved</td><td>Reserved. Must be 0.</td></tr>
<tr><td>335:320</td><td>HPTDID: HPT Domain-ID</td><td>This field is treated as Reserved(0) for implementations not supporting Host Permission Tables (HPTS=0 in Extended Capability Register)

Identifier for the HPT domain to which this scalable-mode PASID Table Entry maps. Hardware uses the HPT domain identifier to tag its internal caches. Refer to Section 6.2.1 for cache tagging details.

The Capability Register reports the domain-id width supported by hardware. For implementations supporting less than 16-bit domain-ids, unused bits of this field are treated as reserved by hardware. For example, for an implementation supporting 8-bit domain-ids, bits 335:328 of this field are treated as reserved.

Scalable-mode PASID table entries programmed with the same HPT domain identifier must always reference the same HPT (HPT Root Pointer field). See Section 6.2.6.2 for details on programming this field.

When not treated as Reserved(0), hardware ignores this field when the HPT Enable field is Clear in the corresponding scalable-mode context entry.</td></tr>
<tr><td>319: HAW+256</td><td>R: Reserved</td><td>Reserved. Must be 0.</td></tr>
<tr><td>HAW+255: 268</td><td>HPTPTR: HPT Root Pointer</td><td>This field is treated as Reserved(0) for implementations not supporting Host Permission Tables (HPTS=0 in Extended Capability Register)

Pointer to root of the Host Permission Table (base of HPTL4 table). Refer to Section 9.10 for HPT paging structure details.

When not treated as Reserved(0), hardware ignores this field when the HPT Enable field is Clear in the corresponding scalable-mode context entry.</td></tr>
<tr><td>267:259</td><td>R: Reserved</td><td>Reserved. Must be 0.</td></tr>
<tr><td>258</td><td>HPTPFD: HPT Prefetch Disable</td><td>This field is treated as Reserved(0) for implementations not supporting Host Permission Tables (HPTS=0 in Extended Capability Register)
• 0: Hardware may prefetch an HPT cache entry corresponding to the translated address after successfully servicing a translation request.
• 1: Hardware will not prefetch an HPT cache entry corresponding to the translated address after successfully servicing a translation request.

When not treated as Reserved(0), hardware ignores this field when the HPT Enable field is Clear in the corresponding scalable-mode context entry.</td></tr>
<tr><td>257:256</td><td>HPTSZ: HPT Size</td><td>This field is treated as Reserved(0) for implementations not supporting Host Permission Tables (HPTS=0 in Extended Capability Register)

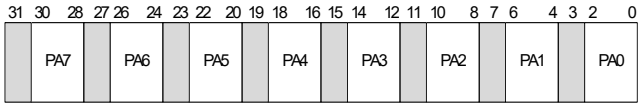
Indicates the size of the HPTL4 table, which controls the maximum physical address that can be used in a translated request.

<table><tr><th>HPT Size</th><th>HPTL4 Table Size</th><th>Maximum supported translated address</th></tr><tr><td>0</td><td>4KB</td><td>2^{49}-1</td></tr><tr><td>1</td><td>8KB</td><td>2^{50}-1</td></tr><tr><td>2</td><td>16KB</td><td>2^{51}-1</td></tr><tr><td>3</td><td>32KB</td><td>2^{52}-1</td></tr></table>
When not treated as Reserved(0), hardware ignores this field when the HPT Enable field is Clear in the corresponding scalable-mode context entry.</td></tr>
</table>

Bits	Field	Description
255:192	R: Reserved	Reserved. Must be 0.
191: HAW+128	R: Reserved	Reserved. Must be 0.
HAW+127: 140	FSPTPTR: First Stage Page Translation Pointer	This field is treated as Reserved(0) for implementations not supporting First-stage Translation (FSTS=0 in Extended Capability Register). Pointer to root of first-stage paging structures (base of FS-PML4 table if FSPM=00b; base of FS-PML5 table if FSPM=01b). Refer to Section 9.7 for first-stage paging structure details. When not treated as Reserved(0), hardware ignores this field for second-stage-only (PGTT=010b) and pass-through (PGTT=100b) translations. The field is interpreted as a Guest Physical Address (GPA), when nested translation are enabled (PGTT=011b)
139:136	R: Reserved	Reserved. Must be 0.
135	EAFE: Extended Accessed Flag Enable	This field is treated as Reserved(0) for implementations not supporting Extended Accessed flag (EAFS=0 in the Extended Capability Register). If Set, Extended-Accessed (EA) flag is atomically Set in first-stage paging-entries referenced by remapping hardware through this scalable-mode PASID Table Entry. When not treated as Reserved(0), hardware ignores this field for second-stage-only (PGTT=010b) and pass-through (PGTT=100b) translations.
134:133	IGN: Ignored	Hardware must ignore programming of this field to ensure backward compatibility with older software.
132	WPE: Write Protect Enable	This field is treated as Reserved(0) for implementations not supporting First-stage Translation (FSTS=0 in Extended Capability Register). Hardware ignores this field for requests with user-level privilege (request-with-PASID with PR=0 or request-without-PASID with RID_PRIV=0) • 0: Allows supervisor-level accesses to write into read-only pages. • 1: Inhibits supervisor-level accesses from writing into read-only pages. When not treated as Reserved(0), hardware ignores this field for second-stage-only (PGTT=010b) and pass-through (PGTT=100b) translations.
131:130	FSPM: First Stage Paging Mode	This field is treated as Reserved(0) for implementations not supporting First-stage Translations (FSTS=0 in Extended Capability Register). This field specifies the paging mode for first-stage translation. • 00: 4-level paging (FSPTPTR is base of FS-PML4) • 01: 5-level paging (FSPTPTR is base of FS-PML5) • 10-11: Reserved For implementations reporting 5-level Paging as not supported (FS5LP = 0) in Capability Register, this field must be programmed as 00b.
129	R: Reserved	Reserved. Must be 0.
128	SRE: Supervisor Request Enable	This field is treated as Reserved(0) for implementations not supporting Supervisor Request Support (SRS=0 in the Extended Capability Register). If Clear, requests with PASID requesting supervisor privilege level are blocked and treated as DMA remapping faults; requests without PASID are blocked if RID_PRIV is 1.

<table>
<tr><th>Bits</th><th>Field</th><th>Description</th></tr>
<tr><td>127:96</td><td>PAT: Page Attribute Table</td><td>
This field is treated as Reserved(0) for implementations not supporting Memory Type (MTS=0 in Extended Capability Register).

This field is used to compute memory-type for requests from devices that operate in processor coherency domain. Refer to Section 3.11, for hardware handling of memory-type.

The format of this field is specified below.


Following encodings are specified for the value programmed in sub-fields PA0 through PA7.
<table>
<tr><th>Encoding</th><th>Mnemonic</th></tr>
<tr><td>00h</td><td>Uncacheable (UC)</td></tr>
<tr><td>01h</td><td>Write Combining (WC)</td></tr>
<tr><td>02h</td><td>Reserved</td></tr>
<tr><td>03h</td><td>Reserved</td></tr>
<tr><td>04h</td><td>Write Through (WT)</td></tr>
<tr><td>05h</td><td>Write Protected (WP)</td></tr>
<tr><td>06h</td><td>Write Back (WB)</td></tr>
<tr><td>07h</td><td>Uncached (UC-)</td></tr>
<tr><td>08h-0Fh</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>95:91</td><td>R: Reserved</td><td>Reserved. Must be 0.</td></tr>
<tr><td>90</td><td>EMTE: Extended Memory Type Enable</td><td>
This field is treated as Reserved(0) for implementations not supporting Memory Type (MTS=0 in Extended Capability Register).

• 0: Extended Memory Type (EMT) field in second-stage leaf paging-entries referenced through SSPTPTR are ignored.
• 1: Extended Memory Type (EMT) field in second-stage leaf paging-entries referenced through SSPTPTR are used for memory type determination for requests from devices operating in processor coherency domain.
Refer to Section 3.11.4 for hardware handling of memory-type.
When not treated as Reserved(0), hardware ignores this field for first-stage-only (PGTT=001b) and pass-through (PGTT=100b) translations.
</td></tr>
<tr><td>89</td><td>CD: Cache Disable</td><td>
This field is treated as Reserved(0) for implementations not supporting Memory Type (MTS=0 in Extended Capability Register).

This field is only applicable for requests from devices that operate in processor coherency domain.
• 0: Normal Cache Mode.
  • Read hits access cache; Read misses may cause replacement.
  • Write hits update cache; Write misses cause cache line fill.
  • Writes to shared lines and write misses update system memory.
  • Write hits can change shared lines to modified under control of MTRR registers or EMT field, with associated read invalidation cycle.
• 1: Cache is disabled.
  • Effective memory-type forced to Un-cacheable (UC), irrespective of programming of MTRR registers and PAT/EMT fields.
  • Cache continues to respond to snoop traffic.
</td></tr>
</table>

Bits	Field	Description
88	PGSNP: Page Snoop	This field is treated as Reserved(0) for implementations not supporting Snoop Control (SC=0 in the Extended Capability Register). • 0: Requests snoop processor caches based on other attributes in the request or other fields in paging structure entries used to translate the request. • 1: Requests snoop processor caches irrespective of, other attributes in the request or other fields in paging structure entries used to translate the request.
87	PWSNP: Page-walk Snoop	This field is treated as Reserved(0) for implementations not supporting Scalable-mode Page-walk Coherency (SMPWC=0 in Extended Capability Register). • 0: Hardware accesses to paging structures (such as scalable-mode FS/SS/HPT tables) do not snoop processor caches. • 1: Hardware accesses to paging structures (such as scalable-mode FS/SS/HPT tables) snoop processor caches.
86:80	R: Reserved	Reserved. Must be 0.
79:64	DID: Domain Identifier	Identifier for the domain to which this scalable-mode PASID Table Entry maps. Hardware uses the domain identifier to tag the internal caches. Refer to Section 6.2.1 for cache tagging details. The Capability Register reports the domain-id width supported by hardware. For implementations supporting less than 16-bit domain-ids, unused bits of this field are treated as reserved by hardware. For example, for an implementation supporting 8-bit domain-ids, bits 79:72 of this field are treated as reserved. Scalable-mode PASID table entries programmed with the same domain identifier must always reference the same second-stage paging structures (SSPTPTR field). See Section for details on programming this field. When Caching Mode (CM) field in Capability Register is reported as Set, the domain-id value of zero is architecturally reserved. Software must not use domain-id value of zero when CM is Set.
63:HAW	R: Reserved	Reserved. Must be 0
HAW-1:12	SSPTPTR: Second Stage Page Translation Pointer	This field is treated as Reserved(0) for implementations not supporting Second-stage Translation (SSTS=0 in Extended Capability Register). This field points to the base of the second-stage page-tables (described in Section 9.8). When not treated as Reserved(0), hardware ignores this field for first-stage-only (PGTT=001b) and pass-through (PGTT=100b) translations.
11:10	R: Reserved	Reserved. Must be 0.
9	SSADE: Second Stage Access/Dirty bit Enable	This field is treated as Reserved(0) for implementations not supporting Second-stage Translation (SSTS=0 in the Extended Capability Register) or not supporting Second Stage Accessed/Dirty bits (SSADS=0 in the Extended Capability Register). • 0: Disable Accessed/Dirty Flags in second-stage paging entries. • 1: Enable Accessed/Dirty Flags in second-stage paging entries. Refer to Section 3.7.2 for details on Accessed/Dirty Flag support in second-stage paging entries. When not treated as Reserved(0), hardware ignores this field for first-stage-only (PGTT=001b) and pass-through (PGTT=100b) translations.

Bits	Field	Description

8:6	PGTT: PASID Granular Translation Type	• 000b: Reserved
		• 001b: First-stage Translation only
		  • This value is treated as reserved for implementations not supporting First-stage Translation (FSTS=0 in the Extended Capability Register).
		  • Untranslated/Translation requests (with or without PASID) processed through this scalable-mode PASID table entry are translated using the first-stage paging structures referenced through the FSPTPTR field.
		• 010b: Second-stage Translation only
		  • This value is treated as reserved for implementations not supporting Second-stage Translation (SSTS=0 in the Extended Capability Register).
		  • Untranslated/Translation requests (with or without PASID) processed through this scalable-mode PASID table entry are translated using the second-stage paging structures referenced through the SSPTPTR field.
		• 011b: Nested Translation
		  • This value is treated as reserved for implementations not supporting Nested Translation (NEST=0 in the Extended Capability Register).
		  • Untranslated/Translation requests (with or without PASID) processed through this scalable-mode PASID table entry are translated using nested first-stage paging structures (referenced through the FSPTPTR field) and second-stage paging structures (reference through the SSPTPTR field).
		• 100b: Pass-through
		  • This value is treated as reserved for implementations not supporting Pass Through (PT=0 in the Extended Capability Register).
		  • Untranslated/Translation Requests (with or without PASID) referencing this scalable-mode PASID Table Entry bypass address translation and are processed as pass-through.
		• 101-111b: Reserved

5	R: Reserved	Reserved. Must be 0.
4:2	AW: Address Width	This field is treated as Reserved(0) for implementations not supporting Second-stage Translation (SSTS=0 in the Extended Capability Register). This field indicates the adjusted guest-address-width (AGAW) to be used by hardware for second-stage translation through paging structures referenced through the SSPTPTR field. • The following encodings are defined for this field: • 001b: 39-bit AGAW (3-level page table) • 010b: 48-bit AGAW (4-level page table) • 011b: 57-bit AGAW (5-level page table) • 000b,100b-111b: Reserved When not treated as Reserved(0), hardware ignores this field for first-stage-only (PGTT=001b) and pass-through (PGTT=100b) translations. The value specified in this field must match an AGAW value supported by hardware (as reported in the SAGAW field in the Capability Register). Requests to addresses above 2^X-1 (where X is the AGAW value indicated by this field) that are subject to second-stage translation are blocked and treated as translation faults.
1	FPD: Fault Processing Disable	Enables or disables recording/reporting of qualified non-recoverable faults. If this scalable-mode PASID entry is referenced through a scalable-mode context entry or a scalable-mode PASID directory entry with the FPD field value set to 1, then this field has no effect. • 0: Qualified non-recoverable faults are recorded/reported for requests (with or without PASID) processed through this scalable-mode PASID table entry. • 1: Qualified non-recoverable faults are not recorded/reported for requests (with or without PASID) processed through this scalable-mode PASID table entry. This field is evaluated by hardware irrespective of the setting of the present (P) field.
0	P: Present	• 0: Indicates the scalable-mode PASID table entry is not present. All other fields except Fault Processing Disable (FPD) field are ignored by hardware. • 1: Indicates the scalable-mode PASID table entry is present.

9.7 First-Stage Paging Entries

The following figure and tables describe the first-stage paging structures. First-stage paging entries are bitwise compatible with Intel® 64 processor's 64-bit paging entry format. Refer to the tables below for additional details not represented in the figure. Notable differences are as follows:

- **Extended-Accessed flag**: First-stage translation supports an Extended-Accessed (EA) flag in the paging entries. The EA flag works similar to the Accessed (A) flag in the paging entries. EA flags can be enabled through the Extended-Accessed-Flag-Enable (EAFE) field in the scalable-mode PASID-table entry (see Section 9.6). When enabled, bit 10 in the first-stage paging entries are treated by hardware as the EA flag, and are atomically set whenever a paging-entry is accessed by hardware. For software usages where the first-stage paging structures are shared across heterogeneous agents (e.g., CPUs and GPUs), EA flag may be used by software to identify pages accessed by non-CPU agent(s) (as opposed to the A flag which indicates access by any agent sharing the paging structures). The EA field is ignored by hardware if EAFE is Clear in the scalable-mode PASID-table entry referencing the first-stage paging entries
- **Memory Type Fields (PAT,PCD,PWT):** For devices operating in the processor coherency domain, these fields indirectly determine the memory type used to access the pages referenced by such entries. For devices not operating in the processor coherency domain or when Memory Type Support (MTS) is reported as Clear in the Extended Capability Register, these fields are ignored. Refer to Section 3.11.1 for memory-type handling.

63–62	61–60	59–58	57–56	55–54	53–52	51	HAW	HAW-1	32–31	30–29	28–27	26–25	24–23	22–21	20–19	18–17	16–15	14–13	12–11	10	9	8	7	6	5	4	3	2	1	0	
Ignored							Rsvd.		Address of PML4 table											Ign	EA	Ign	Rsvd	Ign	A	Ign	Ign	U/S	R/W	P (1)	PML5E
Ignored							Rsvd.		Address of page-directory-pointer table											Ign	EA	Ign	Rsvd	Ign	A	Ign	Ign	U/S	R/W	P (1)	PML4E
Ignored							Rsvd.	Address of 1GB page frame										Reserved		PAT	Ign	EA	Ign	G	1	D	A	PCD	PWT	U/S	R/W	P (1)	PDPE: 1GB page
Ignored							Rsvd.	Address of page directory												Ign	EA	Ign	0	Ign	A	Ign	Ign	U/S	R/W	P (1)	PDPE: page directory
Ignored							Rsvd.	Address of 2MB page frame										Reserved		PAT	Ign	EA	Ign	G	1	D	A	PCD	PWT	U/S	R/W	P (1)	PDE: 2MB page
Ignored							Rsvd.	Address of page table												Ign	EA	Ign	0	Ign	A	Ign	Ign	U/S	R/W	P (1)	PDE: page table
Ignored							Rsvd.	Address of 4KB page frame												Ign	EA	Ign	G	PAT	D	A	PCD	PWT	U/S	R/W	P (1)	PTE: 4KB page

Figure 9-7. Format for First-Stage Paging Entries

Table 34. Format of PML5E that References a PML4 Table

Bits	Field	Description
63:52	IGN: Ignored	Ignored by hardware.
51:HAW	R: Reserved	Reserved (0).
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned PML4 table referenced by this entry. This field is treated as Guest Physical Address (GPA) when PGTT field in the relevant scalable-mode PASID-table entry is programmed as nested (011b).
11	IGN: Ignored	Ignored by hardware.
10	EA: Extended Accessed	If EAFE=1 in the relevant scalable-mode PASID-table entry, this bit indicates whether this entry has been used for address translation. Refer to Section 3.6.2 for extended-accessed bit handling. If EAFE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored.
9:8	IGN: Ignored	Ignored by hardware.
7	R: Reserved	Reserved (0).
6	IGN: Ignored	Ignored by hardware.
5	A: Accessed	Indicates whether this entry has been used for address translation. Refer to Section 3.6.2 for accessed bit handling.
4	IGN: Ignored	Ignored by hardware.
3	IGN: Ignored	Ignored by hardware.
2	U/S: User/Supervisor	If 0, requests with user-level privilege are not allowed to the 256-TByte region controlled by this entry. Refer to Section 3.6.1 for access rights.
1	R/W: Read/Write	If 0, write permission not granted for requests with user-level privilege (and requests with supervisor-level privilege, if WPE=1 in the relevant scalable-mode PASID-table entry) to the 256-TByte region controlled by this entry. Refer to Section 3.6.1 for access rights.
0	P: Present	Must be 1 to reference a PML4 table.

Table 35. Format of PML4E that References a Page-Directory-Pointer Table

Bits	Field	Description
63:52	IGN: Ignored	Ignored by hardware.
51:HAW	R: Reserved	Reserved (0).
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page-directory-pointer table referenced by this entry. This field is treated as Guest Physical Address (GPA) when PGTT field in the relevant scalable-mode PASID-table entry is programmed as nested (011b).
11	IGN: Ignored	Ignored by hardware.
10	EA: Extended Accessed	If EAFE=1 in the relevant scalable-mode PASID-table entry, this bit indicates whether this entry has been used for address translation. Refer to Section 3.6.2 for extended-accessed bit handling. If EAFE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored.
9:8	IGN: Ignored	Ignored by hardware.
7	R: Reserved	Reserved (0).
6	IGN: Ignored	Ignored by hardware.
5	A: Accessed	Indicates whether this entry has been used for address translation. Refer to Section 3.6.2 for accessed bit handling.
4	IGN: Ignored	Ignored by hardware.
3	IGN: Ignored	Ignored by hardware.
2	U/S: User/Supervisor	If 0, requests with user-level privilege are not allowed to the 512-GByte region controlled by this entry. Refer to Section 3.6.1 for access rights.
1	R/W: Read/Write	If 0, write permission not granted for requests with user-level privilege (and requests with supervisor-level privilege, if WPE=1 in the relevant scalable-mode PASID-table entry) to the 512-GByte region controlled by this entry. Refer to Section 3.6.1 for access rights.
0	P: Present	Must be 1 to reference a page-directory-pointer table.

Table 36. Format of PDPE that Maps a 1-GByte Page

Bits	Field	Description
63:52	IGN: Ignored	Ignored by hardware.
51:HAW	R: Reserved	Reserved (0).
(HAW-1):30	ADDR: Address	Physical address of 1-GByte page referenced by this entry. This field is treated as Guest Physical Address (GPA) when PGTT field in the relevant scalable-mode PASID-table entry is programmed as nested (011b).
29:13	R: Reserved	Reserved (0).
12	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the 1-GByte page referenced by this entry. For devices not operating in the processor coherency domain or when Memory Type Support (MTS) is reported as Clear in the Extended Capability Register, this field is ignored. Refer to Section 3.11.1 for memory-type handling.
11	IGN: Ignored	Ignored by hardware.
10	EA: Extended Accessed	If EAFE=1 in the relevant scalable-mode PASID-table entry, this bit indicates whether this entry has been used for address translation. Refer to Section 3.6.2 for extended-accessed bit handling. If EAFE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored.
9	IGN: Ignored	Ignored by hardware.
8	G: Global	Hardware ignores this bit and functions as if G bit was Clear.
7	PS: Page Size	Must be 1 (otherwise this entry references a page directory. Refer to Table 37). This field is treated as Reserved (0) for implementations not supporting First Stage 1-GByte Pages (FS1GP=0 in the Capability Register).
6	D: Dirty	If 1, indicates one or more requests seeking write permission was successfully translated to the 1-GByte page referenced by this entry.
5	A: Accessed	Indicates whether this entry has been used for address translation. Refer to Section 3.6.2 for accessed bit handling.
4	PCD: Page-level Cache Disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the 1-GByte page referenced by this entry. For devices not operating in the processor coherency domain or when Memory Type Support (MTS) is reported as Clear in the Extended Capability Register, this field is ignored. Refer to Section 3.11.1 for memory-type handling.
3	PWT: Page-level Write Through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the 1-GByte page referenced by this entry. For devices not operating in the processor coherency domain or when Memory Type Support (MTS) is reported as Clear in the Extended Capability Register, this field is ignored. Refer to Section 3.11.1 for memory-type handling.
2	U/S: User/Supervisor	If 0, requests with user-level privilege are not allowed to the 1-GByte page referenced by this entry. Refer to Section 3.11.4 for access rights.
1	R/W: Read/Write	If 0, write permission not granted for requests with user-level privilege (and requests with supervisor-level privilege, if WPE=1 in the relevant scalable-mode PASID-table entry) to the 1-GByte page referenced by this entry. Refer to Section 3.6.1 for access rights.
0	P: Present	Must be 1 to map a 1-GByte page.

Table 37. Format of PDPE that References a Page-Directory Table

Bits	Field	Description
63:52	IGN: Ignored	Ignored by hardware.
51:HAW	R: Reserved	Reserved (0).
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page directory referenced by this entry. This field is treated as Guest Physical Address (GPA) when PGTT field in the relevant scalable-mode PASID-table entry is programmed as nested (011b).
11	IGN: Ignored	Ignored by hardware.
10	EA: Extended Accessed	If EAFE=1 in the relevant scalable-mode PASID-table entry, this bit indicates whether this entry has been used for address translation. Refer to Section 3.6.2 for extended-accessed bit handling. If EAFE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored.
9:8	IGN: Ignored	Ignored by hardware.
7	PS: Page Size	Must be 0 (otherwise this entry maps to a 1-GByte page. Refer to Table 36).
6	IGN: Ignored	Ignored by hardware.
5	A: Accessed	Indicates whether this entry has been used for address translation. Refer to Section 3.6.2 for accessed bit handling.
4	IGN: Ignored	Ignored by hardware.
3	IGN: Ignored	Ignored by hardware.
2	U/S: User/Supervisor	If 0, requests with user-level privilege are not allowed to the 1-GByte region controlled by this entry. Refer to Section 3.6.1 for access rights.
1	R/W: Read/Write	If 0, write permission not granted for requests with user-level privilege (and requests with supervisor-level privilege, if WPE=1 in the relevant scalable-mode PASID-table entry) to the 1-GByte region controlled by this entry. Refer to Section 3.6.1 for access rights. for access rights.
0	P: Present	Must be 1 to reference a page directory.

Table 38. Format of PDE that Maps a 2-MByte Page

Bits	Field	Description
63:52	IGN: Ignored	Ignored by hardware.
51:HAW	R: Reserved	Reserved (0).
(HAW-1):21	ADDR: Address	Physical address of 2-MByte page referenced by this entry. This field is treated as Guest Physical Address (GPA) when PGTT field in scalable-mode PASID-table entry is programmed as nested (011b).
20:13	R: Reserved	Reserved (0).
12	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the 2-MByte page referenced by this entry. For devices not operating in the processor coherency domain or when Memory Type Support (MTS) is reported as Clear in the Extended Capability Register, this field is ignored. Refer to Section 3.11.1 for access rights. for memory-type handling.
11	IGN: Ignored	Ignored by hardware.
10	EA: Extended Accessed	If EAFE=1 in the relevant scalable-mode PASID-table entry, this bit indicates whether this entry has been used for address translation. Refer to Section 3.6.2 for access rights. for extended-accessed bit handling. If EAFE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored.
9	IGN: Ignored	Ignored by hardware.
8	G: Global	Hardware ignores this bit and functions as if G bit was Clear.
7	PS: Page Size	Must be 1 (otherwise this entry references a page table. Refer to Table 39).
6	D: Dirty	If 1, indicates one or more requests seeking write permission was successfully translated to the 2-MByte page referenced by this entry. Refer to Section 3.6.2 for dirty bit handling.
5	A: Accessed	Indicates whether this entry has been used for address translation. Refer to Section 3.6.2 for accessed bit handling.
4	PCD: Page-level Cache Disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the 2-MByte page referenced by this entry. For devices not operating in the processor coherency domain or when Memory Type Support (MTS) is reported as Clear in the Extended Capability Register, this field is ignored. Refer to Section 3.11.1 for memory-type handling.
3	PWT: Page-level Write Through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the 2-MByte page referenced by this entry. For devices not operating in the processor coherency domain or when Memory Type Support (MTS) is reported as Clear in the Extended Capability Register, this field is ignored. Refer to Section 3.11.1 for memory-type handling.
2	U/S: User/Supervisor	If 0, requests with user-level privilege are not allowed to the 2-MByte page referenced by this entry. Refer to Section 3.6.1 for access rights.
1	R/W: Read/Write	If 0, write permission not granted for requests with user-level privilege (and requests with supervisor-level privilege, if WPE=1 in the relevant scalable-mode PASID-table entry) to the 2-MByte page referenced by this entry. Refer to Section 3.6.1 for access rights.
0	P: Present	Must be 1 to map a 2-MByte page.

Table 39. Format of PDE that References a Page Table

Bits	Field	Description
63:52	IGN: Ignored	Ignored by hardware.
51:HAW	R: Reserved	Reserved (0).
(HAW-1):12	ADDR: Address	Physical address of 4-KByte aligned page table referenced by this entry. This field is treated as Guest Physical Address (GPA) when PGTT field in the relevant scalable-mode PASID-table entry is programmed as nested (011b).
11	IGN: Ignored	Ignored by hardware.
10	EA: Extended Accessed	If EAFE=1 in the relevant scalable-mode PASID-table entry, this bit indicates whether this entry has been used for address translation. Refer to Section 3.6.2 for extended-accessed bit handling. If EAFE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored.
9:8	IGN: Ignored	Ignored by hardware.
7	PS: Page Size	Must be 0 (otherwise this entry maps to a 2-MByte page. Refer to Table 38).
6	IGN: Ignored	Ignored by hardware.
5	A: Accessed	Indicates whether this entry has been used for address translation. Refer to Section 3.6.2 for accessed bit handling.
4	IGN: Ignored	Ignored by hardware.
3	IGN: Ignored	Ignored by hardware.
2	U/S: User/Supervisor	If 0, requests with user-level privilege are not allowed to the 2-MByte region controlled by this entry. Refer to Section 3.6.1 for access rights.
1	R/W: Read/Write	If 0, write permission not granted for requests with user-level privilege (and requests with supervisor-level privilege, if WPE=1 in the relevant scalable-mode PASID-table entry) to the 2-MByte region controlled by this entry. Refer to Section 3.6.1 for access rights.
0	P: Present	Must be 1 to reference a page table.

Table 40. Format of PTE that Maps a 4-KByte Page

Bits	Field	Description
63:52	IGN: Ignored	Ignored by hardware.
51:HAW	R: Reserved	Reserved (0).
(HAW-1):12	ADDR: Address	Physical address of 4-KByte page referenced by this entry. This field is treated as Guest Physical Address (GPA) when PGTT field in the relevant scalable-mode PASID-table entry is programmed as nested (011b).
11	IGN: Ignored	Ignored by hardware.
10	EA: Extended Accessed	If EAFE=1 in the relevant scalable-mode PASID-table entry, this bit indicates whether this entry has been used for address translation. Refer to Section 3.6.2 for extended-accessed bit handling. If EAFE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored.
9	IGN: Ignored	Ignored by hardware.
8	G: Global	Hardware ignores this bit and functions as if G bit was Clear.
7	PAT: Page Attribute	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the 4-KByte page referenced by this entry. For devices not operating in the processor coherency domain or when Memory Type Support (MTS) is reported as Clear in the Extended Capability Register, this field is ignored. Refer to Section 3.11.1 for memory-type handling.
6	D: Dirty	If 1, indicates one or more requests seeking write permission was successfully translated to the 4-KByte page referenced by this entry. Refer to Section 3.6.2 for dirty bit handling
5	A: Accessed	Indicates whether this entry has been used for address translation. Refer to Section 3.6.2 for accessed bit handling.
4	PCD: Page-level Cache Disable	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the 4-KByte page referenced by this entry. For other devices, this field is ignored. Refer to Section 3.11.1 for memory-type handling.
3	PWT: Page-level Write Through	For devices operating in the processor coherency domain, this field indirectly determines the memory type used to access the 4-KByte page referenced by this entry. For other devices, this field is ignored. Refer to Section 3.11.1 for memory-type handling.
2	U/S: User/Supervisor	If 0, requests with user-level privilege are not allowed to the 4-KByte page referenced by this entry. Refer to Section 3.6.1 for access rights.
1	R/W: Read/Write	If 0, write permission not granted for requests with user-level privilege (and requests with supervisor-level privilege, if WPE=1 in the relevant scalable-mode PASID-table entry) to the 4-KByte page referenced by this entry. Refer to Section 3.6.1 for access rights.
0	P: Present	Must be 1 to map a 4-KByte page.

9.8 Second-Stage Paging Entries

The following figure and tables describe the second-stage paging entries. Second-stage paging entries are bitwise compatible with the Intel® 64 processor's EPT paging entry format. Refer to the tables below for additional details not represented in the figure.

63	62	61	60	59	58	57	56	55	54	53	52	51	HAW	HAW-1	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Ign	IW	IR	Ignored										Rsvd.		Address of Second-stage-PML4 table																						Rsvd	Ign		A	Rsvd	Ign				W	R	SS-PML5E
Ign	IW	IR	Ignored										Rsvd.		Address of Second-stage-page-directory-pointer table																						Rsvd	Ign		A	Rsvd	Ign				W	R	SS-PML4E
Ign	IW	IR	Ignored										Rsvd.		Address of 1GB page frame										Reserved												SNP	Ign	D	A	1	IPAT[1]	EMT[1]	Ign		W	R	SS-PDPE: 1GB page[2]
Ign	IW	IR	Ignored										Rsvd.		Address of second-stage-page-directory table																						Rsvd	Ign		A	0	Ign				W	R	SS-PDPE: page directory
Ign	IW	IR	Ignored										Rsvd.		Address of 2MB page frame																Reserved						SNP	Ign	D	A	1	IPAT	EMT	Ign		W	R	SS-PDE: 2MB page[2]
Ign	IW	IR	Ignored										Rsvd.		Address of second-stage-page table																						Rsvd	Ign		A	0	Ign				W	R	SS-PDE: page table
Ign	IW	IR	Ignored										Rsvd.		Address of 4KB page frame																						SNP	Ign	D	A	Ign	IPAT	EMT	Ign		W	R	SS-PTE: 4KB page

1. EMT and IPAT fields are ignored by hardware if Memory Type Support (MTS) is reported as Clear in the Extended Capability Register or if EMTE=0 in the scalable-mode PASID-table entry referencing the second-stage paging entries.
2. 1-GByte page and 2-MByte page support is reported through Second-stage Large Page Support (SSLPS) in the Capability Register.

Figure 9-8. Format for Second-Stage Paging Entries

Table 41. Format of SS-PML5E Referencing a Second-Stage-PML4 Table

Bits	Field	Description
63	IGN: Ignored	Ignored by hardware.
62	IW: I/O Write	This field is treated as Reserved(0) when Second Stage I/O Read/Write bits are not enabled. When Second Stage I/O Read/Write bits are enabled, if this field is 0, write permission is not granted for requests to the 256-TByte region controlled by this entry.
61	IR: I/O Read	This field is ignored when Second Stage I/O Read/Write bits are not enabled. When Second Stage I/O Read/Write bits are enabled: If this field is 0, read permission is not granted for requests to the 256-TByte region controlled by this entry. For implementations reporting Zero-Length-Read (ZLR) field as Set in the Capability Register, Read permission is not applicable to zero-length requests if I/O Write permission field (IW) is 1.
60:52	IGN: Ignored	Ignored by hardware.
51:HAW	R: Reserved	Reserved (0).
(HAW-1):12	ADDR: Address	Host physical address of 4-KByte aligned second-stage-PML4 table referenced by this entry.
11	R: Reserved	Reserved (0).
10:9	IGN: Ignored	Ignored by hardware.
8	A: Accessed	Indicates whether this entry has been used for address translation. Refer to Section 3.7.2 for accessed bit handling. If SSADE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored by hardware.
7	R: Reserved	Reserved (0).
6:2	IGN: Ignored	Ignored by hardware.
1	W: Write	This field is ignored when Second Stage I/O Read/Write bits are enabled. When Second Stage I/O Read/Write bits are not enabled, if 0, write permission not granted for requests to the 256-TByte region controlled by this entry.
0	R: Read	This field is ignored when Second Stage I/O Read/Write bits are enabled. When Second Stage I/O Read/Write bits are not enabled: • If this field is 0, read permission not granted for requests to the 256-TByte region controlled by this entry. • For implementations reporting Zero-Length-Read (ZLR) field as Set in the Capability Register, read permission is not applicable to zero-length read requests if Write (W) permission field is 1.

Table 42. Format of SS-PML4E Referencing a Second-Stage-Page-Directory-Pointer Table

Bits	Field	Description
63	IGN: Ignored	Ignored by hardware.
62	IW: I/O Write	This field is treated as Reserved(0) when Second Stage I/O Read/Write bits are not enabled. When Second Stage I/O Read/Write bits are enabled, if this field is 0, write permission is not granted for requests to the 512-GByte region controlled by this entry.
61	IR: I/O Read	This field is ignored when Second Stage I/O Read/Write bits are not enabled. When Second Stage I/O Read/Write bits are enabled: If this field is 0, read permission is not granted for requests to the 512-GByte region controlled by this entry. For implementations reporting Zero-Length-Read (ZLR) field as Set in the Capability Register, Read permission is not applicable to zero-length requests if I/O Write permission field (IW) is 1.
60:52	IGN: Ignored	Ignored by hardware.
51:HAW	R: Reserved	Reserved (0).
(HAW-1):12	ADDR: Address	Host physical address of 4-KByte aligned second-stage-page-directory-pointer table referenced by this entry.
11	R: Reserved	Reserved (0).
10:9	IGN: Ignored	Ignored by hardware.
8	A: Accessed	Indicates whether this entry has been used for address translation. Refer to Section 3.7.2 for accessed bit handling. If SSADE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored by hardware.
7	R: Reserved	Reserved (0).
6:2	IGN: Ignored	Ignored by hardware.
1	W: Write	This field is ignored when Second Stage I/O Read/Write bits are enabled. When Second Stage I/O Read/Write bits are not enabled, if 0, write permission not granted for requests to the 512-GByte region controlled by this entry.
0	R: Read	This field is ignored when Second Stage I/O Read/Write bits are enabled. When Second Stage I/O Read/Write bits are not enabled: • If this field is 0, read permission not granted for requests to the 512-GByte region controlled by this entry. • For implementations reporting Zero-Length-Read (ZLR) field as Set in the Capability Register, read permission is not applicable to zero-length read requests if Write (W) permission field is 1.

Table 43. Format of SS-PDPE that Maps a 1-GByte Page

Bits	Field	Description
63	IGN: Ignored	Ignored by hardware.
62	IW: I/O Write	This field is treated as Reserved(0) when Second Stage I/O Read/Write bits are not enabled. When Second Stage I/O Read/Write bits are enabled, if this field is 0, write permission is not granted for requests to the 1-GByte region controlled by this entry.
61	IR: I/O Read	This field is ignored when Second Stage I/O Read/Write bits are not enabled. When Second Stage I/O Read/Write bits are enabled: If this field is 0, read permission is not granted for requests to the 1-GByte region controlled by this entry. For implementations reporting Zero-Length-Read (ZLR) field as Set in the Capability Register, Read permission is not applicable to zero-length requests if I/O Write permission field (IW) is 1.
60:52	IGN: Ignored	Ignored by hardware.
51:HAW	R: Reserved	Reserved (0).
(HAW-1):30	ADDR: Address	Host physical address of 1-GByte page referenced by this entry.
29:12	R: Reserved	Reserved (0).
11	SNP: Snoop	This field indicates whether accesses to this page must snoop processor caches. This field is treated as reserved(0) by hardware implementations not supporting Snoop Control (SC=0 in Extended Capability Register). Refer to Table 6 to see how this bit is used for Untranslated and to Table 10 for Translation requests.
10	IGN: Ignored	Ignored by hardware.
9	D: Dirty	If 1, indicates one or more requests seeking write permission was successfully translated to the 1-GByte page referenced by this entry. Refer to Section 3.7.2 for dirty bit handling. If SSADE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored by hardware.
8	A: Accessed	Indicates whether this entry has been used for address translation. Refer to Section 3.7.2 for accessed bit handling. If SSADE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored by hardware.
7	PS: Page Size	Must be 1 (otherwise this entry references a second-stage-page-directory. Refer to Table 44). This field is treated as Reserved (0) for implementations not supporting 1-GB page in Second Stage Large Page Support (SSLPS) field of the Capability Register.
6	IPAT: Ignore PAT	This field is ignored by hardware when the Extended Memory Type Enable (EMTE) field is Clear in the relevant scalable-mode PASID-table entry or when Translation Table Mode is set to legacy mode (TTM=00b). This field is applicable only when the scalable-mode PASID-table entry used to process the request was set up as nested-translation (PGTT=011b) and the request is from devices operating in the processor coherency domain. • 0: The Page Attribute Table (PAT) in the scalable-mode PASID-table entry is used for effective memory-type determination. • 1: The Page Attribute Table (PAT) in the scalable-mode PASID-table entry is not used for effective memory-type determination. Refer to Section 3.11.4 for memory type handling.
5:3	EMT: Extended Memory Type	This field is ignored by hardware when the Extended Memory Type Enable (EMTE) field is Clear in the relevant scalable-mode PASID-table entry or when Translation Table Mode is set to legacy mode (TTM=00b). When the EMTE field is Set, this field is used to compute effective memory-type for second-stage-only and nested translations (PGTT=010b or 011b) from devices operating in the processor coherency domain. The encodings defined for this field are 0h for Uncacheable (UC), 1h for Write Combining (WC), 4h for Write Through (WT), 5h for Write Protected (WP), and 6h for Write Back (WB). All other values are Reserved. Refer to Section 3.11.4 for memory type handling.
2	IGN: Ignored	Ignored by hardware.
1	W: Write	This field is ignored when Second Stage I/O Read/Write bits are enabled. When Second Stage I/O Read/Write bits are not enabled, if 0, write permission not granted for requests to the 1-GByte page referenced by this entry.

Table 43. Format of SS-PDPE that Maps a 1-GByte Page

0	R: Read	This field is ignored when Second Stage I/O Read/Write bits are enabled. When Second Stage I/O Read/Write bits are not enabled: • If this field is 0, read permission not granted for requests to the 1-GByte page referenced by this entry. • For implementations reporting Zero-Length-Read (ZLR) field as Set in the Capability Register, read permission is not applicable to zero-length read requests if Write (W) permission field is 1.

Table 44. Format of SS-PDPE that References a Second-Stage-Page-Directory

Bits	Field	Description
63	IGN: Ignored	Ignored by hardware.
62	IW: I/O Write	This field is treated as Reserved(0) when Second Stage I/O Read/Write bits are not enabled. When Second Stage I/O Read/Write bits are enabled, if this field is 0, write permission is not granted for requests to the 1-GByte region controlled by this entry.
61	IR: I/O Read	This field is ignored when Second Stage I/O Read/Write bits are not enabled. When Second Stage I/O Read/Write bits are enabled: If this field is 0, read permission is not granted for requests to the 1-GByte region controlled by this entry. For implementations reporting Zero-Length-Read (ZLR) field as Set in the Capability Register, Read permission is not applicable to zero-length requests if I/O Write permission field (IW) is 1.
60:52	IGN: Ignored	Ignored by hardware.
51:HAW	R: Reserved	Reserved (0).
(HAW-1):12	ADDR: Address	Host physical address of 4-KByte aligned second-stage-page-directory referenced by this entry.
11	R: Reserved	Reserved (0).
10:9	IGN: Ignored	Ignored by hardware.
8	A: Accessed	Indicates whether this entry has been used for address translation. Refer to Section 3.7.2 for accessed bit handling. If SSADE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored by hardware.
7	PS: Page Size	Must be 0 (Otherwise this entry references a 1-GByte page. Refer to Table 43).
6:2	IGN: Ignored	Ignored by hardware.
1	W: Write	This field is ignored when Second Stage I/O Read/Write bits are enabled. When Second Stage I/O Read/Write bits are not enabled, if 0, write permission not granted for requests to the 1-GByte region controlled by this entry.
0	R: Read	This field is ignored when Second Stage I/O Read/Write bits are enabled. When Second Stage I/O Read/Write bits are not enabled: • If this field is 0, read permission not granted for requests to the 1-GByte region controlled by this entry. • For implementations reporting Zero-Length-Read (ZLR) field as Set in the Capability Register, read permission is not applicable to zero-length read requests if Write (W) permission field is 1.

Table 45. Format of SS-PDE that Maps to a 2-MByte Page

Bits	Field	Description
63	IGN: Ignored	Ignored by hardware.
62	IW: I/O Write	This field is treated as Reserved(0) when Second Stage I/O Read/Write bits are not enabled. When Second Stage I/O Read/Write bits are enabled, if this field is 0, write permission is not granted for requests to the 2-MByte region controlled by this entry.
61	IR: I/O Read	This field is ignored when Second Stage I/O Read/Write bits are not enabled. When Second Stage I/O Read/Write bits are enabled: If this field is 0, read permission is not granted for requests to the 2-MByte region controlled by this entry. For implementations reporting Zero-Length-Read (ZLR) field as Set in the Capability Register, Read permission is not applicable to zero-length requests if I/O Write permission field (IW) is 1.
60:52	IGN: Ignored	Ignored by hardware.
51:HAW	R: Reserved	Reserved (0).
(HAW-1):21	ADDR: Address	Host physical address of 2-MByte page referenced by this entry.
20:12	R: Reserved	Reserved (0).
11	SNP: Snoop	This field indicates whether accesses to this page must snoop processor caches. This field is treated as reserved(0) by hardware implementations not supporting Snoop Control (SC=0 in Extended Capability Register). Refer to Table 6 to see how this bit is used for Untranslated and to Table 10 for Translation requests.
10	IGN: Ignored	Ignored by hardware.
9	D: Dirty	If 1, indicates one or more requests seeking write permission was successfully translated to the 2-MByte page referenced by this entry. Refer to Section 3.7.2 for dirty bit handling If SSADE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored by hardware.
8	A: Accessed	Indicates whether this entry has been used for address translation. Refer to Section 3.7.2 for accessed bit handling. If SSADE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored by hardware.
7	PS: Page Size	Must be 1 (Otherwise this entry references a second-stage-page table. Refer to Table 46). This field is treated as Reserved (0) for implementations not supporting 2-MB page in Second Stage Large Page Support (SSLPS) field of the Capability Register.
6	IPAT: Ignore PAT	This field is ignored by hardware when Extended Memory Type Enable (EMTE) field is Clear in the relevant scalable-mode PASID-table entry or when Translation Table Mode is set to legacy mode (TTM=00b). This field is applicable only when the scalable-mode PASID-table entry used to process the request was set up as nested-translation (PGTT=011b) and the request is from devices operating in the processor coherency domain. • 0: Page Attribute Table (PAT) in scalable-mode PASID-table entry is used for effective memory-type determination • 1: Page Attribute Table (PAT) in scalable-mode PASID-table entry is not used for effective memory-type determination. Refer to Section 3.11.4 for memory type handling.
5:3	EMT: Extended Memory Type	This field is ignored by hardware when Extended Memory Type Enable (EMTE) field is Clear in the relevant scalable-mode PASID-table entry or when Translation Table Mode is set to legacy mode (TTM=00b). When EMTE, field is Set, this field is used to compute effective memory-type for second-stage-only and nested translations (PGTT=010b or 011b) from devices operating in the processor coherency domain. The encodings defined for this field are 0h for Uncacheable (UC), 1h for Write Combining (WC), 4h for Write Through (WT), 5h for Write Protected (WP), and 6h for Write Back (WB). All other values are Reserved. Refer to Section 3.11.4 for memory type handling.
2	IGN: Ignored	Ignored by hardware.
1	W: Write	This field is ignored when Second Stage I/O Read/Write bits are enabled. When Second Stage I/O Read/Write bits are not enabled, if 0, write permission not granted for requests to the 2-MByte page referenced by this entry.

Table 45. Format of SS-PDE that Maps to a 2-MByte Page

0	R: Read	This field is ignored when Second Stage I/O Read/Write bits are enabled. When Second Stage I/O Read/Write bits are not enabled: • If this field is 0, read permission not granted for requests to the 2-MByte page referenced by this entry. • For implementations reporting Zero-Length-Read (ZLR) field as Set in the Capability Register, read permission is not applicable to zero-length read requests if Write (W) permission field is 1.

Table 46. Format of SS-PDE that References a Second-Stage-Page Table

Bits	Field	Description
63	IGN: Ignored	Ignored by hardware.
62	IW: I/O Write	This field is treated as Reserved(0) when Second Stage I/O Read/Write bits are not enabled. When Second Stage I/O Read/Write bits are enabled, if this field is 0, write permission is not granted for requests to the 2-MByte region controlled by this entry.
61	IR: I/O Read	This field is ignored when Second Stage I/O Read/Write bits are not enabled. When Second Stage I/O Read/Write bits are enabled: If this field is 0, read permission is not granted for requests to the 2-MByte region controlled by this entry. For implementations reporting Zero-Length-Read (ZLR) field as Set in the Capability Register, Read permission is not applicable to zero-length requests if I/O Write permission field (IW) is 1.
60:52	IGN: Ignored	Ignored by hardware.
51:HAW	R: Reserved	Reserved (0).
(HAW-1):12	ADDR: Address	Host physical address of 4-KByte aligned second-stage-page-table referenced by this entry.
11	R: Reserved	Reserved (0).
10:9	IGN: Ignored	Ignored by hardware.
8	A: Accessed	Indicates whether this entry has been used for address translation. Refer to Section 3.7.2 for accessed bit handling. If SSADE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored by hardware.
7	PS: Page Size	Must be 0 (Otherwise this entry references a 2-MByte page. Refer to Table 45).
6:2	IGN: Ignored	Ignored by hardware.
1	W: Write	This field is ignored when Second Stage I/O Read/Write bits are enabled. When Second Stage I/O Read/Write bits are not enabled, if 0, write permission not granted for requests to the 2-MByte region controlled by this entry.
0	R: Read	This field is ignored when Second Stage I/O Read/Write bits are enabled. When Second Stage I/O Read/Write bits are not enabled: • If this field is 0, read permission not granted for requests to the 2-MByte region controlled by this entry. • For implementations reporting Zero-Length-Read (ZLR) field as Set in the Capability Register, read permission is not applicable to zero-length read requests if Write (W) permission field is 1.

Table 47. Format of SS-PTE that Maps 4-KByte Page

Bits	Field	Description
63	IGN: Ignored	Ignored by hardware.
62	IW: I/O Write	This field is treated as Reserved(0) when Second Stage I/O Read/Write bits are not enabled. When Second Stage I/O Read/Write bits are enabled, if this field is 0, write permission is not granted for requests to the 4-KByte region controlled by this entry.
61	IR: I/O Read	This field is ignored when Second Stage I/O Read/Write bits are not enabled. When Second Stage I/O Read/Write bits are enabled: If this field is 0, read permission is not granted for requests to the 4-KByte region controlled by this entry. For implementations reporting Zero-Length-Read (ZLR) field as Set in the Capability Register, Read permission is not applicable to zero-length requests if I/O Write permission field (IW) is 1.
60:52	IGN: Ignored	Ignored by hardware.
51:HAW	R: Reserved	Reserved (0).
(HAW-1):12	ADDR: Address	Host physical address of 4-KByte page referenced by this entry.
11	SNP: Snoop	This field indicates whether accesses to this page must snoop processor caches. This field is treated as reserved(0) by hardware implementations not supporting Snoop Control (SC=0 in Extended Capability Register). Refer to Table 6 to see how this bit is used for Untranslated and to Table 10 for Translation requests.
10	IGN: Ignored	Ignored by hardware.

Table 47. Format of SS-PTE that Maps 4-KByte Page

9	D: Dirty	If 1, indicates one or more requests seeking write permission was successfully translated to the 4-KByte page referenced by this entry. Refer to Section 3.7.2 for dirty bit handling If SSADE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored by hardware.
8	A: Accessed	Indicates whether this entry has been used for address translation. Refer to Section 3.7.2 for accessed bit handling. If SSADE=0 in the relevant scalable-mode PASID-table entry, this bit is ignored by hardware.
7	IGN: Ignored	Ignored by hardware.
6	IPAT: Ignore PAT	This field is ignored by hardware when Extended Memory Type Enable (EMTE) field is Clear in the relevant scalable-mode PASID-table entry or when Translation Table Mode is set to legacy mode (TTM=00b). This field is applicable only when scalable-mode PASID-table entry used to process the request was setup as nested-translation (PGTT=011b) and the request is from devices operating in the processor coherency domain. • 0: Page Attribute Table (PAT) in scalable-mode PASID-table entry is used for effective memory-type determination • 1: Page Attribute Table (PAT) in scalable-mode PASID-table entry is not used for effective memory-type determination. Refer to Section 3.11.4 for memory type handling.
5:3	EMT: Extended Memory Type	This field is ignored by hardware when Extended Memory Type Enable (EMTE) field is Clear in the relevant scalable-mode PASID-table entry or when Translation Table Mode is set to legacy mode (TTM=00b). When EMTE field is Set, this field is used to compute effective memory-type for second-stage-only and nested translations (PGTT=010b or 011b) from devices operating in the processor coherency domain. The encodings defined for this field are 0h for Uncacheable (UC), 1h for Write Combining (WC), 4h for Write Through (WT), 5h for Write Protected (WP), and 6h for Write Back (WB). All other values are Reserved. Refer to Section 3.11.4 for memory type handling.
2	IGN: Ignored	Ignored by hardware.
1	W: Write	This field is ignored when Second Stage I/O Read/Write bits are enabled. When Second Stage I/O Read/Write bits are not enabled, if 0, write permission not granted for requests to the 4-KByte page referenced by this entry.
0	R: Read	This field is ignored when Second Stage I/O Read/Write bits are enabled. When Second Stage I/O Read/Write bits are not enabled: • If this field is 0, read permission not granted for requests to the 4-KByte page referenced by this entry. • For implementations reporting Zero-Length-Read (ZLR) field as Set in the Capability Register, read permission is not applicable to zero-length read requests if Write (W) permission field is 1.

9.9 Interrupt Remapping Table Entry (IRTE) for Remapped Interrupts

The following figure and table describe the interrupt remapping table entry for interrupt requests that are subject to remapping.

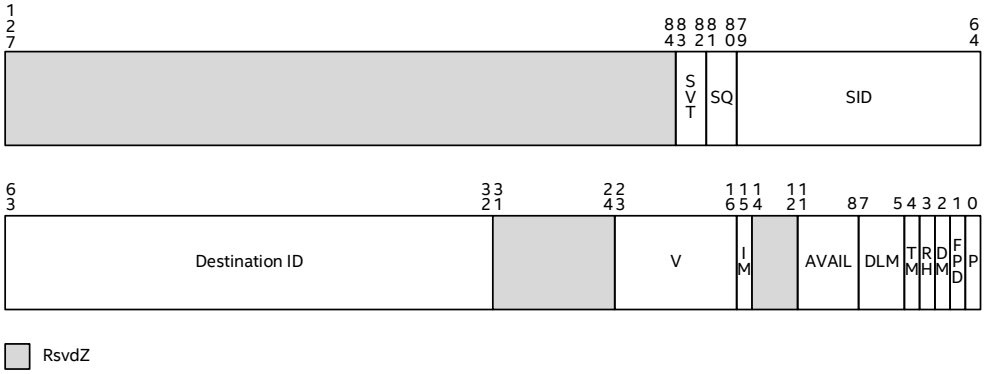


Figure 9-9. Interrupt Remap Table Entry Format for Remapped Interrupts

Bits	Field	Description
127:84	R: Reserved	Reserved. Software must program these bits to 0. This field is evaluated by hardware only when the Present (P) field is Set.
83:82	SVT: Source Validation Type	This field specifies the type of validation that must be performed by the interrupt-remapping hardware on the source-id of the interrupt requests referencing this IRTE. • 00b: No requester-id verification is required. • 01b: Verify requester-id in the interrupt request using the SID and SQ fields in the IRTE. • 10b: Verify that the most significant 8 bits of the requester-id (Bus#) in the interrupt request are equal to or within the Startbus# and EndBus# specified through the upper and lower 8 bits of the SID field respectively. This encoding may be used to verify interrupts originated behind PCI Express-to-PCI/PCI-X bridges. Refer to Section 5.1.1 for more details. • 11b: Reserved. This field is evaluated by hardware only when the Present (P) field is Set.
81:80	SQ: Source-id Qualifier	The SVT field may be used to verify the origination of interrupt requests generated by devices supporting phantom functions. If the SVT field is 01b, the following encodings are defined for the SQ field. • 00b: Verify the interrupt request by comparing all 16 bits of the SID field with the 16-bit requester-id of the interrupt request. • 01b: Verify the interrupt request by comparing the most significant 13 bits of the SID and requester-id of the interrupt request, and comparing the least significant two bits of the SID field and requester-id of the interrupt request. (i.e., ignore the third least significant field of the SID field and requester-id). • 10b: Verify the interrupt request by comparing the most significant 13 bits of the SID and requester-id of the interrupt request, and comparing the least significant bit of the SID field and requester-id of the interrupt request (i.e., ignore the second and third least significant fields of the SID field and requester-id). • 11b: Verify the interrupt request by comparing most significant 13 bits of the SID and requester-id of interrupt request. (i.e., ignore the least three significant fields of the SID field and requester-id). This field is evaluated by hardware only when the Present (P) field is Set and the SVT field is 01b.

Bits	Field	Description
79:64	SID: Source Identifier	This field specifies the originator (source) of the interrupt request that references this IRTE. The format of the SID field is determined by the programming of the SVT field. If the SVT field is: • 01b: The SID field contains the 16-bit requester-id (Bus/Dev/Func #) of the device that is allowed to originate interrupt requests referencing this IRTE. The SQ field is used by hardware to determine which bits of the SID field must be considered for the interrupt request verification. • 10b: The most significant 8 bits of the SID field contain the Startbus#, and the least significant 8 bits of the SID field contain the Endbus#. Interrupt requests that reference this IRTE must have a requester-id whose bus# (most significant 8 bits of requester-id) has a value equal to or within the Startbus# to Endbus# range. This field is evaluated by hardware only when the Present (P) field is Set and the SVT field is 01b or 10b.

63:32	DST: Destination ID	This field identifies the remapped interrupt request's target processor(s). It is evaluated by hardware only when the Present (P) field is Set.
		
		The format of this field in various Interrupt Remapping modes is as follows:
		• Intel® 64 xAPIC Mode (IRTA_REG.EIME=0):
		  • 63:48 - Reserved (0)
		  • 47:40 - APIC DestinationID[7:0]
		  • 39:32 - Reserved (0)
		• Intel® 64 x2APIC Mode (IRTA_REG.EIME=1):
		  • 63:32 - APIC DestinationID[31:0]

31:24	R: Reserved	Reserved. Software must program these bits to 0. This field is evaluated by hardware only when the Present (P) field is Set.
23:16	V: Vector	This 8-bit field contains the interrupt vector associated with the remapped interrupt request. This field is evaluated by hardware only when the Present (P) field is Set.
15	IM: IRTE Mode	A value of 0 in this field indicates that interrupt requests processed through this IRTE are remapped. A value of 1 in this field indicates that interrupt requests processed through this IRTE are posted. Refer to Section 9.10 for the IRTE format for posted interrupts.
14:12	R: Reserved	Reserved. Software must program these bits to 0. This field is evaluated by hardware only when the Present (P) field is Set.
11:8	AVAIL: Available	This field is available to software. Hardware always ignores the programming of this field.

Bits	Field	Description
7:5	DLM: Delivery Mode	This 3-bit field specifies how the remapped interrupt is handled. Delivery Modes operate only in conjunction with specified Trigger Modes (TM). Correct Trigger Modes must be guaranteed by software. Restrictions are indicated below: • *000b (Fixed Mode)* – Deliver the interrupt to all the agents indicated by the Destination ID field. The Trigger Mode for fixed delivery mode can be edge or level. • *001b (Lowest Priority)* – Deliver the interrupt to one (and only one) of the agents indicated by the Destination ID field (the algorithm to pick the target agent is component specific and could include priority based algorithm). The Trigger Mode can be edge or level. • *010b (System Management Interrupt or SMI)*: SMI is an edge triggered interrupt regardless of the setting of the Trigger Mode (TM) field. For systems that rely on SMI semantics, the vector field is ignored, but must be programmed to all zeros for future compatibility. (Support for this delivery mode is implementation specific. Platforms supporting interrupt remapping are expected to generate SMI through dedicated pin or platform-specific special messages).[1] • *100b (NMI)* – Deliver the signal to all the agents listed in the destination field. The vector information is ignored for implementations not supporting NMI-source reporting. NMI is an edge triggered interrupt regardless of the Trigger Mode (TM) setting. (Platforms supporting interrupt remapping are recommended to generate NMI through dedicated pin or platform-specific special messages[1]). For more details on NMI-source reporting, refer to Flexible Return and Event Delivery Specification version 6.0 or higher. • *101b (INIT)* – Deliver this signal to all the agents indicated by the Destination ID field. The vector information is ignored. INIT is an edge triggered interrupt regardless of the Trigger Mode (TM) setting. (Support for this delivery mode is implementation specific. Platforms supporting interrupt remapping are expected to generate INIT through dedicated pin or platform-specific special messages).[1] • *111b (ExtINT)* – Deliver the signal to the INTR signal of all agents indicated by the Destination ID field (as an interrupt that originated from an 8259A compatible interrupt controller). The vector is supplied by the INTA cycle issued by the activation of the ExtINT. ExtINT is an edge triggered interrupt regardless of the Trigger Mode (TM) setting. This field is evaluated by hardware only when the Present (P) field is Set.
4	TM: Trigger Mode	This field indicates the signal type of the interrupt that uses the IRTE. • 0: Indicates edge sensitive. • 1: Indicates level sensitive. This field is evaluated by hardware only when the Present (P) field is Set.
3	RH: Redirection Hint	This bit indicates whether the remapped interrupt request should be directed to one among N processors specified in Destination ID field, under hardware control. • 0: When RH is 0, the remapped interrupt is directed to the processor listed in the Destination ID field. • 1: When RH is 1, the remapped interrupt is directed to 1 of N processors specified in the Destination ID field. This field is evaluated by hardware only when the present (P) field is Set.
2	DM: Destination Mode	This field indicates whether the Destination ID field in an IRTE should be interpreted as logical or physical APIC ID. • 0: Physical • 1: Logical This field is evaluated by hardware only when the present (P) field is Set.
1	FPD: Fault Processing Disable	Enables or disables recording/reporting of faults caused by interrupt messages requests processed through this entry. • 0: Indicates fault recording/reporting is enabled for interrupt requests processed through this entry. • 1: Indicates fault recording/reporting is disabled for interrupt requests processed through this entry. This field is evaluated by hardware irrespective of the setting of the Present (P) field.
0	P: Present	The P field is used by software to indicate to hardware if the corresponding IRTE is present and initialized. • 0: Indicates the IRTE is not currently allocated to any interrupt sources. Block interrupt requests referencing this IRTE. • 1: Process interrupt requests referencing this IRTE per the programming of other fields in this IRTE.

1. Refer to Section 5.1.7 for hardware considerations for handling platform events.

9.10 Interrupt Remapping Table Entry (IRTE) for Posted Interrupts

The following figure and table describe the interrupt remapping table entry for interrupt requests that are processed as posted.

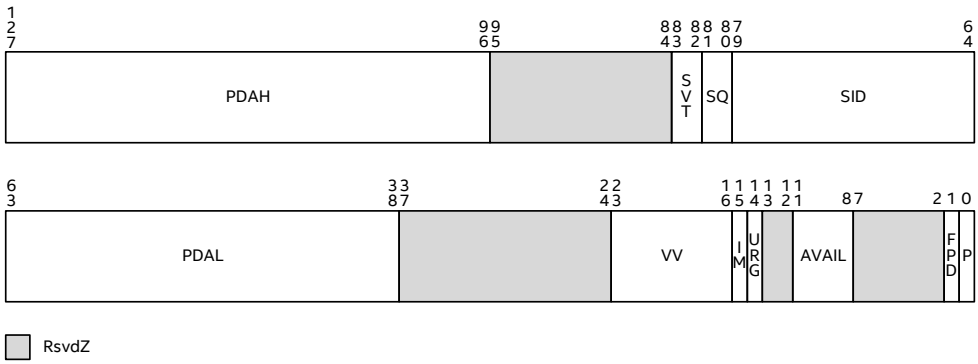


Figure 9-10. Interrupt Remap Table Entry Format for Posted Interrupts

Bits	Field	Description
127:96	PDAH: Posted Descriptor Address High	This field specifies address bits 63:32 of the 64-byte aligned Posted Interrupt Descriptor in memory used by hardware to post interrupt requests processed through this IRTE. Refer to Section 9.11 for Posted Interrupt Descriptor description. This field is evaluated by hardware only when the Present (P) field is Set. When evaluated, hardware treats upper address bits 63:HAW in this field as Reserved(0), where HAW is the Host Address Width of the platform.
95:84	R: Reserved	Reserved. Software must program these bits to 0. This field is evaluated by hardware only when the Present (P) field is Set.
83:82	SVT: Source Validation Type	This field specifies the type of validation that must be performed by the interrupt-remapping hardware on the source-id of the interrupt requests referencing this IRTE. • 00b: No requester-id verification is required. • 01b: Verify requester-id in interrupt request using SID and SQ fields in the IRTE. • 10b: Verify the most significant 8-bits of the requester-id (Bus#) in the interrupt request is equal to or within the Startbus# and EndBus# specified through the upper and lower 8-bits of the SID field respectively. This encoding may be used to verify interrupts originated behind PCI Express-to-PCI/PCI-X bridges. Refer Section 5.1.1 for more details. • 11b: Reserved. This field is evaluated by hardware only when the Present (P) field is Set.

Bits	Field	Description
81:80	SQ: Source-id Qualifier	The SVT field may be used to verify origination of interrupt requests generated by devices supporting phantom functions. If the SVT field is 01b, the following encodings are defined for the SQ field. • 00b: Verify the interrupt request by comparing all 16-bits of SID field with the 16-bit requester-id of the interrupt request. • 01b: Verify the interrupt request by comparing most significant 13 bits of the SID and requester-id of interrupt request, and comparing least significant two bits of the SID field and requester-id of interrupt request. (i.e., ignore the third least significant field of the SID field and requester-id). • 10b: Verify the interrupt request by comparing most significant 13 bits of the SID and requester-id of interrupt request, and comparing least significant bit of the SID field and requester-id of interrupt request. (i.e., ignore the second and third least significant fields of the SID field and requester-id). • 11b: Verify the interrupt request by comparing most significant 13 bits of the SID and requester-id of interrupt request. (i.e., ignore the least three significant fields of the SID field and requester-id). This field is evaluated by hardware only when the Present (P) field is Set and SVT field is 01b.
79:64	SID: Source Identifier	This field specifies the originator (source) of the interrupt request that references this IRTE. The format of the SID field is determined by the programming of the SVT field. If the SVT field is: • 01b: The SID field contains the 16-bit requester-id (Bus/Dev/Func #) of the device that is allowed to originate interrupt requests referencing this IRTE. The SQ field is used by hardware to determine which bits of the SID field must be considered for the interrupt request verification. • 10b: The most significant 8-bits of the SID field contains the Startbus#, and the least significant 8-bits of the SID field contains the Endbus#. Interrupt requests that reference this IRTE must have a requester-id whose bus# (most significant 8-bits of requester-id) has a value equal to or within the Startbus# to Endbus# range. This field is evaluated by hardware only when the Present (P) field is Set and SVT field is 01b or 10b.
63:38	PDAL: Posted Descriptor Address Low	This field specifies address bits 31:6 of the 64-byte aligned Posted Interrupt Descriptor in memory used by hardware to post interrupt requests processed through this IRTE. For optimal performance, software is recommended to reserve a full cacheline to host a Posted Interrupt Descriptor. Refer to Section 9.11 for Posted Interrupt Descriptor description. This field is evaluated by hardware only when the Present (P) field is Set.
37:24	R: Reserved	Reserved. Software must program these bits to 0. This field is evaluated by hardware only when the Present (P) field is Set.
23:16	VV: Virtual Vector	This 8-bit field contains the vector associated with the interrupt requests posted through this IRTE. This field is evaluated by hardware only when the Present (P) field is Set.
15	IM: IRTE Mode	Value of 1 in this field indicate interrupt requests processed through this IRTE are posted. Hardware implementations not supporting Posted Interrupts (PI=0 in Capability Register) treat this field as Reserved(0). Value of 0 in this field indicate interrupt requests processed through this IRTE are remapped. Refer to Section 9.9 for IRTE format for remapped interrupts.
14	URG: Urgent	This field indicates if the interrupt posted through this IRTE has latency sensitive processing requirements. • 0: Interrupt is not treated as urgent. • 1: Interrupt is treated as urgent. This field is evaluated by hardware only when the Present (P) field and IRTE Mode (IM) fields are both Set. Hardware implementations not supporting Posted Interrupts (PI=0 in Capability Register) treat this field as Reserved(0).
13:12	R: Reserved	Reserved. Software must program these bits to 0. This field is evaluated by hardware only when the Present (P) field is Set.

Bits	Field	Description
11:8	AVAIL: Available	This field is available to software. Hardware always ignores the programming of this field.
7:2	R: Reserved	Reserved. Software must program these bits to 0. This field is evaluated by hardware only when the Present (P) field is Set.
1	FPD: Fault Processing Disable	Enables or disables recording/reporting of faults caused by interrupt messages requests processed through this entry. • 0: Indicates fault recording/reporting is enabled for interrupt requests processed through this entry. • 1: Indicates fault recording/reporting is disabled for interrupt requests processed through this entry. This field is evaluated by hardware irrespective of the setting of the Present (P) field.
0	P: Present	The P field is used by software to indicate to hardware if the corresponding IRTE is present and initialized. • 0: Indicates the IRTE is not currently allocated to any interrupt sources. Block interrupt requests referencing this IRTE. • 1: Process interrupt requests referencing this IRTE per the programming of other fields in this IRTE.

9.11 Posted Interrupt Descriptor (PID)

The following figure and table describe the 64-byte aligned Posted Interrupt Descriptor. Software must allocate Posted Interrupt Descriptors is coherent (write-back) memory.

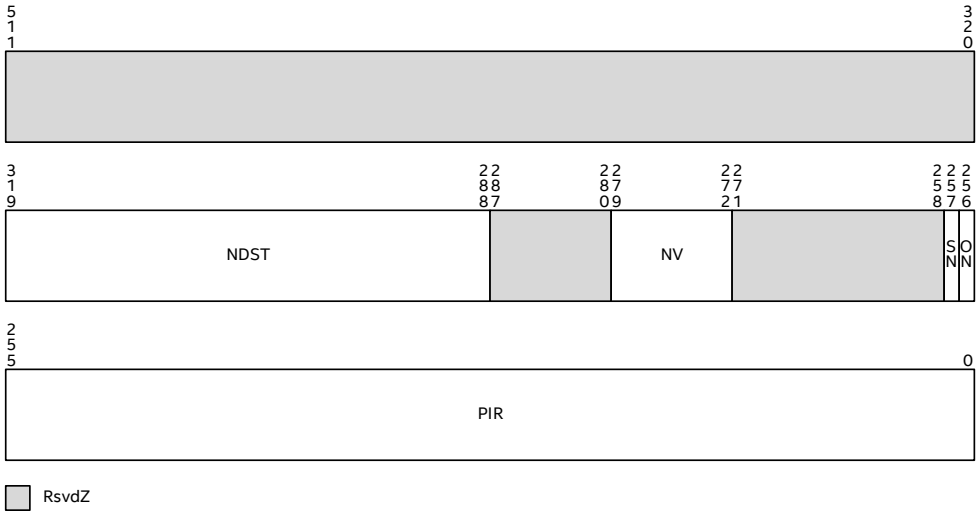


Figure 9-11. Posted Interrupt Descriptor Format

Bits	Field	Description
511:320	R: Reserved	Reserved. Software must program these bits to 0.

319:288	NDST: Notification Destination	This field specifies the Destination (Physical APIC-ID of the logical CPU) for the notification event. The format of this field is as follows:
		• Intel®64 xAPIC Mode (Physical):
		  • 319:304 - Reserved (0)
		  • 303:296 - APIC DestinationID[7:0]
		  • 295:288 - Reserved (0)
		• Intel®64 x2APIC Mode (Physical):
		  • 319:288 - APIC DestinationID[31:0]

287:280	R: Reserved	Reserved. Software must program these bits to 0.
279:272	NV: Notification Vector	This field specifies the physical vector used for the notification event. Notification events are issued as physical interrupts with trigger mode as Edge, and trigger mode level as Asserted.
271:258	R: Reserved	Reserved. Software must program these bits to 0.

Bits	Field	Description
257	SN: Suppress Notification	This field indicates if notification events must be suppressed when posting non-urgent interrupts to this descriptor. • 0: Do not suppress notification event. • 1: Suppress notification event. Non-urgent interrupts are interrupt requests processed through IRTE entries with IM field Set and URG field Clear. Refer to Section 9.10 for description of URG field in IRTE format for posted interrupts. If the value in this field is 1b at the time of hardware posting a non-urgent interrupt to PIR field, hardware functions as if the Outstanding Notification (ON) field value is 1b (i.e., hardware does not generate notification event nor modify the ON field). Refer to Section 5.2.3 on hardware operation for posting non-urgent interrupts.
256	ON: Outstanding Notification	This field indicates if a notification event is outstanding (pending processing by CPU or software) for this posted interrupt descriptor. • 0: No notification event outstanding for this descriptor. • 1: A notification event is outstanding for this descriptor. If this field is Clear at the time of hardware posting an interrupt to PIR field, hardware Sets it and generates a notification event. If this field is already Set at the time of hardware posting an interrupt to PIR field, notification event is not generated.
255:0	PIR: Posted Interrupt Requests	This 256-bit field (one bit for each vector) provide storage for posted interrupts destined for a specific virtual processor. An interrupt request remapped through an IRTE for posted interrupt (IRTE with IM field Set) is considered posted by hardware when the bit corresponding to vector value in the IRTE is Set in this field. Refer to Section 5.2.3 on hardware operation for interrupt-posting.

9.12 Host Permission Table Entries

The following figures and tables describe the Host Permission Table entries. HPT Entries that contain Page Permissions fields, PP*i*, follow the PP*i* format in Table 48. For additional detail of hardware usage of each field, refer to Section 4.2.4.

Table 48. Format of PP*i* in HPTL3E, HPTL2E, and HPTL1E

Bits	Field	Description
3:2	Reserved	Reserved. Software must program these bits to 0.
1	W: Write	If 0, write permission not granted.
0	R: Read	If 0, read permission not granted. For implementations reporting Zero-Length-Read (ZLR) field as Set in the Capability Register, read permission is not applicable to zero-length read requests if the Write (W) permission field is 1.

9.12.1 HPTL4E

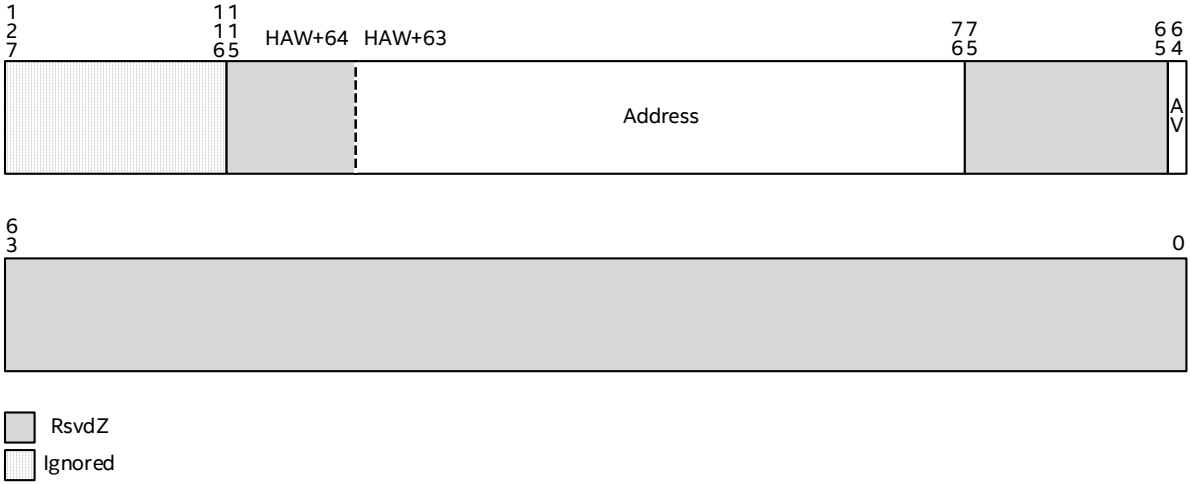


Figure 9-12. HPTL4E Format

Bits	Field	Description
127:116	Ignored	Ignored by hardware
115:HAW+64	Reserved	Reserved (0)
HAW+63:76	Address	Physical address of 4-KByte aligned HPTL3 table referenced by this entry. This field is ignored by hardware when AV is Clear.
75:65	Reserved	Reserved (0)
64	AV: Address Valid	Indicates the Address field points to a valid HPTL3 table.
63:0	Reserved	Reserved (0)

9.12.2 HPTL3E

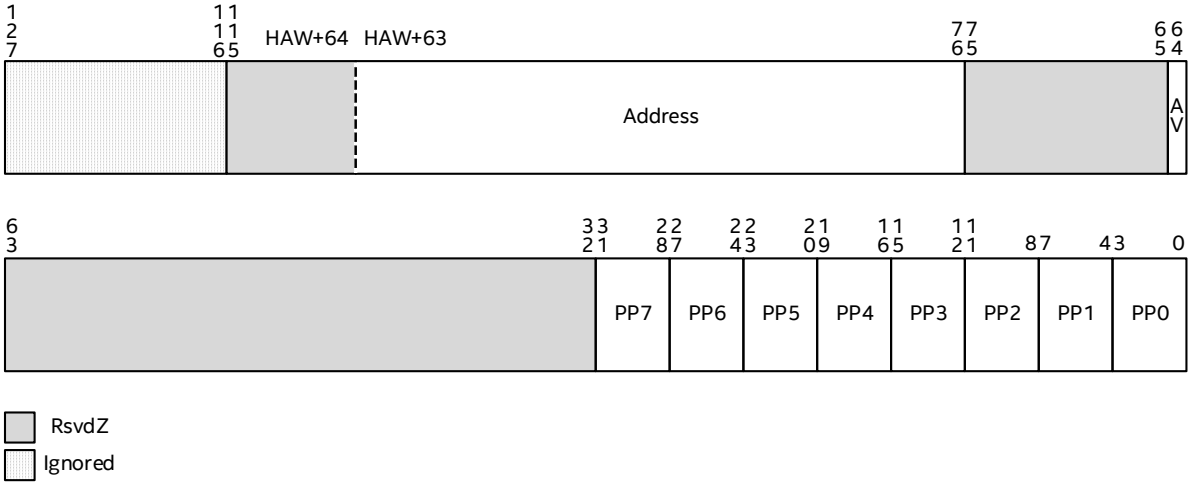


Figure 9-13. HPTL3E Format

Bits	Field	Description
127:116	Ignored	Ignored by hardware
115:HAW+64	Reserved	Reserved (0)
HAW+63:76	Address	Physical address of 4-KByte aligned HPTL2 table referenced by this entry. This field is ignored by hardware when AV is Clear.
75:65	Reserved	Reserved (0)
64	AV: Address Valid	Indicates the Address field points to a valid HPTL2 table.
63:32	Reserved	Reserved (0)
31:0	PP7:PP0	Permissions for the 1 GB pages covered by this entry. Refer to Table 48 for bit definitions for each PP*i* bit-field. This field is treated as Reserved (0) for implementations not supporting 1 GB page size in Host Permission Tables (HPT1GP=0 in the Extended Capability Register).

9.12.3 HPTL2E

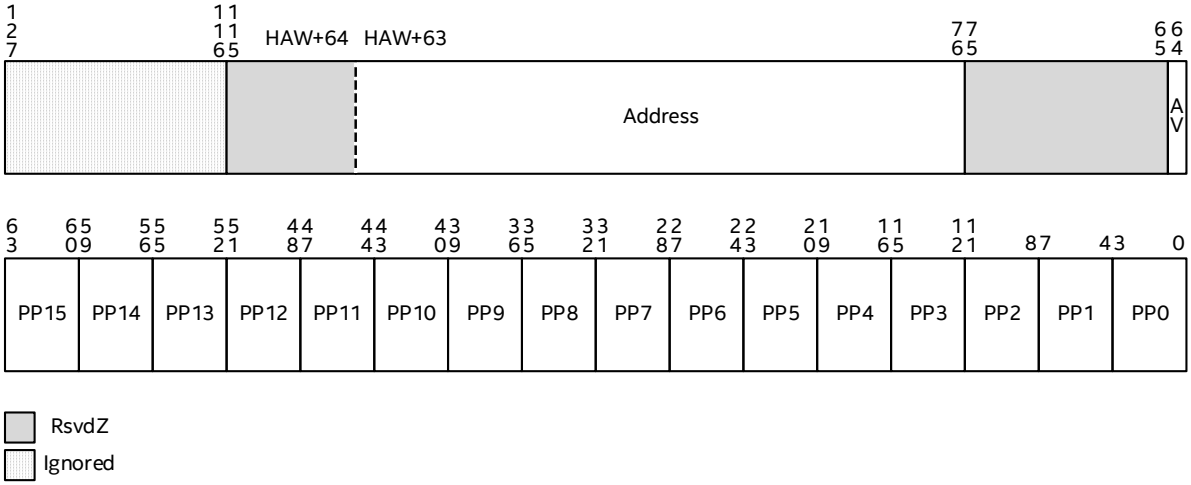


Figure 9-14. HPTL2E Format

Bits	Field	Description
127:116	Ignored	Ignored by hardware
115:HAW+64	Reserved	Reserved (0)
HAW+63:76	Address	Physical address of 4-KByte aligned HPTL1 table referenced by this entry. This field is ignored by hardware when AV is Clear.
75:65	Reserved	Reserved (0)
64	AV: Address Valid	Indicates the Address field points to a valid HPTL1 table.
63:0	PP15:PP0	Permissions for the 2 MB pages covered by this entry. Refer to Table 48 for bit definitions for each PP*i* bit-field.

9.12.4 HPTL1E

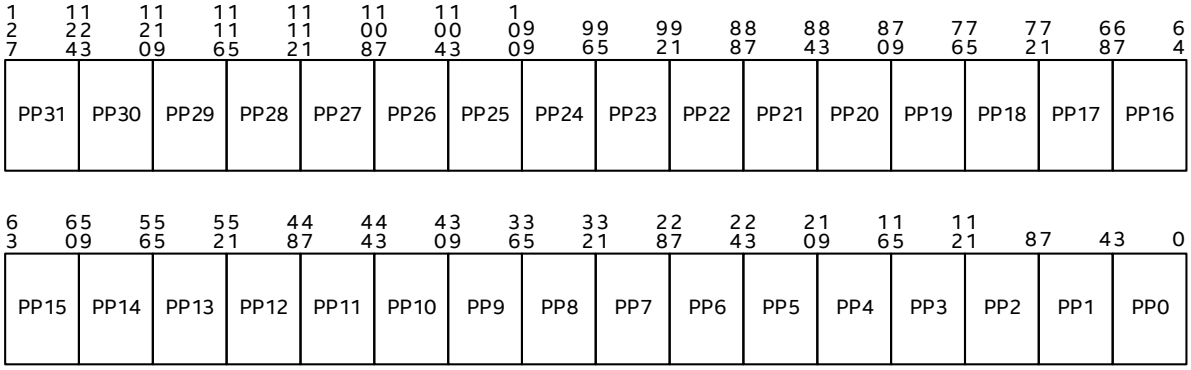


Figure 9-15. HPTL1E Format

Bits	Field	Description
127:0	PP31:PP0	Permissions for the 4 KB pages covered by this entry. Refer to Table 48 for bit definitions for each PP*i* bit-field.

10 Performance Monitoring

The purpose of the performance monitoring infrastructure, perfmon, is to support collection of information about key events occurring during operation of the remapping hardware, to aid performance tuning and debug. This can also be useful to understand usages of key features and operations supported by the device. The perfmon infrastructure includes:

- Capability registers to discover and enumerate the capabilities supported by an implementation.
- Configuration registers to configure and enable the device to monitor specific events.
- Counter registers that report counts of the events being monitored.

Details of these registers are in Section 11.4.13.

10.1 Performance Monitoring Discovery and Enumeration

Each remapping hardware unit has capability registers that indicate support for performance monitoring features and enumerate the capabilities, such as the number of counters, counter width, and events supported. If an implementation does not support performance monitoring, then the Performance Monitoring Capability Register (PERFCAP_REG) is 0 and the other performance monitoring capability, configuration, and counter registers are not supported.

Hardware implementation of perfmon can vary based on numerous parameters which are represented in this specification in the following manner:

- Number of Counter Registers (PERFCAP_REG.NCNTR). Denoted as ***c***.
- Number of globally supported Event Groups (PERFCAP_REG.EGCNT). Denoted as ***g***.
- Number of Event Groups supported for Counter ***c*** (PERFCNTRCAP_REG[***c***].EGCNT). Denoted as e_c.
- Number of Freeze or Overflow Registers. Denoted as ***m.***

The location of specific groups of registers can vary and are reported to software through the following offset registers:

- ***Counter Offset*** = PERFCNTROFF_REG + Remapping Hardware Register Base Address
- ***Configuration Offset*** = PERFCFGOFF_REG + Remapping Hardware Register Base Address
- ***Freeze Offset*** = PERFFRZOFF_REG + Remapping Hardware Register Base Address
- ***Overflow Offset*** = PERFOVFOFF_REG + Remapping Hardware Register Base Address

Software configures a counter to monitor events by specifying the Event Group Index and Event Select bitmask in the corresponding Counter Configuration Register (PERFCNTRCFG_REG). Table 49 defines the Event Groups and a set of Events within each Event Group.

An Event Capability Register (PERFEVNTCAP_REG) corresponding to each Event Group indicates the set of events supported in that Event Group. If an implementation does not support any events for a given Event Group, then the PERFEVNTCAP_REG.ES is zero, and software should not configure any counter to use that Event Group. The events in each Event Group are listed in Section 10.7. When enabling a counter to count events of a given Event Group, an error check is performed by hardware to verify if the configuration is supported. The error checking is described in Section 10.5.

Implementations may support the use of any Event with any counter or may restrict certain Event Groups and specific Events to be used only with certain counters. In the latter case, the Event Groups and sets of events supported by each counter are reported in the Counter Event Capability registers (PERFCNTREVCAP_REG) described in Section 11.4.13.22.

10.2 Performance Monitoring Configuration Registers

Software uses the Enhanced Command Interface described in Section 11.4.14 along with perfmon specific configuration, status, and counter registers to configure and control the performance monitoring hardware. The registers include global configuration and status registers as well as per-counter configuration registers and counter value registers.

Each set of per-counter configuration and counter registers operates independently. Software configures and enables each counter before it begins counting events.

Software configures event counting by selecting an available counter register and programming the corresponding Counter Configuration Register (PERFCNTRCFG_REG) with the desired Event Group and set of events to be monitored. The bits programmed in the Events Select field, PERFCNTRCFG_REG.ES, are interpreted as events in the specified Event Group programmed in the Event Group Index field, PERFCNTRCFG_REG.EGI. Each counter can only count events corresponding to a single Event Group at any one time. Software can configure a counter to count multiple events belonging to the same Event Group by setting multiple bits in the Event Select field. In this case, the counter value reflects the sum of all occurrences of the specified events. If independent (non-additive) counts are required for some events, software needs to program different counters, one per event to be monitored. Similarly, to count events corresponding to different Event Groups, software needs to configure multiple counters, at least one per Event Group desired, with the corresponding Event Select values.

Software may program the Filter Configuration Registers to count only specific occurrences of an event. Refer to Table 49 for per event filter compatibility. Filters are not supported if a counter is configured to count multiple events. The Counter Configuration Register and Filter Configuration Registers must be programmed before enabling the corresponding counter. Refer to Table 51 for which registers and fields are read-only when the Enable field is set in the corresponding Counter Configuration Register (PERFCNTRCFG_REG.EN = 1).

Software uses the Enable Perfmon Counter command to enable counting of a specific counter through the Enhanced Command Register. Similarly the Disable Perfmon Counter command is used to disable a counter.

Software can reset the state of the counter configuration and counter registers to their default initial values by issuing the Reset All Perfmon Counter Configuration command and Reset All Perfmon Counter Values command through the Enhanced Command Interface described in Section 11.4.14. This may be done at any time. Reset All Perfmon Counter Configuration results in all the counters being disabled and all configuration, filter, freeze, and overflow status registers set to their default value. Reset All Perfmon Counter Values command simultaneously clears all counter values.

10.3 Event Counters

The number of counters available is indicated by the Number of Counters field in the Performance Monitoring Capability Register (PERFCAP_REG.NCNTR). The address of the first counter register is determined by the Counter Offset Register. The address of each succeeding counter is determined by adding together the previous counter address and the counter stride (PERFCAP_REG.CS).

Software can read the Counter registers at any time. Software can write to a Counter register prior to enabling the counter to monitor events. Implementations may allow software to write to a Counter register while the counter is enabled, refer to Table 51 for access attributes for the conditions. Some usages of software writing to a counter register include:

- Write to a counter while it is Disabled (PERFCNTRCFG_REG.EN=0) to set the counter with a specific value.

- Write to a counter after it overflows to set the counter with a specific value. Note that the counter may be Enabled and currently counting (not frozen).
- Write to a currently frozen counter to set the counter with a specific value.

The implementation-dependent width of the counter registers is indicated in the Counter Width field, PERFCAP_REG.CW. While this may be less than 64 bits, software is allowed to write a full 64-bit value to the register without an error. Bits above the reported counter width are ignored. If Per-Counter Capabilities is supported by hardware (PERFCAP_REG.PCCS = 1), then the actual width of each counter may be indicated in the corresponding Counter Capability register. This allows an implementation to support counters of different widths.

10.3.1 Counter Overflow

While enabled to count events and an event occurrence causes the counter value to increment and roll over to or past zero, this is termed as a counter overflow. Upon overflow, the corresponding bit in the Overflow Status Registers (PERFOVFSTS_REG) is set. If supported and enabled, an interrupt may also be generated. Normally, the counter continues to count events and does not stop counting upon overflow. If supported, software can set the Global Freeze on Overflow field in the counter configuration register, PERFCNTRCFG_REG.GFO. If this field is set for a counter, an overflow of that counter results in the freeze bits of all counters to be set in all instances of PERFFRZSTS_REG. This forces all the counters to stop counting (freeze) and retain their current count value (until explicitly written or reset by software). The current freeze state of the counters is reported in PERFFRZSTS_REG.

Software can treat the counter register as a signed integer up to the supported width. For example, to cause an overflow on the first occurrence of an event, software can write an initial count value of -1 (e.g., 0xFFFFFFFFFF for a 40-bit counter) prior to enabling the counter. This causes the first occurrence of the specified event, after enabling the counter, to cause an overflow of the counter value and trigger an interrupt (if enabled).

10.3.2 Counter Stop and Resume

The current freeze/unfreeze status of a counter is reported in the PERFFRZSTS_REG registers. These bits are read-only by software. Software can update the freeze status of the counters by issuing commands through the Enhanced Command Interface. The Enhanced Command Capability Registers indicate which commands are supported by hardware. Hardware sets the freeze bits of all counters when a counter encounters an overflow and has the Global Freeze on Overflow field set in the corresponding Counter Configuration Register (PERFCNTRCFG_REG.GFO = 1). Refer to Section 11.4.14 for details of supported commands.

The commands software can use to control the freeze state of the counters are as follows:

- Freeze All Perfmon Counters command: Simultaneously freeze all counters and set all bits in the in PERFFRZSTS_REG registers that pertain to a valid counter.
- Unfreeze All Perfmon Counters command: Simultaneously unfreeze all counters and clear all bits in the in PERFFRZSTS_REG registers that pertain to a valid counter.

When a bit in the Performance Monitoring Freeze Status Registers changes to 1, either due to a Global Freeze on Overflow event or freeze commands issued by software, the associated counter is frozen and will not count future events. Likewise, a counter that was previously frozen, may be resumed by issuing an unfreeze command to the Enhanced Command Interface. This is referred to as an unfreeze operation on the counter and causes it to resume counting of configured events. When unfrozen, the counter continues to increment, starting from the current counter value at the time of the unfreeze operation.

10.4 Filter Support

Filters constrain the counting of selected events based on one or more conditions specified in the filter configuration registers. The filters are described in greater detail beginning in Section 11.4.13.16. A corresponding set of filter configuration registers (one per filter type) is defined for each counter. Each filter type has a corresponding field in the Performance Monitoring Capability Register that reports which, if any, filters are supported by hardware.

Each event might only support a subset of filter types or may not support filters at all. When counting multiple events, filters cannot be enabled. Such a condition will be detected as an error by the Error Check performed in hardware. Software specifies the filters to apply to the events monitored by a given counter by programming the Enable field in the corresponding filter configuration registers for that counter.

When multiple filters are configured for a counter, only the events that satisfy all the specified filters are counted (i.e. a logical AND of all the filter conditions).

10.5 Performance Monitoring Counter Configuration Error Checks

The following checks are performed when an Enable Perfmon Counter command is issued to the Enhanced Command Interface. If any error check fails, then the error is reported in the Enhanced Command Response Register. If no error is detected, then successful completion is reported in the Enhanced Command Response Register and the Enable field in the corresponding Counter Configuration Register is Set to indicate that the counter is enabled. The hardware detects the following errors for the Enable Perfmon Counter commands:

- The Counter Index is greater than the number of counter registers present in hardware.
- Multiple events are enabled to be counted and one or more filters are also enabled.
- The Event Group is not supported by the counter.
- The Event Select field attempts to enable an event within an event group that is not supported by the counter.
- The Event Select field is zero.
- Interrupt on Overflow or Global Freeze on Overflow is enabled when not supported. Interrupt on Overflow support or Global Freeze on Overflow support is indicated by IOS and GFS in the Performance Monitoring Capability register. If a Counter Capability register has the Per-Counter Capabilities field set, then the counter specific IOS and GFS fields indicate if each is supported for that counter.
-
- A filter is enabled for an event that is not supported by hardware as a valid filter and event pairing.

10.6 Interrupt Generation

If the Interrupt on Overflow Support field in PERFCAP_REG is 1, then the implementation supports generation of an event when a counter overflows. Generation of the event is controlled by the Performance Monitoring Interrupt Control, Data, and Address registers. When a performance interrupt is generated by hardware, the Performance Interrupt Status field is set in the Performance Monitoring Interrupt Status Register (PERFINTRSTS_REG.PIS). The Performance Interrupt Status field should be cleared by software by writing a 1, once the interrupt has been serviced.

Upon receiving the interrupt, software can read the Overflow Status register to identify which counters have encountered an overflow condition. Multiple bits may be set in this register indicating that multiple counters have overflowed. Software can clear this register by writing 1 to each bit that it wants to clear.

Read completions due to software reading the Performance Monitoring Interrupt Status Register (PERFINTRCTL_REG) or the Performance Monitoring Interrupt Control Register (PERFINTRSTS_REG) must push (commit) any in-flight performance monitoring interrupt messages generated by the respective hardware unit.

10.7 Performance Monitoring Events

The performance monitoring events in this list have a common definition in all VT-d implementations. However, not all implementations necessarily support all events and applicable filters per events. The capability registers indicate which events are supported by an implementation. Additional events and applicable filters may be added in future implementations. Hardware may report up to a maximum of 15 Event Groups supported through the Event Group Count field in the Performance Monitoring Capability Register. Software can detect which applicable filters are supported for an event on a given implementation by configuring and enabling a counter for the targeted event with a desired filter. Upon enabling the counter, the error check logic will report an error if the filter and event are not a supported pairing.

Table 49. Performance Monitoring Event List

Event Group Index	Event Bit Enable	Mnemonic	Event Description	Applicable Filters
0h	0	IOMMU_CLOCKS	Clock cycles of the IOMMU Hardware	None
	1	IOMMU_REQUESTS	All Incoming Requests (Untranslated, translated, translation)	AT Type
	2	PW_OCCUPANCY	Occupancy of Requests requiring Page Walk	None
	3	ATS_BLOCKED	Back pressure of Translation Responses	None
	4		Unused	
	5	TOTAL_OCCUPANCY	Occupancy of Requests visible to IOMMU	None
	6	MEM_OCCUPANCY	Occupancy of Outstanding accesses to system memory	None
	[27:7]		Unused	
1h	0	IOMMU_MRDS	Memory Reads	None
	[2:1]		Unused	
	3	IOMMU_ATOMIC_AD_UPDATE	Atomic updates of Access/Dirty bits during page walk	None
	4		Unused	
	5	IOMMU_MEM_BLOCKED	Back-Pressure of Memory Requests	None
	6	PG_REQ_POSTED	Page Requests Received and Posted	None
	7	PG_REQ_OVERFLOW	Auto-generated responses due to page request queue overflow	None
	[27:8]		Unused	

Table 49. Performance Monitoring Event List

Event Group Index	Event Bit Enable	Mnemonic	Event Description	Applicable Filters
2h	0	CTXT_CACHE_LOOKUP	Context Cache Lookups	None
	1	CTXT_CACHE_HIT	Context Cache Hits	None
	2	PASID_CACHE_LOOKUP	Pasid Cache Lookups	None
	3	PASID_CACHE_HIT	Pasid Cache Hits	None
	4	SS_NONLEAF_LOOKUP	Second Stage Translation NonLeaf Cache Lookups	None
	5	SS_NONLEAF_HIT	Second Stage Translation NonLeaf Cache Hits	Page Table Level
	6	FS_NONLEAF_LOOKUP	First Stage Translation NonLeaf Cache Lookups	None
	7	FS_NONLEAF_HIT	First Stage Translation NonLeaf Cache Hits	Page Table Level
	8	HPT_NONLEAF_LOOKUP	HPT Paging Structure Cache Lookups	AT Type
	9	HPT_NONLEAF_HIT	HPT Paging Structure Cache Hits	AT Type, Page Table Level
	[27:10]		Unused	
3h	0	IOTLB_LOOKUP	IOTLB Lookups	None
	1	IOTLB_HIT	IOTLB Hit	Page Table Level
	2	HPT_LEAF_LOOKUP	HPT Leaf Cache Lookups	
	3	HPT_LEAF_HIT	HPT Leaf Cache Hits	Page Table Level
	[27:4]		Unused	
4h	0	INT_CACHE_LOOKUP	Interrupt Cache Lookup	None
	1	INT_CACHE_HIT_NONPOSTED	Interrupt Cache Hit with Non-Posted Interrupt Data	None
	2	INT_CACHE_HIT_POSTED	Interrupt Cache Hit with Posted Interrupt Data	None
	[27:3]		Unused	

11 Register Descriptions

This chapter describes the structure and use of the remapping registers.

11.1 Register Location

The register set for each remapping hardware unit in the platform is placed at a size-aligned memory-mapped location. The exact location of the register region is implementation-dependent, and is communicated to system software by BIOS through the ACPI DMA-remapping hardware reporting structures (described in Chapter 8). For security, hardware implementations that support relocating these registers in the system address map must provide ability to lock its location by hardware specific secure initialization software.

11.2 Software Access to Registers

Software interacts with the remapping hardware by reading and writing its memory-mapped registers. The following requirements are defined for software access to these registers.

- Software is expected to access 32-bit registers as aligned doublewords. For example, to modify a field (e.g., bit or byte) in a 32-bit register, the entire doubleword is read, the appropriate field(s) are modified, and the entire doubleword is written back.
- Software must access 64-bit and 128-bit registers as either aligned quadwords or aligned doublewords. Hardware may disassemble a quadword register access as two double-word accesses. In such cases, hardware is required to complete the quad-word read or write request in order (lower doubleword first, upper double-word second).
- When updating registers through multiple accesses (whether in software or due to hardware disassembly), certain registers may have specific requirements on how the accesses must be ordered for proper behavior. These are documented as part of the respective register descriptions.
- For compatibility with future extensions or enhancements, software must assign the last read value to all "Reserved and Preserved" (RsvdP) fields when written. In other words, any updates to a register must be read so that the appropriate merge between the RsvdP and updated fields will occur. Also, software must assign a value of zero for "Reserved and Zero" (RsvdZ) fields when written.
- Locked operations to remapping hardware registers are not supported. Software must not issue locked operations to access remapping hardware registers.

11.3 Register Attributes

The following table defines the attributes used in the remapping Registers. The registers are discussed in Section 11.4.

Attribute	Description
RW	Read-Write field that may be either set or cleared by software to the desired state.
RW1C	"Read-only status, Write-1-to-clear status" field. Software can read this bit to find the value of status. Software can write a value of '1' to Clear this bit. Writing a '0' to the bit has no effect.
RW1CS	"Sticky Read-only status, Write-1-to-clear status" field. Software can read this bit to find the value of status. Software can write a value of '1' to Clear this bit. Writing a '0' to the bit has no effect.This bit is only re-initialized to its default value by a "Power Good Reset".
RWL	"Lockable Read-Write". Software may read or write this field when not locked. When locked, the field is read only. The field's locked status is controlled by a separate configuration bit or other logic.
RWLV	"Lockable Read-Write Volatile". Software may read or write this field when not locked. When locked, the field is read only by software. The field's locked status is controlled by a separate configuration bit or other logic. Hardware may change the value of this field at any time including when locked.
RO	Read-only field that cannot be directly altered by software.
ROS	"Sticky Read-only" field that cannot be directly altered by software. These bits are only re-initialized to their default value by a "Power Good Reset".
WO	Write-only field. The value returned by hardware on read is undefined.
RsvdP	"Reserved and Preserved" field that is reserved for future RW implementations. Registers are read-only and must return 0 when read. Software must preserve the value read for writes.
RsvdZ	"Reserved and Zero" field that is reserved for future RW1C implementations. Registers are read-only and must return 0 when read. Software must use 0 for writes.

11.4 Register Descriptions

The following table summarizes the remapping hardware memory-mapped registers.

Offset	Register Name	Size	Description
000h	Version Register	32	Architecture version supported by the implementation.
004h	Reserved	32	Reserved
008h	Capability Register	64	Hardware reporting of capabilities.
010h	Extended Capability Register	64	Hardware reporting of extended capabilities.
018h	Global Command Register	32	Register controlling general functions.
01Ch	Global Status Register	32	Register reporting general status.
020h	Root Table Address Register	64	Register to set up location of root table.
028h	Context Command Register	64	Register to manage context-entry cache.
030h	Reserved	32	Reserved
034h	Fault Status Register	32	Register to report Fault/Error status
038h	Fault Event Control Register	32	Interrupt control register for fault events.
03Ch	Fault Event Data Register	32	Interrupt message data register for fault events.
040h	Fault Event Address Register	32	Interrupt message address register for fault event messages.
044h	Fault Event Upper Address Register	32	Interrupt message upper address register for fault event messages.
048h	Reserved	64	Reserved
050h	Reserved	64	Reserved
058h	Reserved	64	Reserved
060h	Reserved	32	Reserved
064h	Protected Memory Enable Register	32	Register to enable DMA-protected memory region(s).
068h	Protected Low Memory Base Register	32	Register pointing to base of DMA-protected low memory region.
06Ch	Protected Low Memory Limit Register	32	Register pointing to last address (limit) of the DMA-protected low memory region.
070h	Protected High Memory Base Register	64	Register pointing to base of DMA-protected high memory region.
078h	Protected High Memory Limit Register	64	Register pointing to last address (limit) of the DMA-protected high memory region.
080h	Invalidation Queue Head	64	Offset to the invalidation queue entry that will be read next by hardware.
088h	Invalidation Queue Tail	64	Offset to the invalidation queue entry that will be written next by software.
090h	Invalidation Queue Address Register	64	Base address of memory-resident invalidation queue.
098h	Reserved	32	Reserved

Offset	Register Name	Size	Description
09Ch	Invalidation Completion Status Register	32	Register to indicate the completion of an Invalidation Wait Descriptor with IF=1.
0A0h	Invalidation Completion Event Control Register	32	Register to control Invalidation Queue Events
0A4h	Invalidation Completion Event Data Register	32	Invalidation Queue Event message data register for Invalidation Queue events.
0A8h	Invalidation Completion Event Address Register	32	Invalidation Queue Event message address register for Invalidation Queue events.
0ACh	Invalidation Completion Event Upper Address Register	32	Invalidation Queue Event message upper address register for Invalidation Queue events.
0B0h	Invalidation Queue Error Record Register	64	Register to record various errors related to Invalidation Queue.
0B8h	Interrupt Remapping Table Address Register	64	Register indicating Base Address of Interrupt Remapping Table.
0C0h	Page Request Queue Head Register	64	Offset to the page request queue entry that will be processed next by software.
0C8h	Page Request Queue Tail Register	64	Offset to the page request queue entry that will be written next by hardware.
0D0h	Page Request Queue Address Register	64	Base address of memory-resident page request queue.
0D8h	Reserved	32	Reserved
0DCh	Page Request Status Register	32	Register to indicate one or more pending page requests in page request queue.
0E0h	Page Request Event Control Register	32	Register to control page request events.
0E4h	Page Request Event Data Register	32	Page request event message data register.
0E8h	Page Request Event Address Register	32	Page request event message address register
0ECh	Page Request Event Upper Address Register	32	Page request event message upper address register.
100h	MTRR Capability Register	64	Register for MTRR capability reporting.
108h	MTRR Default Type Register	64	Register to configure MTRR default type.
120h	Fixed-range MTRR Register for 64K_00000	64	Fixed-range memory type range register for 64K range starting at 00000h.
128h	Fixed-range MTRR Register for 16K_80000	64	Fixed-range memory type range register for 16K range starting at 80000h.
130h	Fixed-range MTRR Register for 16K_A0000	64	Fixed-range memory type range register for 16K range starting at A0000h.
138h	Fixed-range MTRR Register for 4K_C0000	64	Fixed-range memory type range register for 4K range starting at C0000h.
140h	Fixed-range MTRR Register for 4K_C8000	64	Fixed-range memory type range register for 4K range starting at C8000h.
148h	Fixed-range MTRR Register for 4K_D0000	64	Fixed-range memory type range register for 4K range starting at D0000h.

Offset	Register Name	Size	Description
150h	Fixed-range MTRR Register for 4K_D8000	64	Fixed-range memory type range register for 4K range starting at D8000h.
158h	Fixed-range MTRR Register for 4K_E0000	64	Fixed-range memory type range register for 4K range starting at E0000h.
160h	Fixed-range MTRR Register for 4K_E8000	64	Fixed-range memory type range register for 4K range starting at E8000h.
168h	Fixed-range MTRR Register for 4K_F0000	64	Fixed-range memory type range register for 4K range starting at F0000h.
170h	Fixed-range MTRR Register for 4K_F8000	64	Fixed-range memory type range register for 4K range starting at F8000h.
180h	Variable-range MTRR Base0	64	Variable-range memory type range0 base register.
188h	Variable-range MTRR Mask0	64	Variable-range memory type range0 mask register.
190h	Variable-range MTRR Base1	64	Variable-range memory type range1 base register.
198h	Variable-range MTRR Mask1	64	Variable-range memory type range1 mask register.
1A0h	Variable-range MTRR Base2	64	Variable-range memory type range2 base register.
1A8h	Variable-range MTRR Mask2	64	Variable-range memory type range2 mask register.
1B0h	Variable-range MTRR Base3	64	Variable-range memory type range3 base register.
1B8h	Variable-range MTRR Mask3	64	Variable-range memory type range3 mask register.
1C0h	Variable-range MTRR Base4	64	Variable-range memory type range4 base register.
1C8h	Variable-range MTRR Mask4	64	Variable-range memory type range4 mask register.
1D0h	Variable-range MTRR Base5	64	Variable-range memory type range5 base register.
1D8h	Variable-range MTRR Mask5	64	Variable-range memory type range5 mask register.
1E0h	Variable-range MTRR Base6	64	Variable-range memory type range6 base register.
1E8h	Variable-range MTRR Mask6	64	Variable-range memory type range6 mask register.
1F0h	Variable-range MTRR Base7	64	Variable-range memory type range7 base register.
1F8h	Variable-range MTRR Mask7	64	Variable-range memory type range7 mask register.
200h	Variable-range MTRR Base8	64	Variable-range memory type range8 base register.
208h	Variable-range MTRR Mask8	64	Variable-range memory type range8 mask register.
210h	Variable-range MTRR Base9	64	Variable-range memory type range9 base register.
218h	Variable-range MTRR Mask9	64	Variable-range memory type range9 mask register.
300h	Performance Monitoring Capability Register	64	Hardware Reporting of Performance Monitoring Capabilities
308h	Reserved	64	Reserved
310h	Performance Monitoring Configuration Offset Register	32	Hardware reporting of ***Configuration Offset***
314h	Performance Monitoring Freeze Offset Register	32	Hardware reporting of ***Freeze Offset***
318h	Performance Monitoring Overflow Offset Register	32	Hardware reporting of ***Overflow Offset***

Offset	Register Name	Size	Description
31Ch	Performance Monitoring Counter Offset Register	32	Hardware reporting of ***Counter Offset***
320h	Reserved	32	Reserved
324h	Performance Monitoring Interrupt Status Register	32	Register to report Perfmon Interrupt status
328h	Performance Monitoring Interrupt Control Register	32	Interrupt control register for Perfmon events.
32Ch	Performance Monitoring Interrupt Data Register	32	Interrupt message data register for Perfmon events.
330h	Performance Monitoring Interrupt Address Register	32	Interrupt message address register for Perfmon event messages.
334h	Performance Monitoring Interrupt Upper Address Register	32	Interrupt message upper address register for Perfmon event messages.
338h-37Fh	Reserved	64	Reserved
380h-3FFh	Performance Monitoring Event Capability Register(s)	64	Hardware Reporting of supported performance monitoring events.
400h	Enhanced Command Register	64	Register to submit command and operand to DMA Remapping hardware.
408h	Enhanced Command Extended Operand Register	64	Register to submit additional operands to DMA Remapping hardware.
410h	Enhanced Command Response Register	64	Register to receive responses from DMA Remapping hardware.
418h	Reserved	64	Reserved for future expansion of the Enhanced Command Response Register.
420h	Enhanced Command Status Register 0	64	Reporting of hardware state associated with Enhanced commands.
428h	Enhanced Command Status Register 1	64	Reporting of hardware state associated with Enhanced commands.
430h	Enhanced Command Capability Register 0	64	Hardware reporting of commands supported by DMA Remapping hardware.
438h	Enhanced Command Capability Register 1	64	Hardware reporting of commands supported by DMA Remapping hardware.
440h	Enhanced Command Capability Register 2	64	Hardware reporting of commands supported by DMA Remapping hardware.
448h	Enhanced Command Capability Register 3	64	Hardware reporting of commands supported by DMA Remapping hardware.
450h	Reserved	64	Reserved for future expansion of Enhanced command Status registers
458h	Reserved	64	Reserved for future expansion of Enhanced command Status registers
DC0h	RDT Configuration Register	64	Register to allow software control of RMID and CLOS associated with accesses to memory from remapping hardware.
DC8h-DFFh	Reserved	64	Reserved for future expansion of RDT Configuration Register
E00h	Virtual Command Register	64	Register to submit command and operand to virtual DMA Remapping hardware.

Offset	Register Name	Size	Description
E08h	Virtual Command Extended Operand Register	64	Register to submit additional operands to virtual DMA Remapping hardware.
E10h	Virtual Command Response Register	64	Register to receive responses from virtual DMA Remapping hardware.
E18h-E2Fh	Reserved	64	Reserved for future expansion of the Virtual Command Response Register.
E30h	Virtual Command Capability Register	64	Hardware reporting of commands supported by virtual-DMA Remapping hardware.
E38h-E4Fh	Reserved	64	Reserved for future expansion of the Virtual Command Capability Register.
XXXh	IOTLB Registers[1]	64	IOTLB registers consists of two 64-bit registers. Section 11.4.6.3 and Section 11.4.6.4 describes the format of the registers.
YYYh	Fault Recording Registers [n][1]	128	Registers to record the translation faults. The starting offset of the fault recording registers is reported through the Capability Register.
Freeze Offset	Performance Monitoring Freeze Status Register(s)	64	Hardware reporting of freeze state of Perfmon Counters.
Overflow Offset	Performance Monitoring Overflow Status Register(s)	64	Registers to observe and update overflow state of Perfmon Counters.
Configuration Offset	Performance Monitoring Counter Configuration Register(s), Performance Monitoring Requester ID Filter Configuration Register(s), Performance Monitoring Domain ID Filter Configuration Register(s), Performance Monitoring PASID Filter Configuration Register(s), Performance Monitoring Address Type Filter Configuration Register(s), Performance Monitoring Page Table Level Filter Configuration Register(s), Performance Monitoring Counter Capabilities Register(s), Performance Monitoring Counter Event Capabilities Register(s)	Varies	Registers to configure perfmon counter(s) and hardware reporting of per counter capabilities. Refer to register definitions for calculation of exact offsets.
Counter Offset	Performance Monitoring Counter Register(s)	64	Registers to observe and update Perfmon Counter Values.

1. Hardware implementations may place IOTLB registers and fault recording registers in any unused or reserved addresses in the 4KB register space, or place them in adjoined 4KB regions. If one or more adjunct 4KB regions are used, unused addresses in those pages must be treated as reserved by hardware. Location of these registers is implementation dependent, and software must read the Capability Register to determine their offset location.

11.4.1 Version Register

Figure 11-1. Version Register

Abbreviation	VER_REG
General Description	Register to report the implementation version. Backward compatibility for the architecture is maintained with new revision numbers, allowing software to load remapping hardware drivers written for prior versions.
Register Offset	000h

Bits	Access	Default	Field	Description
31:8	RsvdZ	0h	R: Reserved	Reserved.
7:4	RO	Xh	MAX: Major Version number	Indicates Major Version of Implementation.
3:0	RO	Yh	MIN: Minor Version number	Indicates Minor Version of Implementation.

11.4.2 Capability Register

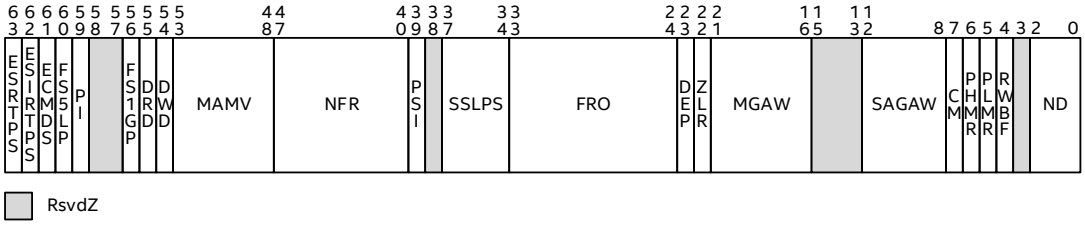


Figure 11-2. Capability Register

Abbreviation	CAP_REG
General Description	Register to report general remapping hardware capabilities
Register Offset	008h

Bits	Access	Default	Field	Description
63	RO	X	ESRTPS: Enhanced Set Root Table Pointer Support	• 0: Hardware does not invalidate all DMA remapping hardware translation caches as part of SRTP flow. • 1: Hardware invalidates all DMA remapping hardware translation caches as part of SRTP flow.
62	RO	X	ESIRTPS: Enhanced Set Interrupt Remap Table Pointer Support	• 0: Hardware does not invalidate all Interrupt remapping hardware translation caches as part of SIRTP flow. • 1: Hardware invalidates all Interrupt remapping hardware translation caches as part of SIRTP flow.
61	RO	X	ECMDS: Enhanced Command Support	• 0: Hardware does not support enhanced command interface. • 1: Hardware supports enhanced command interface.
60	RO	X	FS5LP: First Stage 5-level Paging Support	• 0: Hardware does not support 5-level paging for first-stage translation. • 1: Hardware supports 5-level paging for first-stage translation. Hardware implementations reporting First-stage Translation Support (FSTS) as Clear also report this field as Clear.
59	RO	X	PI: Posted Interrupts Support	• 0: Hardware does not support Posting of Interrupts. • 1: Hardware supports Posting of Interrupts. Hardware implementations reporting Interrupt Remapping support (IR) field in Extended Capability Register as Clear also report this field as Clear.
58:57	RsvdZ	0h	R: Reserved	Reserved.
56	RO	X	FS1GP: First Stage 1-GByte Page Support	A value of 1 in this field indicates 1-GByte page size is supported for first-stage translation. Hardware implementations reporting First-stage Translation Support (FSTS) as Clear also report this field as Clear.

Bits	Access	Default	Field	Description
55	RO	X	DRD: Read Draining	• 0: Hardware does not support draining of read requests on IOTLB Invalidation. • 1: Hardware supports draining of read requests on IOTLB Invalidation. Refer to Section 6.5.4 for description of DMA read draining. Hardware implementation with Major Version 2 or higher (VER_REG), always performs required drain without software explicitly requesting a drain in IOTLB invalidation. This field is deprecated and hardware will always report it as 1 to maintain backward compatibility with software.
54	RO	X	DWD: Write Draining	• 0: Hardware does not support draining of write requests on IOTLB Invalidation. • 1: Hardware supports draining of write requests on IOTLB Invalidation. Refer Section 6.5.4 for description of DMA write draining. Hardware implementation with Major Version 2 or higher (VER_REG), always performs required drain without software explicitly requesting a drain in IOTLB invalidation. This field is deprecated and hardware will always report it as 1 to maintain backward compatibility with software.
53:48	RO	X	MAMV: Maximum Address Mask Value	The value in this field indicates the maximum supported value for the Address Mask (AM) field in the Invalidation Address register (IVA_REG), and IOTLB Invalidation Descriptor (*iotlb_inv_dsc*) used for invalidations of second-stage translation. This field is valid when the PSI field in Capability register is reported as Set. Independent of value reported in this field, implementations supporting SMTS must support address-selective PASID-based IOTLB invalidations (*p_iotlb_inv_dsc*) with any defined address mask.
47:40	RO	X	NFR: Number of Fault-recording Registers	Number of fault recording registers is computed as N+1, where N is the value reported in this field. Implementations must support at least one fault recording register (NFR = 0) for each remapping hardware unit in the platform. The maximum number of fault recording registers per remapping hardware unit is 256.
39	RO	X	PSI: Page Selective Invalidation	• 0: Hardware supports only global and domain-selective invalidates for IOTLB. • 1: Hardware supports page-selective, domain-selective, and global invalidates for IOTLB. Hardware implementations reporting this field as Set are recommended to support a Maximum Address Mask Value (MAMV) value of at least 9 (or 18 if supporting 1GB pages with second level translation). This field is applicable only for IOTLB invalidation descriptor. Irrespective of value reported in this field, implementations supporting SMTS must support page/address selective PASID-based IOTLB invalidation descriptor.
38	RsvdZ	0h	R: Reserved	Reserved.

Bits	Access	Default	Field	Description
37:34	RO	X	SSLPS: Second Stage Large Page Support	This field indicates the large page sizes supported by hardware. A value of 1 in any of these bits indicates the corresponding large-page size is supported. The large-page sizes corresponding to various bit positions within this field are: • 0: 21-bit offset to page frame (2MB) • 1: 30-bit offset to page frame (1GB) • 2: Reserved • 3: Reserved Hardware implementations supporting a specific large-page size must support all smaller large-page sizes. i.e., only valid values for this field are 0000b, 0001b, 0011b.
33:24	RO	X	FRO: Fault-recording Register offset	This field specifies the offset of the first fault recording register relative to the register base address of this remapping hardware unit. If the register base address is X, and the value reported in this field is Y, the address for the first fault recording register is calculated as X+(16*Y).
23	R0	X	DEP: Deprecated	Deprecated. This field must be reported as 0 to ensure backward compatibility with older software.
22	RO	X	ZLR: Zero Length Read	• 0: Indicates the remapping hardware unit blocks (and treats as fault) zero length DMA read requests to write-only pages. • 1: Indicates the remapping hardware unit supports zero length DMA read requests to write-only pages. DMA remapping hardware implementations are recommended to report ZLR field as Set.
21:16	RO	X	MGAW: Maximum Guest Address Width	This field indicates the maximum guest physical address width supported by second-stage translation in remapping hardware. The Maximum Guest Address Width (MGAW) is computed as (N+1), where N is the valued reported in this field. For example, a hardware implementation supporting 48-bit MGAW reports a value of 47 (101111b) in this field. If the value in this field is X, untranslated DMA requests with addresses above $2^{(X+1)}$-1 that are subjected to second-stage translation are blocked by hardware. Device-TLB translation requests to addresses above $2^{(X+1)}$-1 that are subjected to second-stage translation from allowed devices return a null Translation-Completion Data with R=W=0. Guest addressability for a given DMA request is limited to the minimum of the value reported through this field and the adjusted guest address width of the corresponding page-table structure. (Adjusted guest address widths supported by hardware are reported through the SAGAW field). Implementations must support MGAW at least equal to the physical addressability (host address width) of the platform. All Root-Complex integrated devices and fabrics must implement at least MGAW address bits. Implementations must report a non-zero value in this field including when the second-stage translation support (SSTS) field is reported as Clear.
15:13	RsvdZ	0h	R: Reserved	Reserved.

Bits	Access	Default	Field	Description
12:8	RO	X	SAGAW: Supported Adjusted Guest Address Widths	This 5-bit field indicates the supported adjusted guest address widths (which in turn represents the levels of page-table walks for the 4KB base page size) supported by the hardware implementation. A value of 1 in any of these bits indicates the corresponding adjusted guest address width is supported. The adjusted guest address widths corresponding to various bit positions within this field are: • 0: Reserved • 1: 39-bit AGAW (3-level page-table) • 2: 48-bit AGAW (4-level page-table) • 3: 57-bit AGAW (5-level page-table) • 4: Reserved Software must ensure that the adjusted guest address width used to set up the page tables is one of the supported guest address widths reported in this field. Hardware implementations with Major Version 6 or higher (VER_REG) reporting the second-stage translation support (SSTS) field as Clear also report this field as 0.
7	RO	X	CM: Caching Mode	This field applies to all DMA and Interrupt remap tables except FS-tables. Hardware will not cache faulting FS-only translations in IOTLB or FS-paging-structure caches. • 0: Not-present and erroneous entries are not cached in any of the remapping caches. Invalidations are not required for modifications to individual not present or invalid entries. However, any modifications that result in decreasing the effective permissions or partial permission increases require invalidations for them to be effective. • 1: Not-present and erroneous mappings may be cached in the remapping caches. Any software updates to the remapping structures (including updates to "not-present" or erroneous entries) require explicit invalidation. Hardware implementations of this architecture must support a value of 0 in this field. Refer to Section 6.1 for more details on Caching Mode.
6	RO	X	PHMR: Protected High-Memory Region	• 0: Indicates protected high-memory region is not supported. • 1: Indicates protected high-memory region is supported.
5	RO	X	PLMR: Protected Low-Memory Region	• 0: Indicates protected low-memory region is not supported. • 1: Indicates protected low-memory region is supported.
4	RO	X	RWBF: Required Write-Buffer Flushing	• 0: Indicates no write-buffer flushing is needed to ensure changes to memory-resident structures are visible to hardware. • 1: Indicates software must explicitly flush the write buffers to ensure updates made to memory-resident remapping structures are visible to hardware. Refer to Section 6.8 for more details on write buffer flushing requirements. Hardware implementations reporting Scalable Mode Translation Support (SMTS) as Set also report this field as Clear.
3	RsvdZ	0h	R: Reserved	• Reserved.

Bits	Access	Default	Field	Description
2:0	RO	X	ND: Number of domains supported[1]	• 000b: Hardware supports 4-bit domain-ids with support for up to 16 domains. • 001b: Hardware supports 6-bit domain-ids with support for up to 64 domains. • 010b: Hardware supports 8-bit domain-ids with support for up to 256 domains. • 011b: Hardware supports 10-bit domain-ids with support for up to 1024 domains. • 100b: Hardware supports 12-bit domain-ids with support for up to 4K domains. • 101b: Hardware supports 14-bit domain-ids with support for up to 16K domains. • 110b: Hardware supports 16-bit domain-ids with support for up to 64K domains. • 111b: Reserved. For hardware implementations reporting HPT Support (HPTS), this field also indicates the number of HPT domains supported.

1. Each remapping unit in the platform should support as many number of domains as the maximum number of independently DMA-remappable devices expected to be attached behind it.

11.4.3 Extended Capability Register

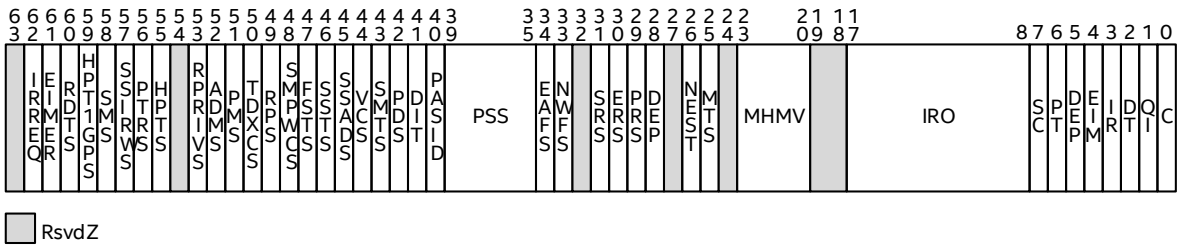


Figure 11-3. Extended Capability Register

Abbreviation	ECAP_REG
General Description	Register to report remapping hardware extended capabilities
Register Offset	010h

Bits	Access	Default	Field	Description
63	RsvdZ	0h	R: Reserved	Reserved.
62	RO	X	IRREQ: Interrupt Remapping Required	• 0: MSI requests received when interrupt remapping is disabled are processed as compatibility format interrupts. • 1: MSI requests received when interrupt remapping is disabled are blocked and reported as an interrupt remapping fault condition. Hardware implementations reporting Interrupt Remapping Support (IR) as Clear also report this field as Clear.
61	RO	X	EIMER: Extended Interrupt Mode Enable Required	• 0: Hardware reports supported interrupt remapping modes through the Extended Interrupt Mode (EIM) field in this register. • 1: Hardware supports operating in Extended Interrupt Mode only. Hardware implementations reporting Extended Interrupt Mode (EIM) as Clear also report this field as Clear.
60	RO	X	RDTS: RDT Configuration Support	• 0: Hardware does not support software control of RMID & CLOS. • 1: Hardware supports software control of RMID & CLOS.
59	RO	X	HPT1GPS: HPT 1G Page Support	• 0: Hardware does not support 1GB Page Size in Host Permission Tables. • 1: Hardware supports 1GB Page Size in Host Permission Tables. Hardware implementations reporting HPT Support (HPTS) as Clear must also report this field as Clear.

Bits	Access	Default	Field	Description
58	RO	X	SMS: Stop Marker Support	Stop Marker Message is defined by PCI Express specification and is used to indicate that function has transmitted all the pending Page Request Messages for a specific PASID. • 0: Remapping hardware does not support Stop Marker Message. If remapping hardware is unable to write a Stop Marker Message into the Page Request Queue (due to the queue being full or a non-recoverable fault), behavior is undefined; otherwise it will write the Stop Marker Message into the Page Request Queue. • 1: Remapping hardware supports Stop Marker Message. If remapping hardware is unable to write a Stop Marker Message into the Page Request Queue (due to the queue being full or a non-recoverable fault), it will silently drop the Stop Marker Message; otherwise it will write the Stop Marker Message into the Page Request Queue. For implementations reporting Stop Marker Support (SMS) as Clear, software is recommended to not enable Page Request Interface on devices that may generate a Stop Marker Message. Hardware implementations reporting Page Request Support (PRS) as Clear also report this field as Clear.
57	RO	X	SSIRWS: Second Stage I/O Read/Write Support	• 0: Hardware does not support I/O Read and I/O Write bits in second stage paging entries • 1: Hardware supports I/O Read and I/O Write bits in second stage paging entries Hardware implementations reporting Second-Level Translation Support (SLTS) as Clear also report this field as Clear.
56	RO	X	PTRS: PASID in Translated Request Support	• 0: Hardware does not support PASID in Translated Requests. • 1: Hardware supports PASID in Translated Requests. Hardware implementations reporting Device-TLB support (DT) as Clear or Process Address Space ID Support (PASID) as Clear also report this field as Clear.
55	RO	X	HPTS: HPT Support	• 0: Hardware does not support host permission tables. • 1: Hardware supports host permission tables. Hardware implementations reporting either Scalable Mode Translation Support (SMTS) or Device TLB Support (DT) as Clear also report this field as Clear.
54	RsvdZ	X	R: Reserved	Reserved.
53	RO	X	RPRIVS: RID_PRIV Support	• 0: Hardware does not support the RID_PRIV field in the scalable-mode context-entry. It uses the value of 0 for RID_PRIV. • 1: Hardware supports the RID_PRIV field in the scalable-mode context-entry. Hardware implementations reporting Supervisor Request Support (SRS) as Clear also report this field as Clear.
52	RO	X	ADMS: Abort DMA Mode Support	• 0: Hardware does not support Abort DMA Mode. • 1: Hardware supports Abort DMA Mode.
51	RO	X	PMS: Performance Monitoring Support	• 0: Hardware does not support Performance Monitoring. • 1: Hardware supports Performance Monitoring. Hardware implementations reporting Enhanced Command Support (ECMDS) as Clear also report this field as Clear.

Bits	Access	Default	Field	Description
50	RO	X	TDXCS: TDX Connect Support	• 0: Hardware does not support TDX Connect Extensions. • 1: Hardware supports TDX Connect Extensions. Refer to documentation of Intel® TDX Connect Architecture Specification for more information.
49	RO	X	RPS: RID-PASID Support	• 0: Hardware does not support RID_PASID field in scalable-mode context-entry. It uses the value of 0 for RID_PASID. • 1: Hardware supports the RID_PASID field in scalable-mode context-entry. Hardware implementations reporting Scalable Mode Translation Support (SMTS) as Clear also report this field as Clear.
48	RO	X	SMPWCS: Scalable-Mode Page-walk Coherency Support	• 0: Hardware access to paging structures accessed through PASID-table entry are not snooped. • 1: Hardware access to paging structures accessed through PASID-table entry are snooped according to the PWSNP field in the scalable-mode PASID-table entry. Hardware implementations reporting Scalable Mode Translation Support (SMTS) as Clear also report this field as Clear. see Section 3.10 for additional details.
47	RO	X	FSTS: First-stage Translation Support	• 0: Hardware does not support PASID Granular Translation Type of first-stage (PGTT=001b) in scalable-mode PASID-Table entry. • 1: Hardware supports PASID Granular Translation Type of first-stage (PGTT=001b) in scalable-mode PASID-Table entry. Hardware implementations reporting Scalable Mode Translation Support (SMTS) as Clear also report this field as Clear.
46	RO	X	SSTS: Second-stage Translation Support	• 0: Hardware does not support PASID Granular Translation Type of second-stage (PGTT=010b) in scalable-mode PASID-Table entry. • 1: Hardware supports PASID Granular Translation Type of second-stage (PGTT=010b) in scalable-mode PASID-Table entry.
45	RO	X	SSADS: Second-Stage Accessed/Dirty Support	• 0: Hardware does not support Accessed/Dirty bits in Second-Stage translation. • 1: Hardware supports Accessed/Dirty bits in Second-Stage translation. Hardware implementations reporting Scalable-Mode Page-walk Coherency Support (SMPWCS) as Clear also report this field as Clear.
44	RO	X	VCS: Virtual Command Support	• 0: Hardware does not support command submission to virtual-DMA Remapping hardware. • 1: Hardware does support command submission to virtual-DMA Remapping hardware. Hardware implementations of this architecture report a value of 0 in this field. Software implementations (emulation) of this architecture may report VCS=1. Software managing remapping hardware should be written to handle both values of VCS. Refer to Section 11.4.16.2 for more details on Virtual Commands.
43	RO	X	SMTS: Scalable Mode Translation Support	• 0: Hardware does not support Scalable Mode DMA Remapping. • 1: Hardware supports Scalable Mode DMA Remapping through scalable-mode context-table and PASID-table structures. Hardware implementation reporting Queued Invalidation (QI) field as Clear also report this field as Clear.

Bits	Access	Default	Field	Description
42	RO	X	PDS: Page-request Drain Support	• 0: Hardware does not support Page-request Drain (PD) flag in *inv_wait_dsc*. • 1: Hardware supports Page-request Drain (PD) flag in *inv_wait_dsc*. Hardware implementations reporting Device-TLB support (DT) as Clear also report this field as Clear.
41	RO	X	DIT: Device-TLB Invalidation Throttle	• 0: Hardware does not support Device-TLB Invalidation Throttling. • 1: Hardware supports Device-TLB Invalidation Throttling. Hardware implementations reporting Device-TLB support (DT) as Clear also report this field as Clear.
40	RO	X	PASID: Process Address Space ID Support	• 0: Hardware does not support requests tagged with Process Address Space IDs. • 1: Hardware supports requests tagged with Process Address Space IDs. Hardware implementations reporting Scalable Mode Translation Support (SMTS) field as Clear also report this field as Clear.
39:35	RO	X	PSS: PASID Size Supported	This field reports the PASID size supported by the remapping hardware for requests-with-PASID. A value of N in this field indicates hardware supports PASID field of N+1 bits (For example, value of 7 in this field, indicates 8-bit PASIDs are supported). Requests-with-PASID with PASID value beyond the limit specified by this field are treated as error by the remapping hardware. This field is unused and reported as 0 if Scalable Mode Translation Support (SMTS) field is Clear.
34	RO	X	EAFS: Extended Accessed Flag Support	• 0: Hardware does not support the extended-accessed (EA) bit in first-stage paging-structure entries. • 1: Hardware supports the extended-accessed (EA) bit in first-stage paging-structure entries. Hardware implementations reporting First Stage Translation Support (FSTS) or Scalable-Mode Page-walk Coherency Support (SWPWCS) as Clear also report this field as Clear.
33	RO	X	NWFS: No Write Flag Support	• 0: Hardware ignores the 'No Write' (NW) flag in Device-TLB translation-requests, and behaves as if NW is always 0. • 1: Hardware supports the 'No Write' (NW) flag in Device-TLB translation-requests. Hardware implementations reporting Device-TLB support (DT) field as Clear also report this field as Clear.
32	RsvdZ	0h	R: Reserved	Reserved.
31	RO	X	SRS: Supervisor Request Support	• 0: Hardware does not support requests (with or without PASID) seeking supervisor privilege. • 1: Hardware supports requests (with or without PASID) seeking supervisor privilege. Hardware implementations reporting Scalable Mode Translation Support (SMTS) field as Clear also report this field as Clear.
30	RO	0h	ERS: Execute Request Support	This field is planned for deprecation. Implementations must report this field as Clear to indicate that the remapping unit does not support requests-with-PASID that have a value of 1 in the Execute-Requested (ER) field.
29	RO	X	PRS: Page Request Support	• 0: Hardware does not support page requests. • 1: Hardware supports page requests. Hardware implementation reporting Device-TLB support (DT) field as Clear or Scalable Mode Translation Support (SMTS) field as Clear also report this field as Clear.

Bits	Access	Default	Field	Description
28	RsvdZ	0h	DEP: Deprecated	Deprecated. This field must be reported as 0 to ensure backward compatibility with older software.
27	RsvdZ	0h	R: Reserved	Reserved.
26	RO	X	NEST: Nested Translation Support	• 0: Hardware does not support nested translations. • 1: Hardware supports nested translations. Hardware implementations reporting Scalable Mode Translation Support (SMTS) field as Clear or First-stage Translation Support (FSTS) field as Clear or Second-stage Translation Support (SSTS) field as Clear also report this field as Clear.
25	RO	X	MTS: Memory Type Support	• 0: Hardware does not support Memory Type in first-stage translation and Extended Memory type in second-stage translation. • 1: Hardware supports Memory Type in first-stage translation and Extended Memory type in second-stage translation. Hardware implementations reporting Scalable Mode Translation Support (SMTS) field as Clear also report this field as Clear. Remapping hardware units with, one or more devices that operate in processor coherency domain, under its scope must report this field as Set.
24	RsvdZ	0h	DEP: Deprecated	In prior versions of this specification this bit was used to enumerate "Extended mode address translation" which is now deprecated. This field must be reported as 0 to ensure backward compatibility with any software that enables extended mode address translation.
23:20	RO	X	MHMV: Maximum Handle Mask Value	The value in this field indicates the maximum supported value for the Interrupt Mask (IM) field in the Interrupt Entry Cache Invalidation Descriptor (*iec_inv_dsc*). This field is unused and is reported as 0 if Interrupt Remapping support (IR) field is Clear.
19:18	RsvdZ	0h	R: Reserved	Reserved.
17:8	RO	X	IRO: IOTLB Register Offset	This field specifies the offset to the IOTLB registers relative to the register base address of this remapping hardware unit. If the register base address is X, and the value reported in this field is Y, the address for the IOTLB registers is calculated as X+(16*Y).
7	R0	X	SC: Snoop Control	• 0: Hardware does not support 1-setting of the SNP field in the second-stage page-table entries and the PGSNP field in the scalable-mode PASID-table entries. • 1: Hardware supports the 1-setting of the SNP field in the second-stage page-table entries and the PGSNP field in the scalable-mode PASID-table entries. Implementations are recommended to support Snoop Control to support software usages that require Snoop Control for assignment of devices behind a remapping hardware unit.
6	R0	X	PT: Pass Through	• 0: Hardware does not support pass-through translation type in context-entries and scalable-mode-pasid-table-entries. • 1: Hardware supports pass-through translation type in context and scalable-mode-pasid-table-entries.
5	R0	0h	DEP: Deprecated	Deprecated. This field must be reported as 0 to ensure backward compatibility with older software.

Bits	Access	Default	Field	Description
4	R0	X	EIM: Extended Interrupt Mode	• 0: On Intel® 64 platforms, hardware supports only 8-bit APIC-IDs (xAPIC Mode). • 1: On Intel® 64 platforms, hardware supports 32-bit APIC-IDs (x2APIC mode). Hardware implementation reporting Interrupt Remapping support (IR) field as Clear also report this field as Clear.
3	RO	X	IR: Interrupt Remapping support	• 0: Hardware does not support interrupt remapping. • 1: Hardware supports interrupt remapping. Hardware implementation reporting Queued Invalidation support (QI) field as Clear also report this field as Clear.
2	RO	X	DT: Device-TLB support	• 0: Hardware does not support Device-TLBs. • 1: Hardware supports Device-TLBs. Hardware implementation reporting Queued Invalidation support (QI) field as Clear also report this field as Clear.
1	RO	X	QI: Queued Invalidation support	• 0: Hardware does not support queued invalidations. • 1: Hardware supports queued invalidations.
0	RO	X	C: Page-walk Coherency	This field indicates if hardware access to the root, scalable-mode root, context, scalable-mode-context, scalable-mode PASID-directory, scalable-mode PASID-table, and interrupt-remap tables, and legacy-mode second-stage paging structures are coherent (snooped) or not. • 0:Indicates hardware accesses to remapping structures are non-coherent. • 1:Indicates hardware accesses to remapping structures are coherent. Hardware access to invalidation queue, invalidation wait descriptor completion status address, page-request queue, and Posted Interrupt Descriptor are always snooped. See Scalable-Mode Page-walk Coherency (SMPWC) field for hardware behavior on paging structures accessed through scalable-mode PASID-table entry.

11.4.4 Global Command Interface Registers

11.4.4.1 Global Command Register

Figure 11-4. Global Command Register

Abbreviation	GCMD_REG
General Description	Register to control remapping hardware. Behavior is undefined if this register is written while a command is in progress on the Enhanced Command Interface (ECRESP_REG.IP=1). If multiple control fields in this register need to be modified, software must serialize the modifications through multiple writes to this register. For example, to update a bit field in this register at offset X with value of Y, software must follow below steps: 1. Tmp = Read GSTS_REG 2. Status = (Tmp & 96FFFFFFh) // Reset the one-shot bits 3. if (Y) {Command = (Status | (1 << X))} else {Command = (Status & ~(1 << X))} 4. Write Command to GCMD_REG 5. Wait until GSTS_REG[X] indicates command is serviced.
Register Offset	018h

Bits	Access	Default	Field	Description
31	WO	0	TE: Translation Enable	Software writes to this field to request hardware to enable/disable DMA remapping: • 0: Disable DMA remapping • 1: Enable DMA remapping Hardware reports the status of the translation enable operation through the TES field in the Global Status register. There may be active DMA requests in the platform when software updates this field. Hardware must enable or disable remapping logic only at deterministic transaction boundaries, so that any in-flight transaction is either subject to remapping or not at all. Hardware implementations supporting DMA draining must drain any in-flight DMA read/write requests queued within the Root-Complex before completing the translation enable command and reflecting the status of the command through the TES field in the Global Status register. For implementations reporting Scalable Mode Translation Support (SMTS) field as Set, hardware performs global invalidation on all DMA remapping translation caches as part of Translation Disable operation. The value returned on a read of this field is undefined.

Bits	Access	Default	Field	Description
30	WO	0	SRTP: Set Root Table Pointer	Software sets this field to set/update the root-table pointer (and translation table mode) used by hardware. The root-table pointer (and translation table mode) is specified through the Root Table Address (RTADDR_REG) register. Hardware reports the status of the 'Set Root Table Pointer' operation through the RTPS field in the Global Status register. The 'Set Root Table Pointer' operation must be performed before enabling or re-enabling (after disabling) DMA remapping through the TE field. For details on invalidation that software may have to perform after the 'Set Root Table Pointer' operation refer to Section 6.6. Clearing this bit has no effect. The value returned on a read of this field is undefined.
29:28	RsvdZ	0h	R: Reserved	Reserved.
27	WO	0	WBF: Write Buffer Flush[1]	This bit is valid only for implementations requiring write buffer flushing. Software sets this field to request that hardware flush the Root-Complex internal write buffers. This is done to ensure any updates to the memory-resident remapping structures are not held in any internal write posting buffers. Refer to Section 6.8 for details on write-buffer flushing requirements. Hardware reports the status of the write buffer flushing operation through the WBFS field in the Global Status register. Clearing this bit has no effect. The value returned on a read of this field is undefined.
26	WO	0	QIE: Queued Invalidation Enable	This field is valid only for implementations supporting queued invalidations. Software writes to this field to enable or disable queued invalidations. • 0: Disable queued invalidations. • 1: Enable use of queued invalidations. Hardware reports the status of queued invalidation enable operation through QIES field in the Global Status register. Refer to Section 6.5.2 for software requirements for enabling/disabling queued invalidations. The value returned on a read of this field is undefined.
25	WO	0h	IRE: Interrupt Remapping Enable	This field is valid only for implementations supporting interrupt remapping. • 0: Disable interrupt-remapping hardware • 1: Enable interrupt-remapping hardware Hardware reports the status of the interrupt remapping enable operation through the IRES field in the Global Status register. There may be active interrupt requests in the platform when software updates this field. Hardware must enable or disable interrupt-remapping logic only at deterministic transaction boundaries, so that any in-flight interrupts are either subject to remapping or not at all. For implementations reporting the Enhanced Set Interrupt Remap Table Pointer Support (ESIRTPS) field as Set, hardware performs global invalidation on all Interrupt remapping caches as part of Interrupt Remapping Disable operation. Hardware implementations must drain any in-flight interrupts requests queued in the Root-Complex before completing the interrupt-remapping enable command and reflecting the status of the command through the IRES field in the Global Status register. The value returned on a read of this field is undefined.

Bits	Access	Default	Field	Description
24	WO	0	SIRTP: Set Interrupt Remap Table Pointer	This field is valid only for implementations supporting interrupt-remapping. Software sets this field to set/update the interrupt remapping table pointer used by hardware. The interrupt remapping table pointer is specified through the Interrupt Remapping Table Address (IRTA_REG) register. Hardware reports the status of the 'Set Interrupt Remap Table Pointer' operation through the IRTPS field in the Global Status register. The 'Set Interrupt Remap Table Pointer' operation must be performed before enabling or re-enabling (after disabling) interrupt-remapping hardware through the IRE field. For details on invalidation that software may have to perform after the 'Set Interrupt Remap Table Pointer' operation, refer to Section 6.6. Clearing this bit has no effect. The value returned on a read of this field is undefined.
23	WO	0	CFI: Compatibility Format Interrupt	This field is valid only for Intel® 64 implementations supporting interrupt-remapping. Software writes to this field to enable or disable Compatibility Format interrupts on Intel® 64 platforms. The value in this field is effective only when interrupt-remapping is enabled and Extended Interrupt Mode (x2APIC mode) is not enabled. • 0: Block Compatibility format interrupts. • 1: Process Compatibility format interrupts as pass-through (bypass interrupt remapping). Hardware reports the status of updating this field through the CFIS field in the Global Status register. Refer to Section 5.1.2.1 for details on Compatibility Format interrupt requests. The value returned on a read of this field is undefined.
22:0	RsvdZ	0h	R: Reserved	Reserved.

1. Implementations reporting write-buffer flushing as required in Capability register must perform implicit write buffer flushing as a pre-condition to all context-cache and IOTLB invalidation operations.

11.4.4.2 Global Status Register

Figure 11-5. Global Status Register

Abbreviation	GSTS_REG
General Description	Register to report general remapping hardware status.
Register Offset	01Ch

Bits	Access	Default	Field	Description
31	RO	0	TES: Translation Enable Status	This field indicates the status of DMA-remapping hardware. • 0: DMA remapping is not enabled • 1: DMA remapping is enabled
30	RO	0	RTPS: Root Table Pointer Status	This field indicates the status of the root-table pointer in hardware. This field is cleared by hardware when software sets the SRTP field in the Global Command register. This field is set by hardware when hardware completes the 'Set Root Table Pointer' operation using the value provided in the Root Table Address register.
29:28	RsvdZ	0h	R: Reserved	• Reserved.
27	RO	0	WBFS: Write Buffer Flush Status	This field is valid only for implementations requiring write buffer flushing. This field indicates the status of the write buffer flush command. It is • Set by hardware when software sets the WBF field in the Global Command register. • Cleared by hardware when hardware completes the write buffer flushing operation.
26	RO	0	QIES: Queued Invalidation Enable Status	This field indicates queued invalidation enable status. • 0: queued invalidation is not enabled • 1: queued invalidation is enabled
25	RO	0	IRES: Interrupt Remapping Enable Status	This field indicates the status of Interrupt-remapping hardware. • 0: Interrupt-remapping hardware is not enabled • 1: Interrupt-remapping hardware is enabled
24	RO	0	IRTPS: Interrupt Remapping Table Pointer Status	This field indicates the status of the interrupt remapping table pointer in hardware. This field is cleared by hardware when software sets the SIRTP field in the Global Command register. This field is Set by hardware when hardware completes the 'Set Interrupt Remap Table Pointer' operation using the value provided in the Interrupt Remapping Table Address register.

Bits	Access	Default	Field	Description
23	RO	0	CFIS: Compatibility Format Interrupt Status	This field indicates the status of Compatibility format interrupts on Intel® 64 implementations supporting interrupt-remapping. The value reported in this field is applicable only when interrupt-remapping is enabled and extended interrupt mode (x2APIC mode) is not enabled. • 0: Compatibility format interrupts are blocked. • 1: Compatibility format interrupts are processed as pass-through (bypassing interrupt remapping).
22:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.5 Root Table Address Register

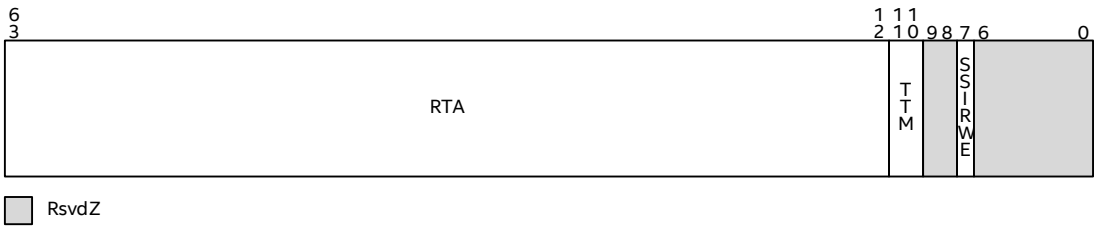


Figure 11-6. Root Table Address Register

Abbreviation	RTADDR_REG
General Description	Register providing the base address of root-table and the translation table mode. Software programs the desired values in this register but these values take effect only after software executes Set Root Table Pointer command through the SRTP field in the Global Command Register (GCMD_REG).
Register Offset	020h

Bits	Access	Default	Field	Description
63:12	RW	0h	RTA: Root Table Address	This field points to the base of the page-aligned, 4KB-sized root-table in system memory. Hardware may ignore and not implement bits 63:HAW, where HAW is the host address width. The value of this field takes effect only after software executes Set Root Table Pointer command. Software is allowed to modify this field while DMA remapping is active (TES=1 in Global Status register) independent of the value of Enhanced SRTP Support (ESRTPS) field in the Capability register.
11:10	RW	0	TTM: Translation Table Mode	This field specifies the translation mode used for DMA remapping. • 00: legacy mode - uses root tables and context tables. • 01: scalable mode - uses scalable-mode root tables and scalable-mode context tables. • 10: reserved - in prior version of this specification, this encoding was used to enable extended mode which is no longer supported. • 11: abort-dma mode. The value of this field takes effect only after software executes Set Root Table Pointer command. Software is allowed to modify this field while DMA remapping is active (TES=1 in Global Status register) only for implementations reporting Enhanced SRTP Support (ESRTPS) field as Set in the Capability register.
9:8	RsvdZ	0h	R: Reserved	Reserved.

Bits	Access	Default	Field	Description
7	RW	0h	SSIRWE: Second Stage I/O Read/Write Enable	This field is treated as Reserved(0) for implementations not supporting Second Stage I/O Read/Write Support. (SSIRWS field reported as 0 in Extended Capability register) • 0: Hardware uses bit 0 (R) and bit 1 (W) in second-stage paging entries to calculate effective permissions. • 1: Hardware uses bit 61 (IR) and bit 62 (IW) in second-stage paging entries to calculate effective permission. The value of this field takes effect only after software executes Set Root Table Pointer command. Software is allowed to modify this field while DMA remapping is active (TES=1 in Global Status register) only for implementations reporting Enhanced SRTP Support (ESRTPS) field as Set in the Capability register.
6:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.6 Register Based Invalidation Interface

11.4.6.1 Context Command Register

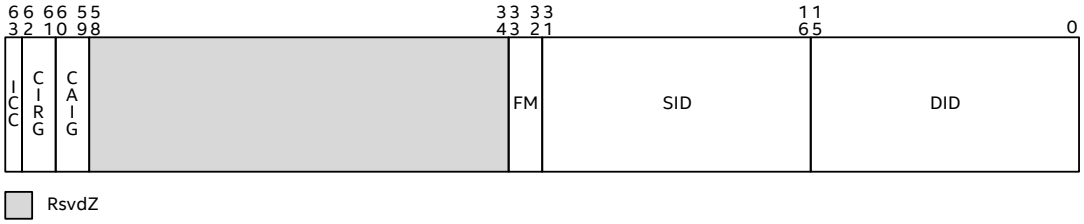


Figure 11-7. Context Command Register

Abbreviation	CCMD_REG
General Description	Register to manage context-cache.The act of writing the uppermost byte of the CCMD_REG with the ICC field Set causes the hardware to perform the context-cache invalidation.
Register Offset	028h

Bits	Access	Default	Field	Description
63	RW	0	ICC: Invalidate Context-Cache	Software requests invalidation of context-cache by setting this field. Software must also set the requested invalidation granularity by programming the CIRG field. Software must read back and check the ICC field is Clear to confirm the invalidation is complete. Software must not update this register when this field is Set. Hardware clears the ICC field to indicate the invalidation request is complete.Hardware also indicates the granularity at which the invalidation operation was performed through the CAIG field. Software must submit a context-cache invalidation request through this field only when there are no invalidation requests pending at this remapping hardware unit. Since information from the context-cache may be used by hardware to tag IOTLB entries, software must perform domain-selective (or global) invalidation of IOTLB after the context-cache invalidation has completed. Hardware implementations reporting a write-buffer flushing requirement (RWBF=1 in the Capability register) must implicitly perform a write buffer flush before invalidating the context-cache. Refer to Section 6.8 for write buffer flushing requirements. When Translation Table Mode field in Root Table Address register is not setup as legacy mode (RTADDR_REG.TTM!=00b), hardware will ignore the value provided by software in this register, treat it as an incorrect invalidation request, and report a value of 00b in CAIG field.

Bits	Access	Default	Field	Description
62:61	RW	0h	CIRG: Context Invalidation Request Granularity	Software provides the requested invalidation granularity through this field when setting the ICC field: • 00: Reserved. • 01: Global Invalidation request. • 10: Domain-selective invalidation request. The target domain-id must be specified in the DID field. • 11: Device-selective invalidation request. The target source-id(s) must be specified through the SID and FM fields, and the domain-id [that was programmed in the context-entry for these device(s)] must be provided in the DID field. Hardware implementations may process an invalidation request by performing invalidation at a coarser granularity than requested. Hardware indicates completion of the invalidation request by clearing the ICC field. At this time, hardware also indicates the granularity at which the actual invalidation was performed through the CAIG field.
60:59	RO	Xh	CAIG: Context Actual Invalidation Granularity	Hardware reports the granularity at which an invalidation request was processed through the CAIG field at the time of reporting invalidation completion (by clearing the ICC field). The following are the encodings for this field: • 00: Error. This indicates hardware detected an incorrect invalidation request and ignored the request, e.g., register based invalidation when Translation Table Mode (TTM) in Root Table Address Register is not programmed to legacy mode (RTADDR_REG.TTM!=00b). On hardware implementations with Major Version 6 or higher (VER_REG), all invalidation requests through this register are treated as incorrect invalidation requests. Software should use the Queued Invalidation interface to perform context-cache invalidations for such hardware implementations. Refer to Section 6.5 for more details. • 01: Global Invalidation performed. This could be in response to a global, domain-selective, or device-selective invalidation request. • 10: Domain-selective invalidation performed using the domain-id specified by software in the DID field. This could be in response to a domain-selective or device-selective invalidation request. • 11: Device-selective invalidation performed using the source-id and domain-id specified by software in the SID and FM fields. This can only be in response to a device-selective invalidation request.
58:34	RsvdZ	0h	R: Reserved	Reserved.
33:32	WO	0h	FM: Function Mask	Software may use the Function Mask to perform device-selective invalidations on behalf of devices supporting PCI Express Phantom Functions. This field specifies which bits of the function number portion (least significant three bits) of the SID field to mask when performing device-selective invalidations.The following encodings are defined for this field: • 00: No bits in the SID field masked • 01: Mask bit 2 in the SID field • 10: Mask bits 2:1 in the SID field • 11: Mask bits 2:0 in the SID field The context-entries corresponding to the source-ids specified through the SID and FM fields must have the domain-id specified in the DID field. The value returned on a read of this field is undefined.
31:16	WO	0h	SID: Source-ID	Indicates the source-id of the device whose corresponding context-entry needs to be selectively invalidated.This field along with the FM field must be programmed by software for device-selective invalidation requests. The value returned on a read of this field is undefined.

Bits	Access	Default	Field	Description
15:0	RW	0h	DID: Domain-ID	Indicates the id of the domain whose context-entries need to be selectively invalidated. This field must be programmed by software for both domain-selective and device-selective invalidation requests. The Capability register reports the domain-id width supported by hardware. Software must ensure that the value written to this field is within this limit. Hardware ignores (and may not implement) bits 15:N, where N is the supported domain-id width reported in the Capability register.

11.4.6.2 IOTLB Registers

IOTLB registers consists of two adjacently placed 64-bit registers:

- IOTLB Invalidate Register (IOTLB_REG)
- Invalidate Address Register (IVA_REG)

Offset	Register Name	Size	Description
XXXh	Invalidate Address Register	64	Register to provide the target address for page-selective IOTLB invalidation. The offset of this register is reported through the IRO field in Extended Capability register.
XXXh + 008h	IOTLB Invalidate Register	64	Register for IOTLB invalidation command

These registers are described in the following sections.

11.4.6.3 IOTLB Invalidate Register

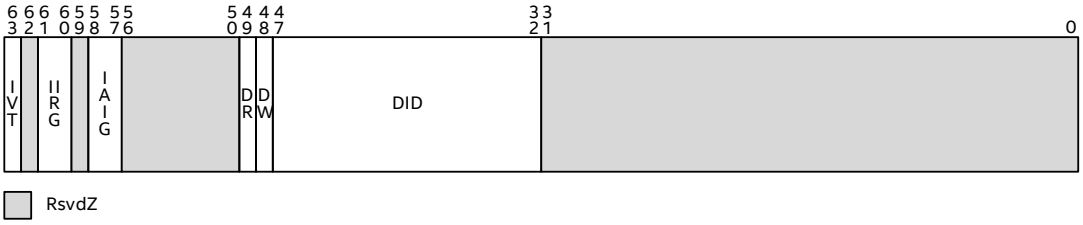


Figure 11-8. IOTLB Invalidate Register

Abbreviation	IOTLB_REG
General Description	Register to invalidate IOTLB. The act of writing the upper byte of the IOTLB_REG with the IVT field Set causes the hardware to perform the IOTLB invalidation.
Register Offset	XXXh + 008h (where XXXh is the location of the IVA_REG)

Bits	Access	Default	Field	Description
63	RW	0	IVT: Invalidate IOTLB	Software requests IOTLB invalidation by setting this field. Software must also set the requested invalidation granularity by programming the IIRG field. Hardware clears the IVT field to indicate the invalidation request is complete. Hardware also indicates the granularity at which the invalidation operation was performed through the IAIG field. Software must not submit another invalidation request through this register while the IVT field is Set, nor update the associated Invalidate Address register. Software must not submit IOTLB invalidation requests when there is a context-cache invalidation request pending at this remapping hardware unit. Hardware implementations reporting a write-buffer flushing requirement (RWBF=1 in Capability register) must implicitly perform a write buffer flushing before invalidating the IOTLB. Refer to Section 6.8 for write buffer flushing requirements. When Translation Table Mode field in Root Table Address registers is not setup as legacy mode (RTADDR_REG.TTM!=00b), hardware will ignore the value provided by software in this register, treat it as an incorrect invalidation request, and report a value of 00b in IAIG field.
62	RsvdZ	0	R: Reserved	Reserved.

Bits	Access	Default	Field	Description
61:60	RW	0h	IIRG: IOTLB Invalidation Request Granularity	When requesting hardware to invalidate the IOTLB (by setting the IVT field), software writes the requested invalidation granularity through this field. The following are the encodings for the field. • 00: Reserved. • 01: Global invalidation request. • 10: Domain-selective invalidation request. The target domain-id must be specified in the DID field. • 11: Page-selective-within-domain invalidation request. The target address, mask, and invalidation hint must be specified in the Invalidate Address register, and the domain-id must be provided in the DID field. Hardware implementations may process an invalidation request by performing invalidation at a coarser granularity than requested. Hardware indicates completion of the invalidation request by clearing the IVT field. At that time, the granularity at which actual invalidation was performed is reported through the IAIG field.
59	RsvdZ	0	R: Reserved	Reserved.
58:57	RO	Xh	IAIG: IOTLB Actual Invalidation Granularity	Hardware reports the granularity at which an invalidation request was processed through this field when reporting invalidation completion (by clearing the IVT field). The following are the encodings for this field. • 00: Error. This indicates hardware detected an incorrect invalidation request and ignored the request, e.g., register based invalidation when Translation Table Mode (TTM) in Root Table Address Register is not programmed to legacy mode (RTADDR_REG.TTM!=00b), detected an unsupported address mask value in Invalidate Address register for page-selective invalidation requests. On hardware implementations with Major Version 6 or higher (VER_REG), all invalidation requests through this register are treated as incorrect invalidation requests. Software should use the Queued Invalidation interface to perform IOTLB invalidations for such hardware implementations. Refer to Section 6.5 for more details. • 01: Global Invalidation performed. This could be in response to a global, domain-selective, or page-selective invalidation request. • 10: Domain-selective invalidation performed using the domain-id specified by software in the DID field. This could be in response to a domain-selective or a page-selective invalidation request. • 11: Page-selective-within-domain invalidation performed using the address, mask and hint specified by software in the Invalidate Address register and domain-id specified in DID field. This can be in response to a page-selective-within-domain invalidation request.
56:50	RsvdZ	0h	R: Reserved	Reserved.
49	RW	0h	DR: Drain Reads	This field is ignored by hardware if the DRD field is reported as Clear in the Capability register. When the DRD field is reported as Set in the Capability register, the following encodings are supported for this field: • 0: Hardware may complete the IOTLB invalidation without draining DMA read requests. • 1: Hardware must drain DMA read requests. Refer Section 6.5.4 for description of DMA draining.
48	RW	0h	DW: Drain Writes	This field is ignored by hardware if the DWD field is reported as Clear in the Capability register. When the DWD field is reported as Set in the Capability register, the following encodings are supported for this field: • 0: Hardware may complete the IOTLB invalidation without draining DMA write requests. • 1: Hardware must drain relevant translated DMA write requests. Refer Section 6.5.4 for description of DMA draining.

Bits	Access	Default	Field	Description
47:32	RW	0h	DID: Domain-ID	Indicates the ID of the domain whose IOTLB entries need to be selectively invalidated. This field must be programmed by software for domain-selective and page-selective invalidation requests. The Capability register reports the domain-id width supported by hardware. Software must ensure that the value written to this field is within this limit. Hardware may ignore and not implement bits 47:(32+N), where N is the supported domain-id width reported in the Capability register.
31:0	RsvdP	Xh	R: Reserved	Reserved.

11.4.6.4 Invalidate Address Register

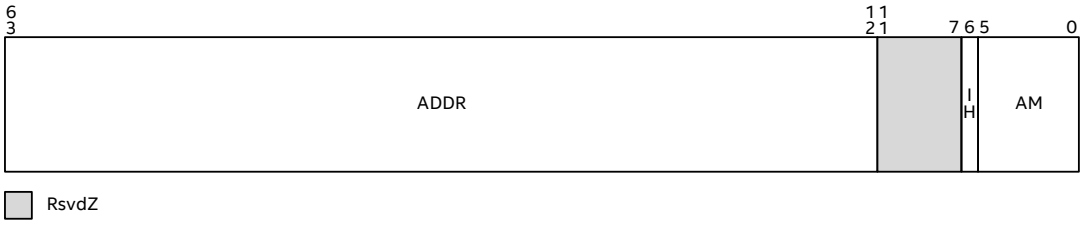


Figure 11-9. Invalidate Address Register

Abbreviation	IVA_REG
General Description	Register to provide the DMA address whose corresponding IOTLB entry needs to be invalidated through the corresponding IOTLB Invalidate register. This register is a write-only register. A value returned on a read of this register is undefined.
Register Offset	XXXh (XXXh is QWORD aligned and reported through the IRO field in the Extended Capability register)

Bits	Access	Default	Field	Description
63:12	WO	0h	ADDR: Address	Software provides the second-stage-input-address that needs to be page-selectively invalidated. To make a page-selective-within-domain invalidation request to hardware, software must first write the appropriate fields in this register, and then issue the page-selective-within-domain invalidate command through the IOTLB_REG. Hardware ignores bits 63:N, where N is the maximum guest address width (MGAW) supported. A value returned on a read of this field is undefined.
11:7	RsvdZ	0	R: Reserved	Reserved.
6	WO	0	IH: Invalidation Hint	The field provides hints to hardware about preserving or flushing the non-leaf (context-entry) entries that may be cached in hardware: • 0: Software may have modified both leaf and non-leaf second-stage paging-structure entries corresponding to mappings specified in the ADDR and AM fields. On a page-selective-within-domain invalidation request, hardware must invalidate the cached entries associated with the mappings specified by DID, ADDR and AM fields, in both IOTLB and paging-structure caches. Refer to Section 6.5.1.2 for exact invalidation requirements when IH=0. • 1: Software has not modified any second-stage non-leaf paging entries associated with the mappings specified by the ADDR and AM fields. On a page-selective-within-domain invalidation request, hardware may preserve the cached second-stage mappings in paging-structure-caches. Refer to Section 6.5.1.2 for exact invalidation requirements when IH=1. A value returned on a read of this field is undefined.

<table>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Field</th><th>Description</th></tr>
<tr><td>5:0</td><td>WO</td><td>0</td><td>AM: Address Mask</td><td>The value in this field specifies the number of low order bits of the ADDR field that must be masked for the invalidation operation. This field enables software to request invalidation of contiguous mappings for size-aligned regions. For example:

<table><tr><th>Mask Value</th><th>ADDR bits masked</th><th>Pages invalidated</th></tr><tr><td>0</td><td>None</td><td>1</td></tr><tr><td>1</td><td>12</td><td>2</td></tr><tr><td>2</td><td>13:12</td><td>4</td></tr><tr><td>3</td><td>14:12</td><td>8</td></tr><tr><td>4</td><td>15:12</td><td>16</td></tr><tr><td>...</td><td>...</td><td>...</td></tr></table>
When invalidating mappings for large-pages, software must specify the appropriate mask value. For example, when invalidating mapping for a 2MB page, software must specify an address mask value of at least 9.

Hardware implementations report the maximum supported address mask value through the Capability register.

A value returned on a read of this field is undefined.</td></tr>
</table>

11.4.7 Fault Reporting Interface

11.4.7.1 Fault Status Register

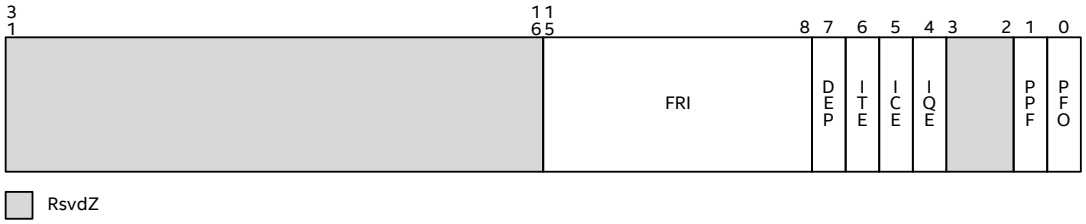


Figure 11-10. Fault Status Register

Abbreviation	FSTS_REG
General Description	Register indicating the various error status.
Register Offset	034h

Bits	Access	Default	Field	Description
31:16	RsvdZ	0h	R: Reserved	Reserved.
15:8	ROS	0	FRI: Fault Record Index	This field is valid only when the PPF field is Set. The FRI field indicates the index (from base) of the fault recording register to which the first pending fault was recorded when the PPF field was Set by hardware. The value read from this field is undefined when the PPF field is Clear.
7	RsvdZ	0	DEP: Deprecated	This bit has been deprecated and is treated as RsvdZ. Previous versions of this specification had PRO in this location which has been moved to Page Request Status register.
6	RW1CS	0h	ITE: Invalidation Time-out Error	Hardware detected a Device-TLB invalidation completion time-out. At this time, a fault event may be generated based on the programming of the Fault Event Control register. Refer to the description of the ITESID field in Section 11.4.9.9 for the requester-id associated with the Invalidation Time-out error. Hardware implementations not supporting Device-TLBs implement this bit as RsvdZ.
5	RW1CS	0h	ICE: Invalidation Completion Error	Hardware received an unexpected or invalid Device-TLB invalidation completion. This could be due to either an invalid ITag or invalid source-id in an invalidation completion response. At this time, a fault event may be generated based on the programming of the Fault Event Control register. Refer to the description of the ICESID field in Section 11.4.9.9 for the requester-id associated with the Invalidation Completion error. Hardware implementations not supporting Device-TLBs implement this bit as RsvdZ.

Bits	Access	Default	Field	Description
4	RW1CS	0	IQE: Invalidation Queue Error	Hardware detected an error associated with the invalidation queue. Refer to the description of the IQEI field in Section 11.4.9.9 for all possible conditions resulting in an Invalidation Queue Error. At this time, a fault event may be generated based on the programming of the Fault Event Control register. Hardware implementations not supporting queued invalidations implement this bit as RsvdZ.
3:2	RsvdZ	0h	R: Reserved	Reserved.
1	ROS	0	PPF: Primary Pending Fault	This field indicates if there are one or more pending faults logged in the fault recording registers. Hardware computes this field as the logical OR of Fault (F) fields across all the fault recording registers of this remapping hardware unit. • 0: No pending faults in any of the fault recording registers • 1: One or more fault recording registers has pending faults. The FRI field is updated by hardware whenever the PPF field is Set by hardware. Also, depending on the programming of Fault Event Control register, a fault event is generated when hardware sets this field.
0	RW1CS	0	PFO: Primary Fault Overflow	Hardware sets this field to indicate overflow of the fault recording registers. When this field is Set, hardware does not record any new faults until software clears this field.

11.4.7.2 Fault Event Control Register

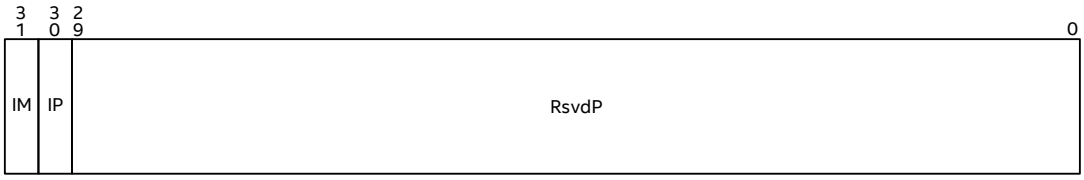


Figure 11-11. Fault Event Control Register

Abbreviation	FECTL_REG
General Description	Register specifying the fault event interrupt message control bits. Section 7.3 describes hardware handling of fault events.
Register Offset	038h

Bits	Access	Default	Field	Description
31	RW	1	IM: Interrupt Mask	• 0: No masking of interrupts. When a interrupt condition is detected, hardware issues an interrupt message (using the Fault Event Data and Fault Event Address register values). • 1: This is the value on reset. Software may mask interrupt message generation by setting this field.Hardware is prohibited from sending the interrupt message when this field is Set.

30	RO	0	IP: Interrupt Pending	Hardware sets the IP field whenever it detects an interrupt condition, which is defined as:
				• When primary fault logging is active, an interrupt condition occurs when hardware records a fault through one of the Fault Recording registers and sets the PPF field in the Fault Status register.
				• Hardware detected error associated with the Invalidation Queue, setting the IQE field in the Fault Status register.
				• Hardware detected invalid Device-TLB invalidation completion, setting the ICE field in the Fault Status register.
				• Hardware detected Device-TLB invalidation completion time-out, setting the ITE field in the Fault Status register.
				If any of the status fields in the Fault Status register was already Set at the time of setting any of these fields, it is not treated as a new interrupt condition.
				
				The IP field is kept Set by hardware while the interrupt message is held pending. The interrupt message could be held pending due to the interrupt mask (IM field) being Set or other transient hardware conditions.
				
				The IP field is cleared by hardware as soon as the interrupt message pending condition is serviced. This could be due to either:
				• Hardware issuing the interrupt message due to either a change in the transient hardware condition that caused the interrupt message to be held pending, or due to software clearing the IM field.
				• Software servicing all the pending interrupt status fields in the Fault Status register as follows.
				  • When primary fault logging is active, software clearing the Fault (F) field in all the Fault Recording registers with faults, causing the PPF field in the Fault Status register to be evaluated as Clear.
				  • Software clearing other status fields in the Fault Status register by writing back the value read from the respective fields.

Bits	Access	Default	Field	Description
29:0	RsvdP	Xh	R: Reserved	Reserved.

11.4.7.3 Fault Event Data Register

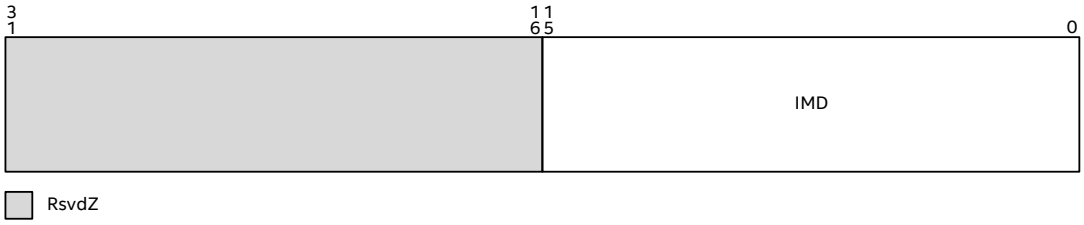


Figure 11-12. Fault Event Data Register

Abbreviation	FEDATA_REG
General Description	Register specifying the interrupt message data.
Register Offset	03Ch

Bits	Access	Default	Field	Description
31:16	RsvdZ	0h	R: Reserved	Reserved
15:0	RW	0h	IMD: Interrupt Message data	Data value in the interrupt request. Software requirements for programming this register are described in Section 5.1.6.

11.4.7.4 Fault Event Address Register

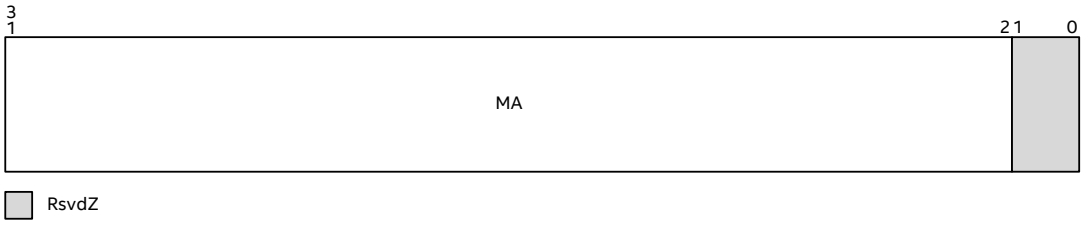


Figure 11-13. Fault Event Address Register

Abbreviation	FEADDR_REG
General Description	Register specifying the interrupt message address.
Register Offset	040h

Bits	Access	Default	Field	Description
31:2	RW	0h	MA: Message address	When fault events are enabled, the contents of this register specify the DWORD-aligned address (bits 31:2) for the interrupt request. Software requirements for programming this register are described in Section 5.1.6.
1:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.7.5 Fault Event Upper Address Register

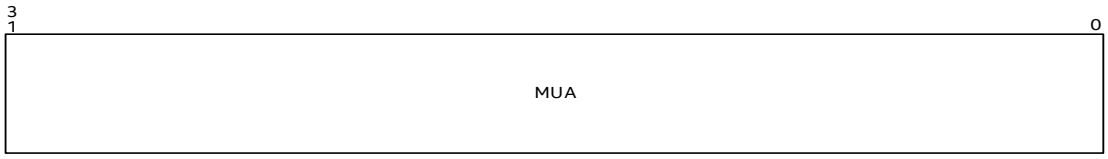


Figure 11-14. Fault Event Upper Address Register

Abbreviation	FEUADDR_REG
General Description	Register specifying the interrupt message upper address.
Register Offset	044h

Bits	Access	Default	Field	Description
31:0	RW	0h	MUA: Message upper address	Hardware implementations supporting Extended Interrupt Mode are required to implement this register. Software requirements for programming this register are described in Section 5.1.6. Hardware implementations not supporting Extended Interrupt Mode may treat this field as RsvdZ.

11.4.7.6 Fault Recording Registers [n]

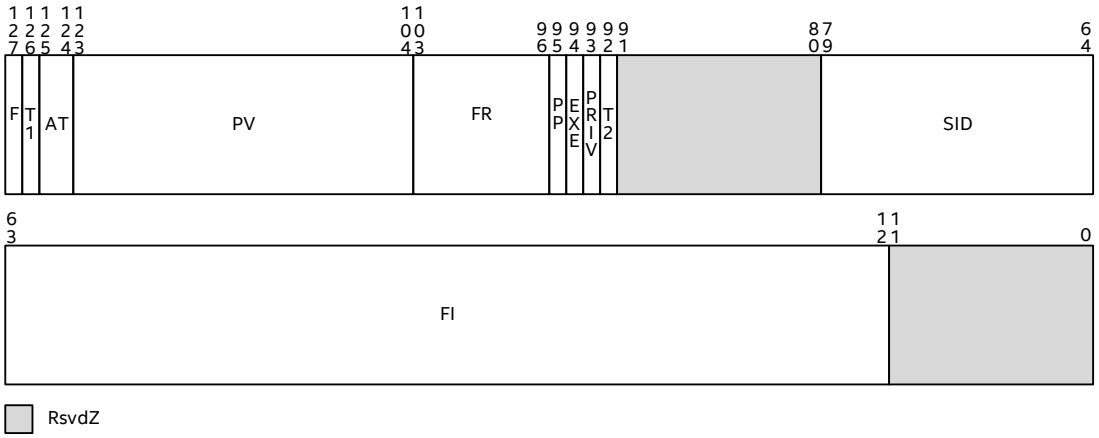


Figure 11-15. Fault Recording Register

Abbreviation	FRCD_REG [n]
General Description	Registers to record fault information when primary fault logging is active.Hardware reports the number and location of fault recording registers through the Capability register. This register is relevant only for primary fault logging.
Register Offset	YYYh (YYYh must be 128-bit aligned)

Bits	Access	Default	Field	Description
127	RW1CS	0	F: Fault[1]	Hardware sets this field to indicate a fault is logged in this Fault Recording register. The F field is Set by hardware after the details of the fault is recorded in other fields. When this field is Set, hardware may collapse additional faults from the same source-id (SID). Refer to Section 7.2.1 for hardware details of primary fault logging.
126	ROS	X	T1: Type bit1	Type of the faulted request: • 0: Write request or Page Request • 1: Read request or AtomicOp request This field is relevant only when the F field is Set, and when the fault reason (FR) indicates one of the address translation fault conditions.

Bits	Access	Default	Field	Description
125:124	ROS	Xh	AT: Address Type	This field captures the AT field from the faulted request. • 00b: Untranslated Request • 01b: Translation Request • 10b: Translated Request • 11b: Reserved Hardware implementations not supporting Device-TLBs (DT field Clear in Extended Capability register) treat this field as RsvdZ. When supported, this field is valid only for Read/Write/AtomicOp Requests, when the F field is Set, and when the fault reason (FR) indicates one of the non-recoverable address translation fault conditions.
123:104	ROS	Xh	PV: PASID Value	PASID value used by the faulted request. This field is relevant only when the PP field is Set. Hardware implementations not supporting PASID (PASID field Clear in Extended Capability register) implement this field as RsvdZ.
103:96	ROS	Xh	FR: Fault Reason	Reason for the fault. The enumerations for the various fault reason encodings are presented in Table 30 in Section 7.1.3 for DMA remapping faults, and Table 15 in Section 5.1.4.1 for interrupt remapping faults. This field is relevant only when the F field is Set.
95	ROS	X	PP: PASID Present	When Set, indicates the faulted request has a PASID TLP Prefix. The value of the PASID field is reported in the PASID Value (PV) field. This field is relevant only when the F field is Set, and when the fault reason (FR) indicates one of the non-recoverable address translation fault conditions. Hardware implementations not supporting PASID (PASID field Clear in Extended Capability register) implement this field as RsvdZ.
94	ROS	X	EXE: Execute Permission Requested	When Set, indicates Execute permission was requested by the faulted read request. This field is relevant only when the PP field is Set, T1=1, and T2=0 (Read Request). Hardware implementations not supporting PASID (PASID field Clear in Extended Capability register) implement this field as RsvdZ.
93	ROS	X	PRIV: Privilege Mode Requested	When Set, indicates Supervisor privilege was requested by the faulted request. This field is relevant only when the PP field is Set. Hardware implementations not supporting PASID (PASID field Clear in Extended Capability register) implement this field as RsvdZ.

<table>
<tr><th>Bits</th><th>Access</th><th>Default</th><th>Field</th><th>Description</th></tr>
<tr><td>92</td><td>ROS</td><td>X</td><td>T2: Type bit2</td><td>This field expands the Type field from 1-bit to 2-bit. It should be used in conjunction with field T1.

<table><tr><th>T1</th><th>T2</th><th>Description</th></tr><tr><td>0</td><td>0</td><td>Write Request</td></tr><tr><td>0</td><td>1</td><td>Page Request</td></tr><tr><td>1</td><td>0</td><td>Read Request</td></tr><tr><td>1</td><td>1</td><td>AtomicOp Request</td></tr></table>
This field is relevant only when the F field is Set, and when the fault reason (FR) indicates one of the address translation fault conditions.</td></tr>
<tr><td>91:80</td><td>RsvdZ</td><td>0h</td><td>R: Reserved</td><td>Reserved.</td></tr>
<tr><td>79:64</td><td>ROS</td><td>Xh</td><td>SID: Source Identifier</td><td>Requester-id associated with the fault condition.
This field is relevant only when the F field is Set.</td></tr>
<tr><td>63:12</td><td>ROS</td><td>Xh</td><td>FI: Fault Info</td><td>When the Fault Reason (FR) field indicates one of the address translation fault conditions, this field contains bits 63:12 of the page address in the faulted request. Hardware treats bits 63:N as reserved (0), where N corresponds to the largest AGAW value supported by hardware.
When the Fault Reason (FR) field indicates interrupt-remapping Fault Reason 20h, 25h, 29h, 2Ah, and 2Bh, contents of this field is undefined. When the Fault Reason (FR) field indicates interrupt-remapping fault conditions other than Fault Reason 20h, 25h, 29h, 2Ah, and 2Bh, bits 63:48 of this field indicate the interrupt_index computed for the faulted interrupt request, and bits 47:12 are cleared.

This field is relevant only when the F field is Set.</td></tr>
<tr><td>11:0</td><td>RsvdZ</td><td>0h</td><td>R: Reserved</td><td>Reserved.</td></tr>
</table>

1. Hardware updates to this register may be disassembled as multiple doubleword writes. To ensure consistent data is read from this register, software must first check the Primary Pending Fault (PPF) field in the FSTS_REG is Set before reading the fault reporting register at offset as indicated in the FRI field of FSTS_REG. Alternatively, software may read the highest doubleword in a fault recording register and check if the Fault (F) field is Set before reading the rest of the data fields in that register.

11.4.8 Protected Memory Range Registers

11.4.8.1 Protected Memory Enable Register

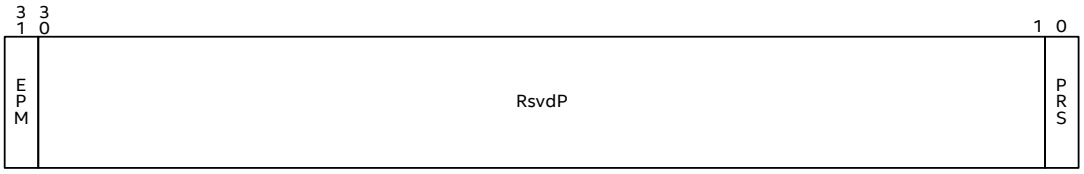


Figure 11-16. Protected Memory Enable Register

Abbreviation	PMEN_REG
General Description	Register to enable the DMA-protected memory regions set up through the PLMBASE, PLMLIMT, PHMBASE, PHMLIMIT registers. This register is always treated as RO for implementations not supporting protected memory regions (PLMR and PHMR fields reported as Clear in the Capability register). Protected memory regions may be used by software to securely initialize remapping structures in memory. Software must not program protected memory regions to overlap with a region designated as MMIO. Software must set up DMA-protected memory regions consistently across all the remapping hardware units supported by host platform for the protection to be effective. To avoid impact to legacy BIOS usage of memory, software is recommended to not overlap protected memory regions with any reserved memory regions of the platform reported through the Reserved Memory Region Reporting (RMRR) structures described in Chapter 8. New software development should not use Protected Memory Registers as they are planned for deprecation in a future generation. Abort-dma mode can be used to protect memory during DMA Remapping initialization. See Section 3.4.4 for more details.
Register Offset	064h

Bits	Access	Default	Field	Description

31	RW	0h	EPM: Enable Protected Memory	This field controls DMA accesses to the protected low-memory and protected high-memory regions.
				• 0: Protected memory regions are disabled.
				• 1: Protected memory regions are enabled. DMA requests accessing protected memory regions are handled as follows:
				  • PCI and CXL.io links
				  • Hardware implementations with Major Version 2 or higher (VER_REG) block all DMA requests accessing protected memory regions whether or not DMA remapping is enabled.
				  • Hardware implementations starting with 6th Generation Intel® Core™ (codename: Skylake) and Intel® Xeon® Scalable Processors (codename: Skylake) block all DMA requests accessing protected memory regions whether or not DMA remapping is enabled.
				  • Some earlier hardware implementations (earlier than those referred above) do not block DMA requests that are subject to address remapping (i.e. requests other than pass-through and translated) from accessing the protected memory when DMA remapping is enabled. On such old hardware, software must ensure that there are no mappings programmed in the remapping structures that map to the protected memory region.
				• CXL.cache links
				  • PMR ranges do not protect accesses through CXL.cache. Software enabling CXL.cache must protect desired memory region via DMA remapping page-tables and not depend on PMR
				
				Remapping hardware access to the remapping structures are not subject to protected memory region checks.DMA requests blocked due to protected memory region violation are not recorded or reported as remapping faults.
				Hardware reports the status of the protected memory enable/disable operation through the PRS field in this register. Hardware implementations supporting DMA draining must drain any in-flight translated DMA requests queued within the Root-Complex before indicating the protected memory region as enabled through the PRS field.
				
				After writing to this field software must wait for the operation to be completed and reflected in the PRS status field (bit 0) before changing the value of this field again.

30:1	RsvdP	Xh	R: Reserved	Reserved.
0	RO	0h	PRS: Protected Region Status	This field indicates the status of protected memory region(s): • 0: Protected memory region(s) disabled. • 1: Protected memory region(s) enabled.

11.4.8.2 Protected Low-Memory Base Register

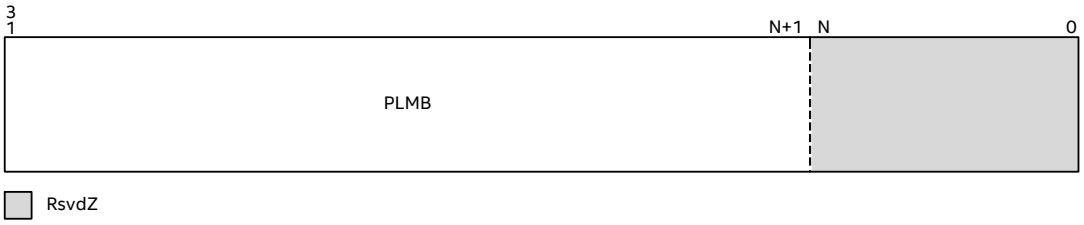


Figure 11-17. Protected Low-Memory Base Register

Abbreviation	PLMBASE_REG
General Description	Register to set up the base address of DMA-protected low-memory region below 4GB. This register must be set up before enabling protected memory through PMEN_REG, and must not be updated when protected memory regions are enabled. This register is always treated as RO for implementations not supporting protected low memory region (PLMR field reported as Clear in the Capability register). The alignment of the protected low memory region base depends on the number of reserved bits (N:0) of this register. Software may determine N by writing all 1s to this register, and finding the most significant bit position with 0 in the value read back from the register. Bits N:0 of this register are decoded by hardware as all 0s. Software must setup the protected low memory region below 4GB. Section 11.4.8.3 describes the Protected Low-Memory Limit register and hardware decoding of these registers. Software must not modify this register when protected memory regions are enabled (PRS field Set in PMEN_REG)
Register Offset	068h

Bits	Access	Default	Field	Description
31:(N+1)	RW	0h	PLMB: Protected Low-Memory Base	This register specifies the base of protected low-memory region in system memory.
N:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.8.3 Protected Low-Memory Limit Register

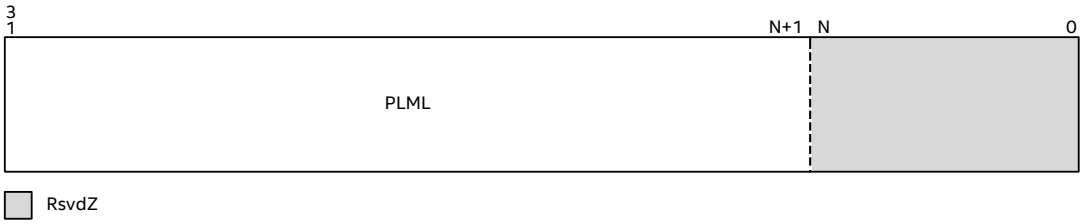


Figure 11-18. Protected Low-Memory Limit Register

Abbreviation	PLMLIMIT_REG
General Description	Register to set up the limit address of DMA-protected low-memory region below 4GB. This register must be set up before enabling protected memory through PMEN_REG, and must not be updated when protected memory regions are enabled. This register is always treated as RO for implementations not supporting protected low memory region (PLMR field reported as Clear in the Capability register). The alignment of the protected low memory region limit depends on the number of reserved bits (N:0) of this register. Software may determine N by writing all 1's to this register, and finding most significant zero bit position with 0 in the value read back from the register. Bits N:0 of the limit register are decoded by hardware as all 1s. The Protected low-memory base and limit registers function as follows: • Programming the protected low-memory base and limit registers the same value in bits 31:(N+1) specifies a protected low-memory region of size $2^{(N+1)}$ bytes. • Programming the protected low-memory limit register with a value less than the protected low-memory base register disables the protected low-memory region. Software must not modify this register when protected memory regions are enabled (PRS field Set in PMEN_REG)
Register Offset	06Ch

Bits	Access	Default	Field	Description
31:(N+1)	RW	0h	PLML: Protected Low-Memory Limit	This register specifies the last host physical address of the DMA-protected low-memory region in system memory.
N:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.8.4 Protected High-Memory Base Register

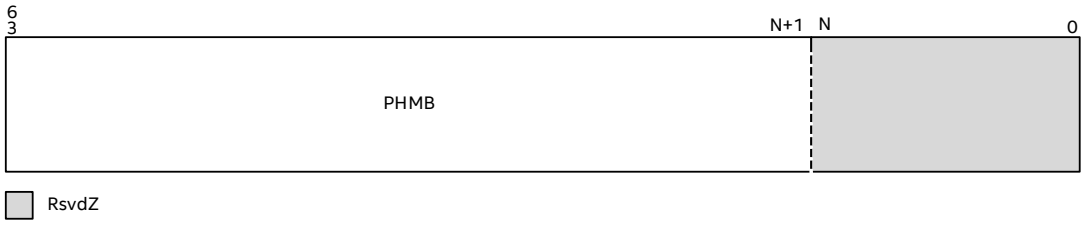


Figure 11-19. Protected High-Memory Base Register

Abbreviation	PHMBASE_REG
General Description	Register to set up the base address of DMA-protected high-memory region. This register must be set up before enabling protected memory through PMEN_REG, and must not be updated when protected memory regions are enabled. This register is always treated as RO for implementations not supporting protected high memory region (PHMR field reported as Clear in the Capability register). The alignment of the protected high memory region base depends on the number of reserved bits (N:0) of this register. Software may determine N by writing all 1's to this register, and finding most significant zero bit position below host address width (HAW) in the value read back from the register. Bits N:0 of this register are decoded by hardware as all 0s. Software may setup the protected high memory region either above or below 4GB. Section 11.4.8.5 describes the Protected High-Memory Limit register and hardware decoding of these registers. Software must not modify this register when protected memory regions are enabled (PRS field Set in PMEN_REG)
Register Offset	070h

Bits	Access	Default	Field	Description
63:(N+1)	RW	0h	PHMB: Protected High-Memory Base	This register specifies the base of protected (high) memory region in system memory. Hardware may ignore and not implement bits 63:HAW, where HAW is the host address width.
N:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.8.5 Protected High-Memory Limit Register

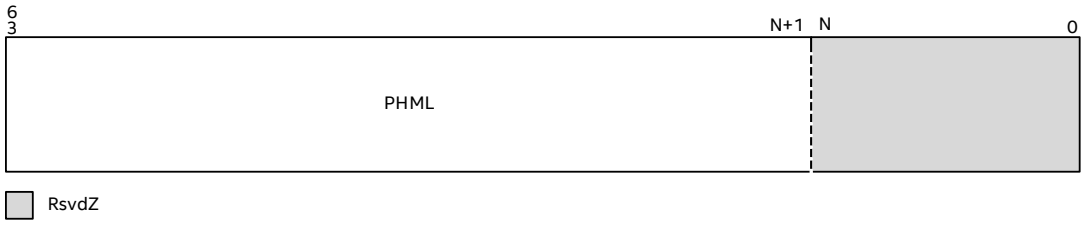


Figure 11-20. Protected High-Memory Limit Register

Abbreviation	PHMLIMIT_REG
General Description	Register to set up the limit address of DMA-protected high-memory region. This register must be set up before enabling protected memory through PMEN_REG, and must not be updated when protected memory regions are enabled. This register is always treated as RO for implementations not supporting protected high memory region (PHMR field reported as Clear in the Capability register). The alignment of the protected high memory region limit depends on the number of reserved bits (N:0) of this register. Software may determine N by writing all 1's to this register, and finding most significant zero bit position below host address width (HAW) in the value read back from the register. Bits N:0 of the limit register are decoded by hardware as all 1s. The protected high-memory base & limit registers function as follows. • Programming the protected low-memory base and limit registers with the same value in bits HAW:(N+1) specifies a protected low-memory region of size $2^{(N+1)}$ bytes. • Programming the protected high-memory limit register with a value less than the protected high-memory base register disables the protected high-memory region. Software must not modify this register when protected memory regions are enabled (PRS field Set in PMEN_REG)
Register Offset	078h

Bits	Access	Default	Field	Description
63:(N+1)	RW	0h	PHML: Protected High-Memory Limit	This register specifies the last host physical address of the DMA-protected high-memory region in system memory. Hardware may ignore and not implement bits 63:HAW, where HAW is the host address width.
N:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.9 Invalidation Queue Interface

11.4.9.1 Invalidation Queue Head Register

Figure 11-21. Invalidation Queue Head Register

Abbreviation	IQH_REG
General Description	Register indicating the invalidation queue head. This register is treated as RsvdZ by implementations reporting Queued Invalidation (QI) as not supported in the Extended Capability register.
Register Offset	080h

Bits	Access	Default	Field	Description
63:19	RsvdZ	0h	R: Reserved	Reserved.
18:4	RO	0h	QH: Queue Head	Specifies the offset (128-bit or 256-bit aligned) to the invalidation queue for the command that will be processed next by hardware. When Descriptor Width (DW) field in Invalidation Queue Address Register (IQA_REG) is Set (256-bit descriptors), hardware treats bit-4 as reserved and will always write a value of 0 in the bit. Hardware resets this field to 0 whenever the queued invalidation is disabled (QIES field Clear in the Global Status register).
3:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.9.2 Invalidation Queue Tail Register

Figure 11-22. Invalidation Queue Tail Register

Abbreviation	IQT_REG
General Description	Register indicating the invalidation tail. This register is treated as RsvdZ by implementations reporting Queued Invalidation (QI) as not supported in the Extended Capability register.
Register Offset	088h

Bits	Access	Default	Field	Description
63:19	RsvdZ	0h	R: Reserved	Reserved.
18:4	RW	0h	QT: Queue Tail	Specifies the offset (128-bit or 256-bit aligned) to the invalidation queue for the command that will be written next by software. When Descriptor Width (DW) field in Invalidation Queue Address Register (IQA_REG) is Set (256-bit descriptors), hardware treats bit-4 as reserved and a value of 1 in the bit will result in invalidation queue error.
3:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.9.3 Invalidation Queue Address Register

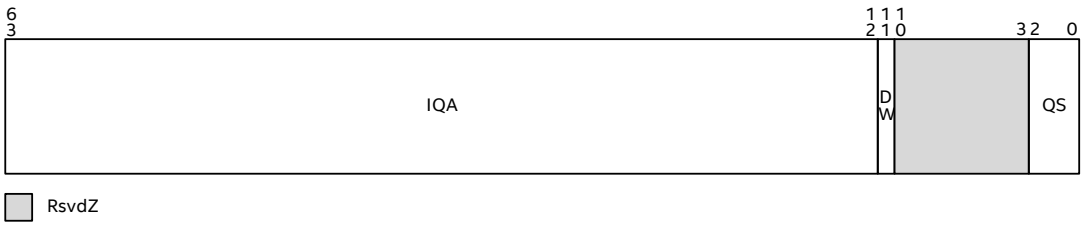


Figure 11-23. Invalidation Queue Address Register

Abbreviation	IQA_REG
General Description	Register to configure the base address and size of the invalidation queue. This register is treated as RsvdZ by implementations reporting Queued Invalidation (QI) as not supported in the Extended Capability register. Software should not modify this register while the invalidation queue is not empty. (Refer to Section 6.5.2 for more details.)
Register Offset	090h

Bits	Access	Default	Field	Description
63:12	RW	0h	IQA: Invalidation Queue Base Address	This field points to the base of 4KB aligned invalidation request queue. Hardware may ignore and not implement bits 63:HAW, where HAW is the host address width. Reads of this field return the value that was last programmed to it.
11	RW	0h	DW: Descriptor Width	This field specifies the size of the descriptors submitted into invalidation request queue. • 0: 128-bit descriptors • 1: 256-bit descriptors When both scalable mode translation and abort-dma mode are not supported (ECAP_REG.SMTS=0 and ECAP_REG.ADMS=0), hardware treats a value of 1 in this field as an Invalidation Queue Error (see Invalidation Queue Error Info field in Section 11.4.9.9 for details).
10:3	RsvdZ	0h	R: Reserved	Reserved.
2:0	RW	0h	QS: Queue Size	This field specifies the size of the invalidation request queue. A value of X in this field indicates an invalidation request queue of (2^X) 4KB pages. The number of entries in the invalidation queue is $2^{(X + 8)}$ if Descriptor Width is 0 and $2^{(X + 7)}$ if Descriptor Width is 1.

11.4.9.4 Invalidation Completion Status Register

Figure 11-24. Invalidation Completion Status Register

Abbreviation	ICS_REG
General Description	Register to report completion status of invalidation wait descriptor with Interrupt Flag (IF) Set. This register is treated as RsvdZ by implementations reporting Queued Invalidation (QI) as not supported in the Extended Capability register.
Register Offset	09Ch

Bits	Access	Default	Field	Description
31:1	RsvdZ	0h	R: Reserved	Reserved.
0	RW1CS	0	IWC: Invalidation Wait Descriptor Complete	Indicates completion of Invalidation Wait Descriptor with Interrupt Flag (IF) field Set. Hardware implementations not supporting queued invalidations implement this field as RsvdZ.

11.4.9.5 Invalidation Event Control Register

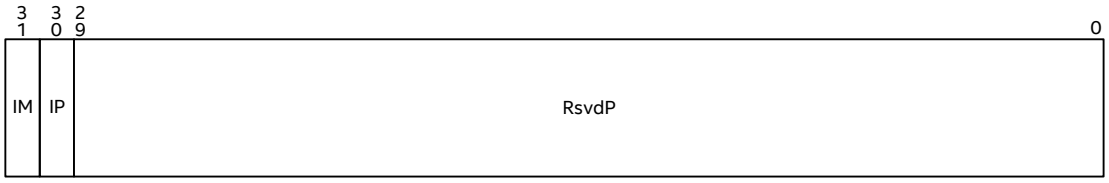


Figure 11-25. Invalidation Event Control Register

Abbreviation	IECTL_REG
General Description	Register specifying the invalidation event interrupt control bits. This register is treated as RsvdZ by implementations reporting Queued Invalidation (QI) as not supported in the Extended Capability register.
Register Offset	0A0h

Bits	Access	Default	Field	Description
31	RW	1	IM: Interrupt Mask	• 0: No masking of interrupt. When a invalidation event condition is detected, hardware issues an interrupt message (using the Invalidation Event Data & Invalidation Event Address register values). • 1: This is the value on reset. Software may mask interrupt message generation by setting this field. Hardware is prohibited from sending the interrupt message when this field is Set.
30	RO	0	IP: Interrupt Pending	Hardware sets the IP field whenever it detects an interrupt condition. Interrupt condition is defined as: • An Invalidation Wait Descriptor with Interrupt Flag (IF) field Set completed, setting the IWC field in the Invalidation Completion Status register. • If the IWC field in the Invalidation Completion Status register was already Set at the time of setting this field, it is not treated as a new interrupt condition. The IP field is kept Set by hardware while the interrupt message is held pending. The interrupt message could be held pending due to interrupt mask (IM field) being Set, or due to other transient hardware conditions. The IP field is cleared by hardware as soon as the interrupt message pending condition is serviced. This could be due to either: • Hardware issuing the interrupt message due to either change in the transient hardware condition that caused interrupt message to be held pending or due to software clearing the IM field. • Software servicing the IWC field in the Invalidation Completion Status register.
29:0	RsvdP	Xh	R: Reserved	Reserved.

11.4.9.6 Invalidation Event Data Register

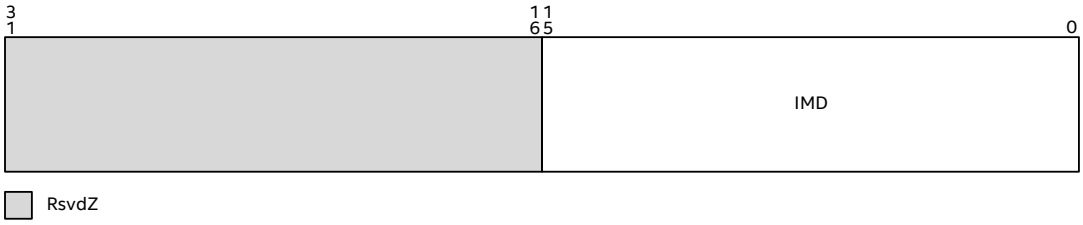


Figure 11-26. Invalidation Event Data Register

Abbreviation	IEDATA_REG
General Description	Register specifying the Invalidation Event interrupt message data. This register is treated as RsvdZ by implementations reporting Queued Invalidation (QI) as not supported in the Extended Capability register.
Register Offset	0A4h

Bits	Access	Default	Field	Description
31:16	RsvdZ	0h	R: Reserved	Reserved
15:0	RW	0h	IMD: Interrupt Message data	Data value in the interrupt request. Software requirements for programming this register are described in Section 5.1.6.

11.4.9.7 Invalidation Event Address Register

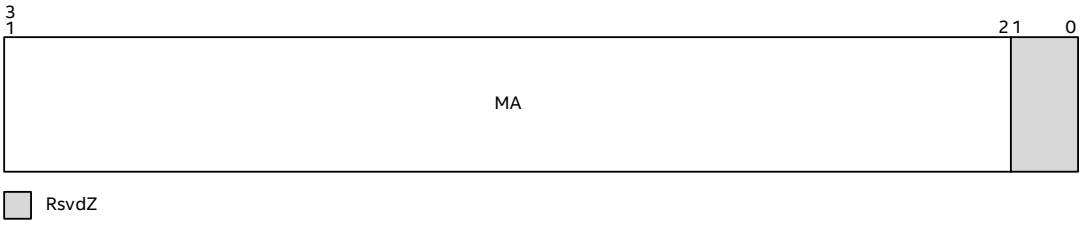


Figure 11-27. Invalidation Event Address Register

Abbreviation	IEADDR_REG
General Description	Register specifying the Invalidation Event Interrupt message address. This register is treated as RsvdZ by implementations reporting Queued Invalidation (QI) as not supported in the Extended Capability register.
Register Offset	0A8h

Bits	Access	Default	Field	Description
31:2	RW	0h	MA: Message address	When fault events are enabled, the contents of this register specify the DWORD-aligned address (bits 31:2) for the interrupt request. Software requirements for programming this register are described in Section 5.1.6.
1:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.9.8 Invalidation Event Upper Address Register

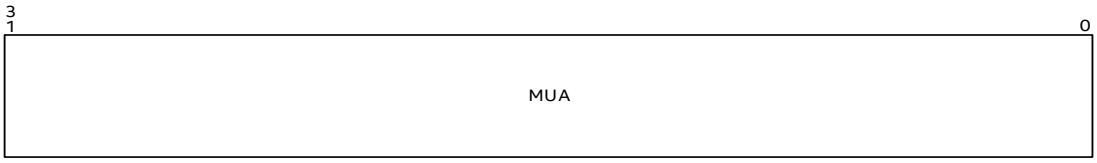


Figure 11-28. Invalidation Event Upper Address Register

Abbreviation	IEUADDR_REG
General Description	Register specifying the Invalidation Event interrupt message upper address. This register is treated as RsvdZ by implementations reporting Queued Invalidation (QI) as not supported in the Extended Capability register.
Register Offset	0ACh

Bits	Access	Default	Field	Description
31:0	RW	0h	MUA: Message upper address	Hardware implementations supporting Queued Invalidations and Extended Interrupt Mode are required to implement this register. Software requirements for programming this register are described in Section 5.1.6. Hardware implementations not supporting Queued Invalidations or Extended Interrupt Mode may treat this field as RsvdZ.

11.4.9.9 Invalidation Queue Error Record Register

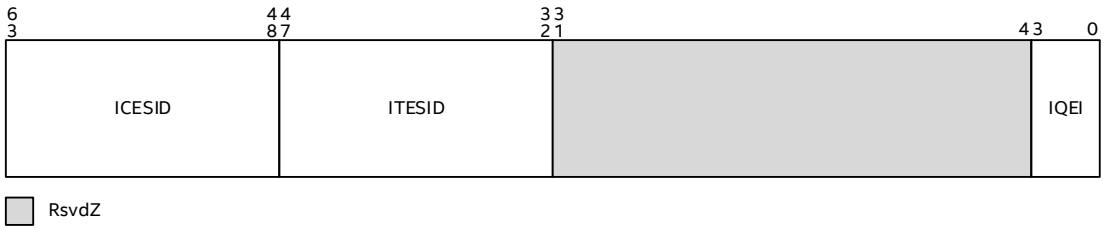


Figure 11-29. Invalidation Queue Error Record Register

Abbreviation	IQERCD_REG
General Description	Register providing information about Invalidation Queue Error. This register is treated as RsvdZ by implementations reporting Queued Invalidation (QI) as not supported in the Extended Capability register.
Register Offset	0B0h

Bits	Access	Default	Field	Description
63:48	ROS	0h	ICESID: Invalidation Completion Error Source Identifier	Requester-id associated with Device-TLB invalidation completion that causes an Invalidation Completion Error. A value of 0 in this field indicates that this is an older version of DMA remapping hardware which does not provide additional details about the Invalidation Completion Error. If the ICE field is Clear at the time of the Invalidation Completion Error detection, hardware copies the Requester-id in the Invalidation Completion Message that resulted in the error into this field. This field is valid only when the ICE field is Set in FSTS_REG. The value read from this field is undefined when the ICE field is Clear.
47:32	ROS	0h	ITESID: Invalidation Time-out Error Source Identifier	Requester-id associated with Invalidation Time-out Error. A value of 0 in this field indicates that this is an older version of DMA remapping hardware which does not provide additional details about the Invalidation Time-out Error. If the ITE field is Clear at the time of the Invalidation Time-out error detection, hardware copies the Requester-id associated with error into this field. If multiple Invalidation Time-out errors are detected at the same time, hardware chooses one of them to be reported. This field is valid only when the ITE field is Set in FSTS_REG. The value read from this field is undefined when the ITE field is Clear.
31:4	RsvdZ	0h	R: Reserved	Reserved

Bits	Access	Default	Field	Description

3:0	ROS	0h	IQEI: Invalidation Queue Error Info	This field is valid only when the IQE field is Set in FSTS_REG. The value read from this field is undefined when the IQE field is Clear.
				
				This field provides additional details about the what caused IQE field to be Set.
				• 0: info not available. (This is an older version of DMA Remapping hardware which does not provide additional details about IQ errors.)
				• 1: hardware detected an invalid Tail Pointer.
				• 2: hardware attempt to fetch descriptor resulted in error.
				• 3: hardware detected an invalid descriptor type. (See Table 26 for details.)
				• 4: hardware detected a reserved field violation for a valid descriptor type
				• 5: hardware detected an invalid descriptor width programmed in the Invalidation Queue Address Register (IQA_REG)
				  • Descriptor width of 128-bit (IQA_REG.DW=0) when operating in scalable mode (RTADDR_REG.TTM=01b) or abort-dma mode (RTADDR_REG.TTM=11b).
				  • Descriptor width of 256-bit (IQA_REG.DW=1) when both Scalable Mode Translation Support and Abort-DMA Mode Support are reported as Clear (ECAP_REG.SMTS=0 and ECAP_REG.ADMS=0)
				• 6: hardware detected that Queue Tail is not aligned to the descriptor width (i.e. IQA_REG.DW=1 and IQT.b[4]≠0).
				• 7: hardware detected an invalid value in the TTM field of the Root Table Address (RTADDR_REG) register.
				• 8-15: undefined.

11.4.10 Interrupt Remapping Table Address Register

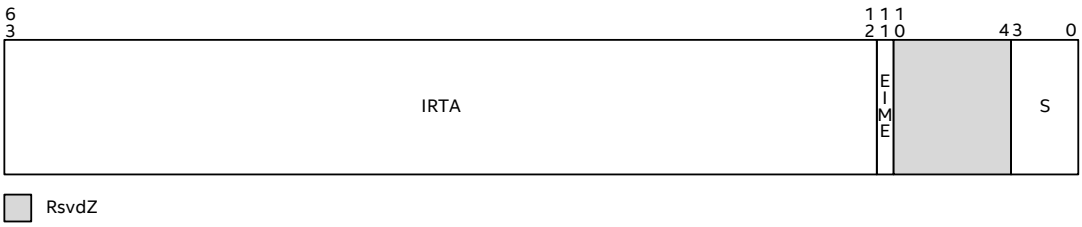


Figure 11-30. Interrupt Remapping Table Address Register

Abbreviation	IRTA_REG
General Description	Register providing the base address of Interrupt remapping table. This register is treated as RsvdZ by implementations reporting Interrupt Remapping (IR) as not supported in the Extended Capability register.
Register Offset	0B8h

Bits	Access	Default	Field	Description
63:12	RW	0h	IRTA: Interrupt Remapping Table Address	This field points to the base of 4KB aligned interrupt remapping table. Hardware may ignore and not implement bits 63:HAW, where HAW is the host address width. The value of this field takes effect only after software executes Set Interrupt Remap Table Pointer command.
11	RW	0	EIME: Extended Interrupt Mode Enable	This field is used by hardware on Intel® 64 platforms as follows: • 0: xAPIC mode is active. Hardware interprets only 8-bits ([15:8]) of Destination-ID field in the IRTEs. The high 16-bits and low 8-bits of the Destination-ID field are treated as reserved. • 1: x2APIC mode is active. Hardware interprets all 32-bits of Destination-ID field in the IRTEs. This field is implemented as RsvdZ on implementations reporting Extended Interrupt Mode (EIM) field as Clear in Extended Capability register. Software must program this field as Set for implementations reporting Extended Interrupt Mode Enable Required (EIMER) as Set in the Extended Capability register. Software must not modify this field while Interrupt remapping is active (IRES=1 in Global Status register). The value of this field takes effect only after software executes Set Interrupt Remap Table Pointer command.
10:4	RsvdZ	0h	R: Reserved	Reserved.
3:0	RW	0h	S: Size	This field specifies the size of the interrupt remapping table. The number of entries in the interrupt remapping table is 2^{X+1}, where X is the value programmed in this field. The value of this field takes effect only after software executes Set Interrupt Remap Table Pointer command.

11.4.11 Page Request Queue Interface

11.4.11.1 Page Request Queue Head Register

Figure 11-31. Page Request Queue Head Register

Abbreviation	PQH_REG
General Description	Register indicating the page request queue head. This register is treated as RsvdZ by implementations reporting Page Request Support (PRS) as not supported in the Extended Capability register.
Register Offset	0C0h

Bits	Access	Default	Field	Description
63:19	RsvdZ	0h	R: Reserved	Reserved.
18:5	RW	0h	PQH: Page Queue Head	Specifies the offset (32-bytes aligned) to the page request queue for the request that will be processed next by software.
4:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.11.2 Page Request Queue Tail Register

Figure 11-32. Page Request Queue Tail Register

Abbreviation	PQT_REG
General Description	Register indicating the page request queue tail. This register is treated as RsvdZ by implementations reporting Page Request Support (PRS) as not supported in the Extended Capability register.
Register Offset	0C8h

Bits	Access	Default	Field	Description
63:19	RsvdZ	0h	R: Reserved	Reserved.
18:5	RW	0h	PQT: Page Queue Tail	Specifies the offset (32-bytes aligned) to the page request queue for the request that will be written next by hardware.
4:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.11.3 Page Request Queue Address Register

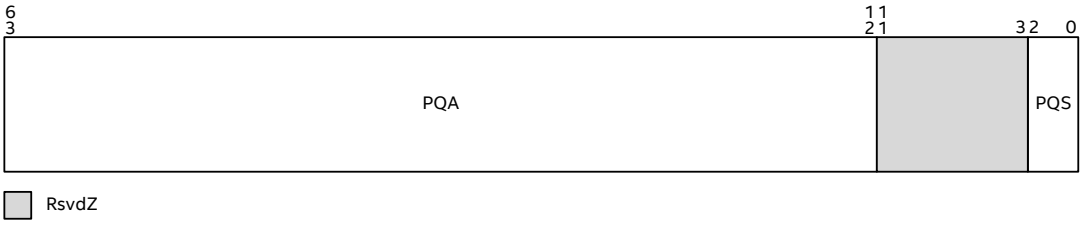


Figure 11-33. Page Request Queue Address Register

Abbreviation	PQA_REG
General Description	Register to configure the base address and size of the page request queue. This register is treated as RsvdZ by implementations reporting Page Request Support (PRS) as not supported in the Extended Capability register.
Register Offset	0D0h

Bits	Access	Default	Field	Description
63:12	RW	0h	PQA: Page Request Queue Base Address	This field points to the base of 4KB aligned page request queue. Hardware may ignore and not implement bits 63:HAW, where HAW is the host address width. Software must configure this register to point to host memory before enabling page requests in any scalable-mode context-entries.
11:3	RsvdZ	0h	R: Reserved	Reserved.
2:0	RW	0h	PQS: Page Queue Size	This field specifies the size of the page request queue. A value of X in this field indicates a page request queue of (2^X) 4KB pages. The number of entries in the page request queue is $2^{(X + 7)}$.

11.4.11.4 Page Request Status Register

Figure 11-34. Page Request Status Register

Abbreviation	PRS_REG
General Description	Register to report pending page request in page request queue. This register is treated as RsvdZ by implementations reporting Page Request Support (PRS) as not supported in the Extended Capability register.
Register Offset	0DCh

Bits	Access	Default	Field	Description
31:2	RsvdZ	0h	R: Reserved	Reserved.
1	RW1CS	0	PRO: Page Request Overflow	Hardware sets this field to indicate Page Request Queue overflow condition. Page Request event is generated based on programming of the Page Request Event Control register. Page request message that led to setting this bit and subsequent messages received while this field is already Set are discarded or responded by hardware as described in Section 7.4.1. Software writing a 1 to this field clears it.
0	RW1CS	0	PPR: Pending Page Request	Indicates pending page requests to be serviced by software in the page request queue. This field is Set by hardware when a page request entry (*page_req_dsc*) is added to the page request queue. Software writing a 1 to this field clears it.

11.4.11.5 Page Request Event Control Register

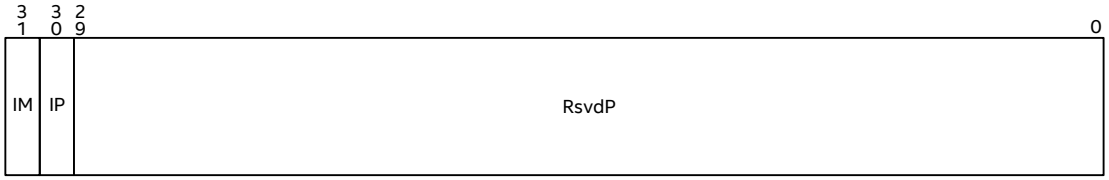


Figure 11-35. Page Request Event Control Register

Abbreviation	PECTL_REG
General Description	Register specifying the page request event interrupt control bits. This register is treated as RsvdZ by implementations reporting Page Request Support (PRS) as not supported in the Extended Capability register.
Register Offset	0E0h

Bits	Access	Default	Field	Description
31	RW	1	IM: Interrupt Mask	• 0: No masking of interrupt. When a page request event condition is detected, hardware issues an interrupt message (using the Page Request Event Data & Page Request Event Address register values). • 1: This is the value on reset. Software may mask interrupt message generation by setting this field. Hardware is prohibited from sending the interrupt message when this field is Set.
30	RO	0	IP: Interrupt Pending	Hardware sets the IP field whenever it detects an interrupt condition. An Interrupt condition is defined as: • A page request entry (page_req_dsc) was added to page request queue, resulting in hardware setting the Pending Page Request (PPR) field in Page Request Status register. • Hardware detected Page Request Queue overflow condition, setting the PRO field in the Page Request Status register. If any of the status fields in the Page Request Event Status register was already Set at the time of setting any of these fields, it is not treated as a new interrupt condition. The IP field is kept Set by hardware while the interrupt message is held pending. The interrupt message could be held pending due to interrupt mask (IM field) being Set, or due to other transient hardware conditions. The IP field is cleared by hardware as soon as the interrupt message pending condition is serviced. This could be due to either: • Hardware issuing the interrupt message due to either change in the transient hardware condition that caused interrupt message to be held pending or due to software clearing the IM field. • Software servicing the PPR field in the Page Request Event Status register.
29:0	RsvdP	Xh	R: Reserved	Reserved.

11.4.11.6 Page Request Event Data Register

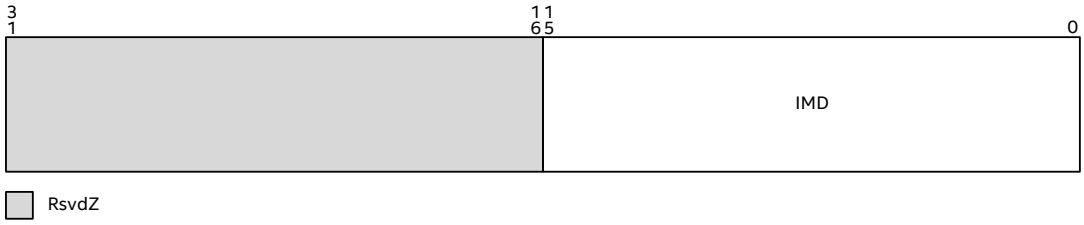


Figure 11-36. Page Request Event Data Register

Abbreviation	PEDATA_REG
General Description	Register specifying the Page Request Event interrupt message data. This register is treated as RsvdZ by implementations reporting Page Request Support (PRS) as not supported in the Extended Capability register.
Register Offset	0E4h

Bits	Access	Default	Field	Description
31:16	RsvdZ	0h	R: Reserved	Reserved
15:0	RW	0h	IMD: Interrupt Message data	Data value in the interrupt request. Software requirements for programming this register are described in Section 5.1.6.

11.4.11.7 Page Request Event Address Register

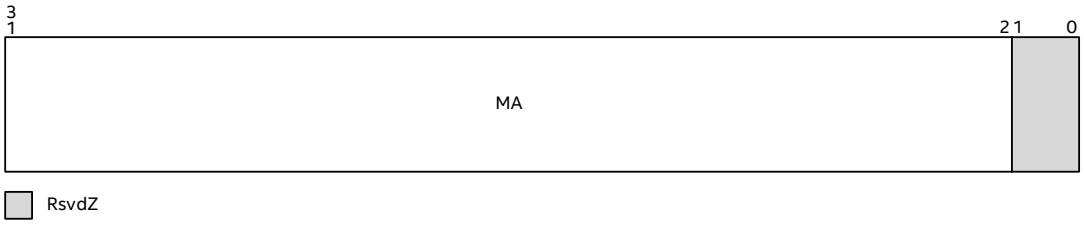


Figure 11-37. Page Request Event Address Register

Abbreviation	PEADDR_REG
General Description	Register specifying the Page Request Event Interrupt message address.This register is treated as RsvdZ by implementations reporting Page Request Support (PRS) as not supported in the Extended Capability register.
Register Offset	0E8h

Bits	Access	Default	Field	Description
31:2	RW	0h	MA: Message address	When fault events are enabled, the contents of this register specify the DWORD-aligned address (bits 31:2) for the interrupt request. Software requirements for programming this register are described in Section 5.1.6.
1:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.11.8 Page Request Event Upper Address Register

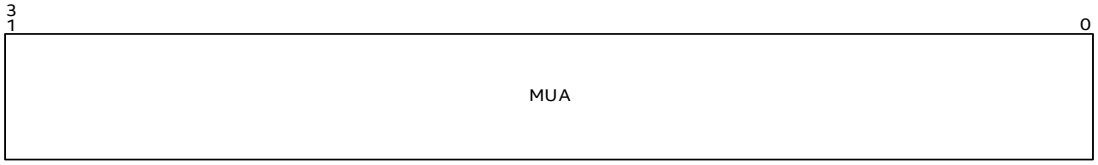


Figure 11-38. Page Request Event Upper Address Register

Abbreviation	PEUADDR_REG
General Description	Register specifying the Page Request Event interrupt message upper address.
Register Offset	0ECh

Bits	Access	Default	Field	Description
31:0	RW	0h	MUA: Message upper address	This field specifies the upper address (bits 63:32) for the page request event interrupt. Software requirements for programming this register are described in Section 5.1.6. Hardware implementations not supporting Extended Interrupt Mode may treat this field as RsvdZ.

11.4.12 Memory Type Range Registers

11.4.12.1 MTRR Capability Register

Figure 11-39. MTRR Capability Register

Abbreviation	MTRRCAP_REG
General Description	Register reporting the Memory Type Range Register Capability. This register is treated as RsvdZ by implementations reporting Memory Type Support (MTS) as not supported in the Extended Capability register. When implemented, value reported in this register must match IA32_MTRRCAP Model Specific Register (MSR) value reported by the host IA-32 processor(s).
Register Offset	100h

Bits	Access	Default	Field	Description
63:11	RsvdZ	0h	R: Reserved	Reserved
10	RO	X	WC: Write Combining	0: Write-combining (WC) memory type is not supported 1: Write-combining (WC) memory type is supported
9	RsvdZ	X	R: Reserved	Reserved
8	RO	X	FIX: Fixed Range MTRRs Supported	• 0: No fixed range MTRRs are supported • 1: Fixed range MTRRs (MTRR_FIX64K_00000 through MTRR_FIX4K_0F8000) are supported
7:0	RO	X	VCNT: Variable MTRR Count	Indicates number of variable range MTRRs supported

11.4.12.2 MTRR Default Type Register

Figure 11-40. MTRR Default Type Register

Abbreviation	MTRRDEF_REG
General Description	Register for enabling/configuring Memory Type Range Registers. This register is treated as RsvdZ by implementations reporting Memory Type Support (MTS) as not supported in the Extended Capability register.
Register Offset	108h

Bits	Access	Default	Field	Description
63:12	RsvdZ	0h	R: Reserved	Reserved.
11	RW	0	E: MTRR Enable	• 0: Disable MTRRs; UC memory type is applied. FE field has no effect. • 1: Enable MTRRs. FE field can disable the fixed-range MTRRs. Type specified in the default memory type field is used for areas of memory not already mapped by either fixed or variable MTRR.
10	RW	0	FE: Fixed Range MTRR Enable	• 0: Disable fixed range MTRRs. • 1: Enable fixed range MTRRs. When fixed range MTRRs are enabled, they take priority over the variable range MTRRs when overlaps in ranges occur. If the fixed-range MTRRs are disabled, the variable range MTRRs can still be used and can map the range ordinarily covered by the fixed range MTRRs.
9:8	RsvdZ	0h	R: Reserved	Reserved.
7:0	RW	0h	TYPE: Default Memory Type	Indicates default memory type used for physical memory address ranges that do not have a memory type specified for them by an MTRR. Legal values for this field are 0,1,4, 5 and 6.

11.4.12.3 Fixed-Range MTRRs

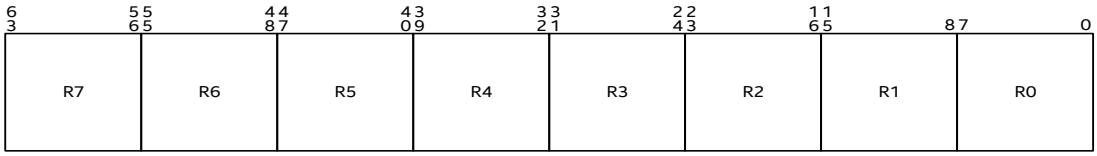


Figure 11-41. Fixed-Range MTRR Format

Abbreviation	MTRR_FIX64_00000_REG MTRR_FIX16K_80000_REG MTRR_FIX16K_A0000_REG MTRR_FIX4K_C0000_REG MTRR_FIX4K_C8000_REG MTRR_FIX4K_D0000_REG MTRR_FIX4K_D8000_REG MTRR_FIX_4K_E0000_REG MTRR_FIX_4K_E8000REG MTRR_FIX4K_F0000_REG MTRR_FIX_4K_F8000_REG
General Description	Fixed-range Memory Type Range Registers. These include 11 registers as illustrated in Table 50. These registers are treated as RsvdZ by implementations reporting Memory Type Support (MTS) as not supported in the Extended Capability register.
Register Offsets	120h, 128h, 130h, 138h, 140h, 148h, 150h, 158h, 160h, 168h, 170h

Bits	Access	Default	Field	Description
63:56	RW	0h	R7	Register Field 7
55:48	RW	0h	R6	Register Field 6
47:40	RW	0h	R5	Register Field 5
39:32	RW	0h	R4	Register Field 4
31:24	RW	0h	R3	Register Field 3
23:16	RW	0h	R2	Register Field 2
15:8	RW	0h	R1	Register Field 1
7:0	RW	0h	R0	Register Field 0

Table 50. Address Mapping for Fixed-Range MTRRs

Address Range (hexadecimal)								MTRR
63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0	
70000 - 7FFFF	60000 - 6FFFF	50000 - 5FFFF	40000 - 4FFFF	30000 - 3FFFF	20000 - 2FFFF	10000 - 1FFFF	00000 - 0FFFF	MTRR_FIX64K_ 00000_REG
9C000- 9FFFF	98000- 98FFF	94000- 97FFF	90000- 93FFF	8C000- 8FFFF	88000- 8BFFF	84000- 87FFF	80000- 83FFF	MTRR_FIX16K_ 80000_REG
BC000- BFFFF	B8000- B8FFF	B4000- B7FFF	B0000- B3FFF	AC000- AFFFF	A8000- ABFFF	A4000- A7FFF	A0000- A3FFF	MTRR_FIX16K_ A0000_REG
C7000- C7FFF	C6000- C6FFF	C5000- C5FFF	C4000- C4FFF	C3000- C3FFF	C2000- C2FFF	C1000- C1FFF	C0000- C0FFF	MTRR_FIX4K_ C0000_REG
CF000- CFFFF	CE000- CEFFF	CD000- CDFFF	CC000- CCFFF	CB000- CBFFF	CA000- CAFFF	C9000- C9FFF	C8000- C8FFF	MTRR_FIX4K_ C8000_REG
D7000- D7FFF	D6000- D6FFF	D5000- D5FFF	D4000- D4FFF	D3000- D3FFF	D2000- D2FFF	D1000- D1FFF	D0000- D0FFF	MTRR_FIX4K_ D0000_REG
DF000- DFFFF	DE000- DEFFF	DD000- DDFFF	DC000- DCFFF	DB000- DBFFF	DA000- DAFFF	D9000- D9FFF	D8000- D8FFF	MTRR_FIX4K_ D8000_REG
E7000- E7FFF	E6000- E6FFF	E5000- E5FFF	E4000- E4FFF	E3000- E3FFF	E2000- E2FFF	E1000- E1FFF	E0000- E0FFF	MTRR_FIX4K_ E0000_REG
EF000- EFFFF	EE000- EEFFF	ED000- EDFFF	EC000- ECFFF	EB000- EBFFF	EA000- EAFFF	E9000- E9FFF	E8000- E8FFF	MTRR_FIX4K_ E8000_REG
F7000- F7FFF	F6000- F6FFF	F5000- F5FFF	F4000- F4FFF	F3000- F3FFF	F2000- F2FFF	F1000- F1FFF	F0000- F0FFF	MTRR_FIX4K_ F0000_REG
FF000- FFFFF	FE000- FEFFF	FD000- FDFFF	FC000- FCFFF	FB000- FBFFF	FA000- FAFFF	F9000- F9FFF	F8000- F8FFF	MTRR_FIX4K_ F8000_REG

11.4.12.4 Variable-Range MTRRs

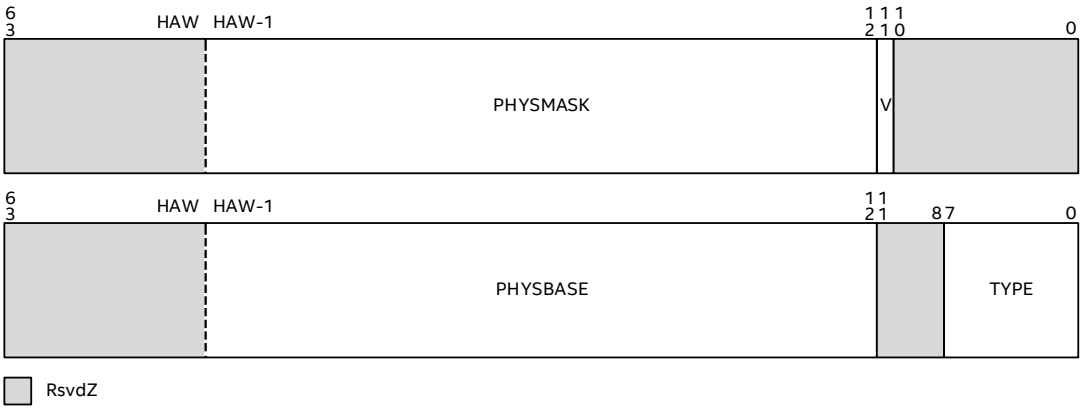


Figure 11-42. Variable-Range MTRR Format

Abbreviation	MTRR_PHYSBASE0_REG, MTRR_PHYSMASK0_REG MTRR_PHYSBASE1_REG, MTRR_PHYSMASK1_REG MTRR_PHYSBASE2_REG, MTRR_PHYSMASK2_REG MTRR_PHYSBASE3_REG, MTRR_PHYSMASK3_REG MTRR_PHYSBASE4_REG, MTRR_PHYSMASK4_REG MTRR_PHYSBASE5_REG, MTRR_PHYSMASK5_REG MTRR_PHYSBASE6_REG, MTRR_PHYSMASK6_REG MTRR_PHYSBASE7_REG, MTRR_PHYSMASK7_REG MTRR_PHYSBASE8_REG, MTRR_PHYSMASK8_REG MTRR_PHYSBASE9_REG, MTRR_PHYSMASK9_REG
General Description	Variable-range Memory Type Range Registers. Each Variable-range MTRR register includes a low 64-bit Base register and a high 64-bit Mask register. VCNT field in MTRRCAP_REG reports number of Variable-range MTRRs supported by hardware. These registers are treated as RsvdZ by implementations reporting Memory Type Support (MTS) as not supported in the Extended Capability register.
Register Offsets	180h, 188h, 190h, 198h, A0h, 1A8h, 1B0h, 1B8h, 1C0h, 1C8h, 1D0h, 1D8h, 1E0h, 1E8h, 1F0h, 1F8h, 200h, 208h, 210h, 218h

Bits	Access	Default	Field	Description
63:HAW	RsvdZ	0h	R: Reserved	Reserved
HAW-1:12	RW	0h	PHYSMASK: Physical Mask	Mask for range
11	RW	0	V: Valid	• 0: Not Valid • 1: Valid
10:0	RsvdZ	0h	R: Reserved	Reserved

Bits	Access	Default	Field	Description
63:HAW	RsvdZ	0h	R: Reserved	Reserved
HAW-1:12	RW	0h	PHYSBASE: Physical Base	Base address of range

Bits	Access	Default	Field	Description
11:8	RsvdZ	0h	R: Reserved	Reserved
7:0	RW	0h	TYPE: Type	Memory type for range

11.4.13 Performance Monitoring Registers

The performance monitoring registers allow software to discover capabilities, configure, and control the performance monitoring facilities. The capability registers include a global performance monitoring capability register, per-event group capabilities, and optionally per-counter capabilities.

Some performance monitoring registers are read-only while counting is enabled. Refer to Table 51 below for details.

Table 51. Performance Monitoring Registers with Conditional Read-only Attributes

Perfmon Register	Conditions under which register is Read-only
PERFCNTRCFG_REG	Fields of a given register with RWL attribute are read-only while the Enable field of that register is 1; read-write otherwise.
PERFCNTR_FLTR_REG	Fields with RWL attribute are read-only while corresponding Counter Configuration Register Enable field is 1 (PERFCNTRCFG_REG.EN=1); read-write otherwise.
PERFCNTR_REG	Fields with RWLV attribute are read-only when the following conditions are satisfied: • The corresponding Counter Configuration Register Enable field is 1 (PERFCNTRCFG_REG.EN=1) • The Counters Writable While Enabled field is 0 in the Performance Monitoring Capability Register (PERFCAP_REG.CWE=0) Fields with RWLV attribute are read-write under all other conditions.

11.4.13.1 Performance Monitoring Register Layout

The register layout of the performance monitoring registers are governed by:

- Performance Monitoring Configuration Offset Register
- Performance Monitoring Freeze Offset Register
- Performance Monitoring Overflow Offset Register
- Performance Monitoring Counter Offset Register
- Counter Stride field in the Performance Monitoring Capability Register

An example register map of the performance monitoring registers is described below in Figure 11-43:

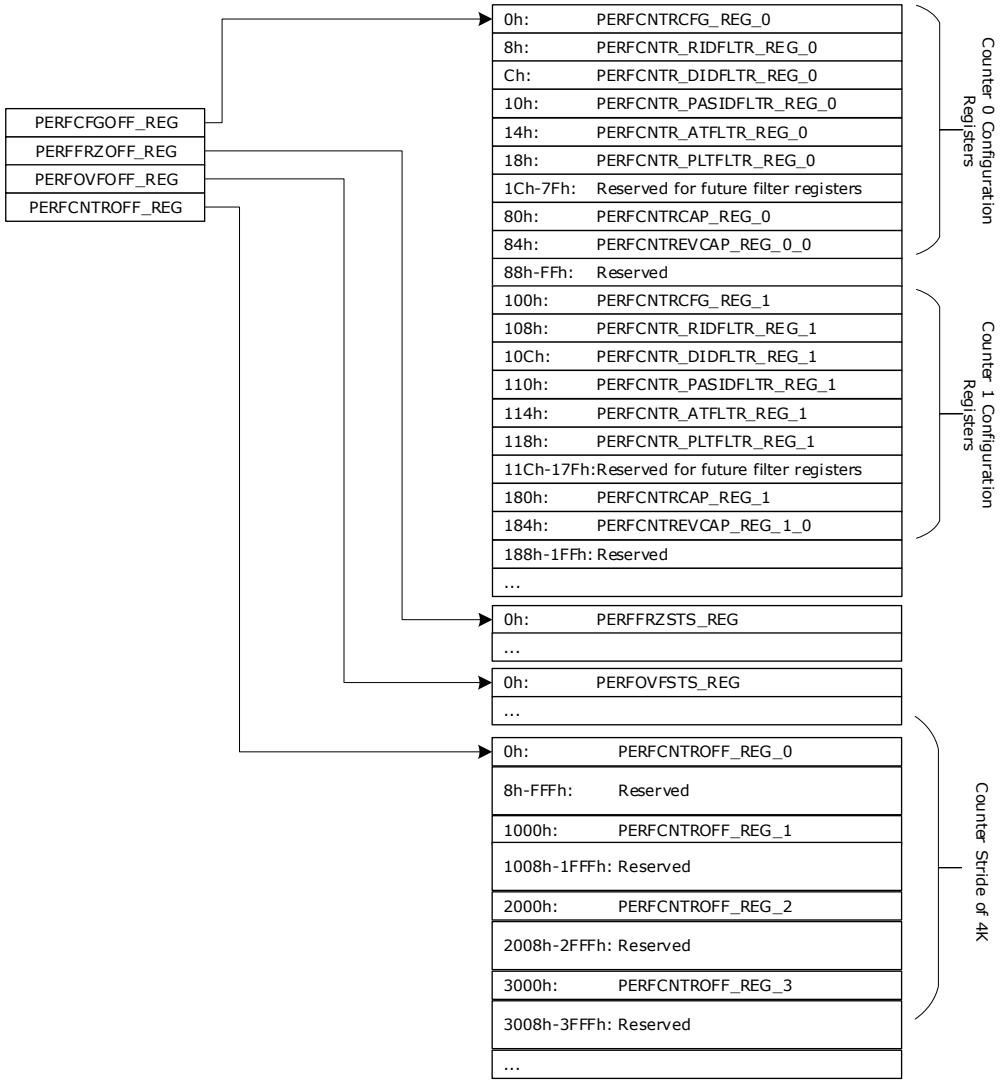


Figure 11-43. Example Register Layout of Performance Monitoring Registers

11.4.13.2 Performance Monitoring Capability Register

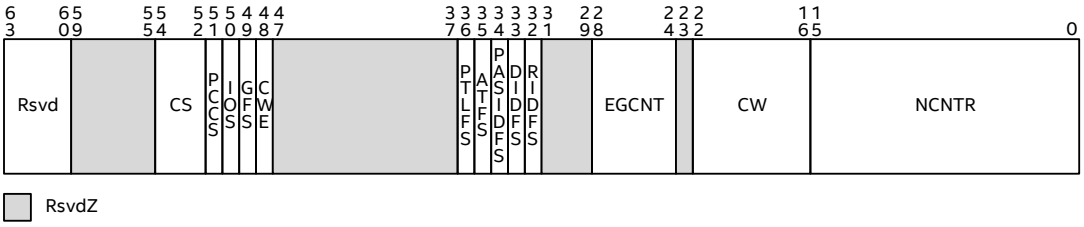


Figure 11-44. Performance Monitoring Capability Register

Abbreviation	PERFCAP_REG
General Description	Register to report performance monitoring hardware capabilities. This register is treated as RsvdZ by implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	300h

Bits	Access	Default	Field	Description
63:60	Rsvd	Fh	R: Reserved	Reserved
59:55	RsvdZ	0h	R: Reserved	Reserved
54:52	RO	X	CS: Counter Stride	Each counter register is equally spaced based on value in this field. A value of X in this field indicates a counter stride of $2^{(X+10)}$ bytes.
51	RO	X	PCCS: Per Counter Capabilities Support	• **0:** Counter Capability registers are not present. All counters have the same capabilities • **1:** Counter Capability registers are present to describe the capabilities of each counter.
50	RO	X	IOS: Interrupt on Overflow Support	• **0:** Device does not support generation of interrupts upon counter overflow. • **1:** Device supports generation of interrupt upon counter overflow. Interrupt generation is controlled by the Interrupt on Overflow bit in the PERFCNTRCFG registers.
49	RO	X	GFS: Global Freeze on Overflow Support	• **0:** Global Freeze on Overflow is not supported. • **1:** Global Freeze on Overflow is supported.
48	RO	X	CWE: Counters Writable while Enabled	• **0:** Indicates that software is not allowed to write to a Counter register while that counter is enabled. Counter registers are always writable while disabled. • **1:** Indicates that hardware supports writes to a Counter register while it is enabled.
47:37	RsvdZ	0h	R: Reserved	Reserved
36	RO	X	PTLFS: Page Table Level Filter Support	• 0: Page Table Level Filter is not supported. • 1: Page Table Level Filter is supported.
35	RO	X	ATFS: Address Type Filter Support	• 0: Address Type Filter is not supported. • 1: Address Type Filter is supported.

Bits	Access	Default	Field	Description
34	RO	X	PASIDFS: PASID Filter Support	• 0: PASID Filter is not supported. • 1: PASID Filter is supported.
33	RO	X	DIDFS: Domain ID Filter Support	• 0: Domain ID Filter is not supported. • 1: Domain ID Filter is supported.
32	RO	X	RIDFS: Requester ID Filter Support	• 0: Requester ID Filter is not supported. • 1: Requester ID Filter is supported.
31:29	RsvdZ	0h	R: Reserved	Reserved
28:24	RO	X	EGCNT: Event Group Count	The value of this field indicates the number of PERFEVNTCAP_REG registers supported, which indicate the supported events for each Event Group.
23	RsvdZ	0h	R: Reserved	Reserved
22:16	RO	X	CW: Counter Width	The value of this field represents the number of bits supported per counter. If the value of this field is N, then each counter is an N-bit counter and the max value it can count is 2^N-1. If per-counter capabilities are supported, the counter width indicated in the PERFCNTRCAP registers overrides this value.
15:0	RO	X	NCNTR: Number of Counters	Indicates the number of counters present in hardware and the associated registers for each.

11.4.13.3 Performance Monitoring Configuration Offset Register

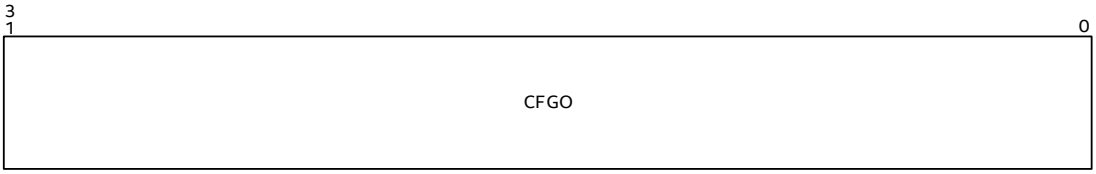


Figure 11-45. Performance Monitoring Configuration Offset Register

Abbreviation	PERFCFGOFF_REG
General Description	Register to indicate the offset of the Counter Configuration and Capability registers from the base of the register space of the remapping hardware unit. The Counter Configuration and Capability registers occupy 256 bytes per counter, therefore bits 7:0 of this register are 0. This register is treated as RsvdZ by implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	310h

Bits	Access	Default	Field	Description
31:0	RO	X	CFGO: Configuration Offset	Offset of Performance Monitoring Counter Configuration Registers from the Remapping Hardware Register Base Address.

11.4.13.4 Performance Monitoring Freeze Offset Register

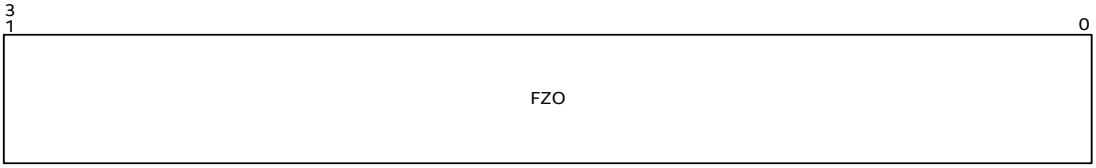


Figure 11-46. Performance Monitoring Freeze Offset Register

Abbreviation	PERFFRZOFF_REG
General Description	Register to indicate the offset of the Freeze Status Registers from the base of the register space of the remapping hardware unit. The Freeze Status Registers are 8-byte aligned, therefore bits 2:0 of this register are 0. This register is treated as RsvdZ by implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	314h

Bits	Access	Default	Field	Description
31:0	RO	X	FZO: Freeze Offset	Offset of Performance Monitoring Freeze Status Registers from the Remapping Hardware Register Base Address.

11.4.13.5 Performance Monitoring Overflow Offset Register

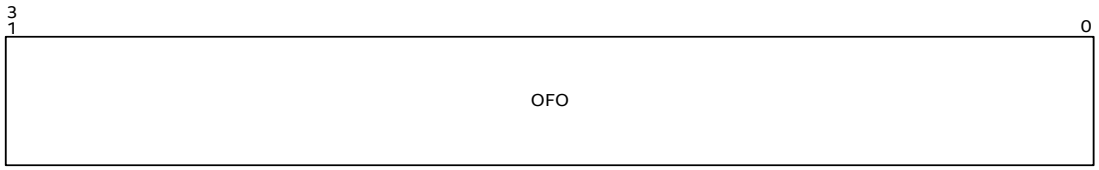


Figure 11-47. Performance Monitoring Overflow Offset Register

Abbreviation	PERFOVFOFF_REG
General Description	Register to indicate the offset of the Overflow Status registers from the base of the register space of the remapping hardware unit. The Overflow Status registers are 8-byte aligned, therefore bits 2:0 of this register are 0. This register is treated as RsvdZ by implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	318h

Bits	Access	Default	Field	Description
31:0	RO	X	OFO: Overflow Offset	Offset of Performance Monitoring Overflow Status registers from the Remapping Hardware Register Base Address.

11.4.13.6 Performance Monitoring Counter Offset Register

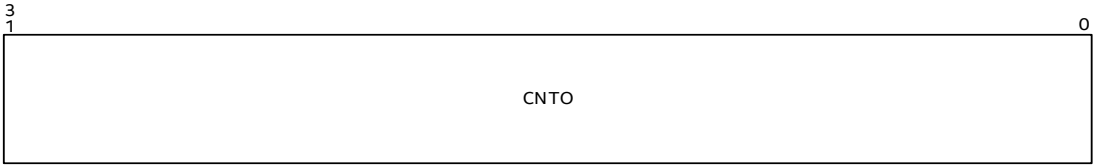


Figure 11-48. Performance Monitoring Counter Offset Register

Abbreviation	PERFCNTROFF_REG
General Description	Register to indicate the offset of the Counter registers from the base of the register space of the remapping hardware unit. The Counter registers are aligned based on the Counter Stride value reported in PERFCAP_REG.CS, therefore the least significant bits corresponding to the aligned size are 0. This register is treated as RsvdZ by implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	31Ch

Bits	Access	Default	Field	Description
31:0	RO	X	CNTO: Counter Offset	Offset of Performance Monitoring Counter Registers from the Remapping Hardware Register Base Address.

11.4.13.7 Performance Monitoring Interrupt Status Register

Figure 11-49. Performance Monitoring Interrupt Status Register

Abbreviation	PERFINTRSTS_REG
General Description	Register to indicates whether a performance monitoring event has occurred. This register is treated as RsvdZ by implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	324h

Bits	Access	Default	Field	Description
31:1	RsvdZ	0h	R: Reserved	Reserved
0	RW1C	0h	PIS: Performance Interrupt Status	This field is set to 1 by hardware when a performance monitoring event occurs, specifically when a counter overflows and the corresponding Interrupt on Overflow field in the corresponding Counter Configuration register is 1.

11.4.13.8 Performance Monitoring Interrupt Control Register

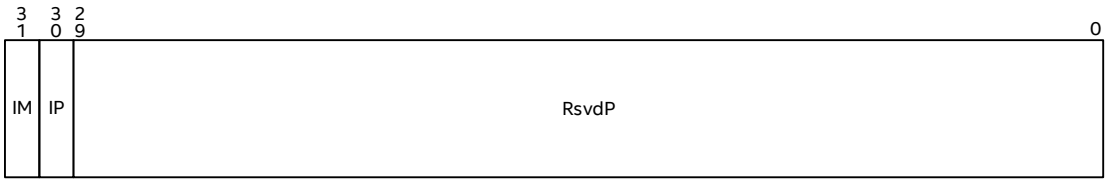


Figure 11-50. Performance Monitoring Interrupt Control Register

Abbreviation	PERFINTRCTL_REG
General Description	Register specifying the performance event interrupt message control bits. This register is treated as RsvdZ by implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	328h

Bits	Access	Default	Field	Description
31	RW	1h	IM: Interrupt Mask	• **0:** Interrupt is not masked. When a performance monitoring event occurs, hardware issues an interrupt message, using the values in the Performance Interrupt Data and Address registers. If this field is set to 0 while Interrupt Pending is 1, the interrupt is delivered immediately. • **1:** Interrupt is masked. Software may mask interrupt message generation by setting this field. Hardware is prohibited from sending the interrupt message when this field is Set.
30	RO	0h	IP: Interrupt Pending	Hardware sets this field when a performance monitoring event occurs and is held pending. The field remains set until the interrupt message pending condition is serviced due to either: • Hardware issuing the interrupt message due to either change in transient hardware condition that caused the interrupt message to be held pending or due to software clearing the IM field. • Software writing to clear the PIS field in the Performance Monitoring Interrupt Status Register.
29:0	RsvdP	Xh	R: Reserved	Reserved

11.4.13.9 Performance Monitoring Interrupt Data Register

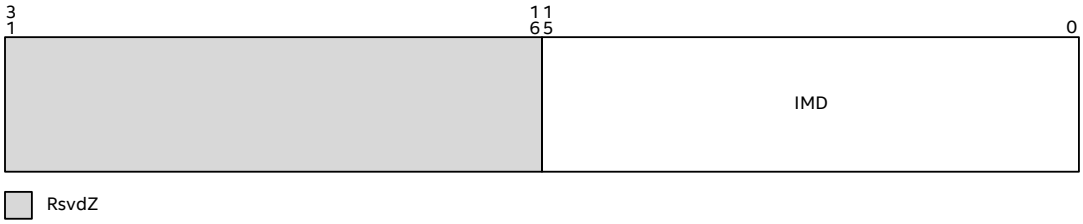


Figure 11-51. Performance Monitoring Interrupt Data Register

Abbreviation	PERFINTRDATA_REG
General Description	Register specifying the Performance Monitoring Interrupt message data. This register is treated as RsvdZ by implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	32Ch

Bits	Access	Default	Field	Description
31:16	RsvdZ	0h	R: Reserved	Reserved
15:0	RW	0h	IMD: Interrupt Message data	Data value in the interrupt request. Software requirements for programming this register are described in Section 5.1.6.

11.4.13.10 Performance Monitoring Interrupt Address Register

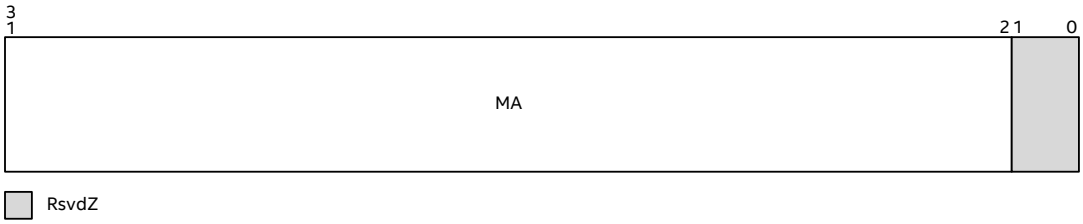


Figure 11-52. Performance Monitoring Interrupt Address Register

Abbreviation	PERFINTRADDR_REG
General Description	Register specifying the lower 32 bits of the message address for performance monitoring events. The address must be 4-byte aligned. This register is treated as RsvdZ by implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	330h

Bits	Access	Default	Field	Description
31:2	RW	0h	MA: Interrupt Message Address	When fault events are enabled, the contents of this register specify the DWORD-aligned address (bits 31:2) for the interrupt request. Software requirements for programming this register are described in Section 5.1.6.
1:0	RsvdZ	0h	R: Reserved	Reserved

11.4.13.11 Performance Monitoring Interrupt Upper Address Register

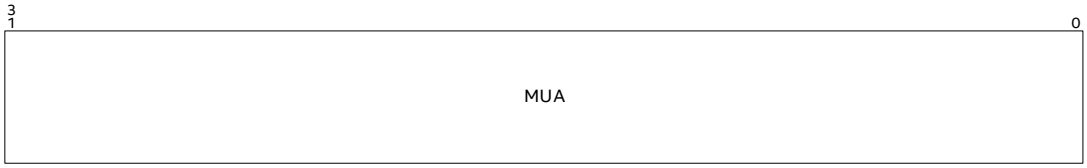


Figure 11-53. Performance Monitoring Interrupt Upper Address Register

Abbreviation	PERFINTRUADDR_REG
General Description	Register specifying the upper 32 bits of the message address for performance monitoring events. This register is treated as RsvdZ by implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	334h

Bits	Access	Default	Field	Description
31:0	RW	0h	MUA: Interrupt Message Upper Address	Hardware implementations supporting Extended Interrupt Mode are required to implement this register. Software requirements for programming this register are described in Section 5.1.6. Hardware implementations not supporting Extended Interrupt Mode may treat this field as RsvdZ.

11.4.13.12 Performance Monitoring Freeze Status Registers

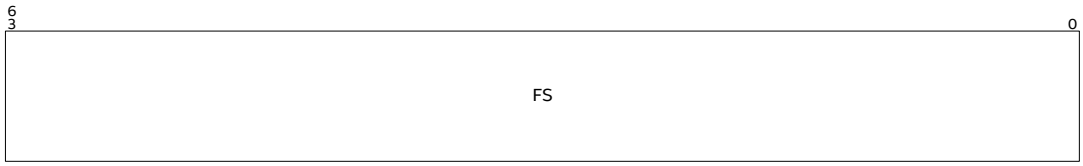


Figure 11-54. Performance Monitoring Freeze Status Registers

Abbreviation	PERFFRZSTS_REG [***m***]
General Description	Register to indicate the freeze status of all the performance monitoring counters. There are multiple Freeze Status Registers in order to provide 1 bit for each counter indicated by the PERFCAP_REG register. The offset of these registers is given by the Freeze Offset register. These registers are not present for implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	***Freeze Offset*** + (***m*** * 8)

Bits	Access	Default	Field	Description
63:0	RO	0h	FS: Freeze State	This register is a bit vector with one bit per counter. • **0:** The counter is currently not frozen. The counter may be disabled, or may be enabled and counting events. • **1:** The counter is currently frozen and not counting events. It remains frozen until explicitly unfrozen by software. The freeze state of the counters can be changed by software using the Enhanced Command Interface. See Table 52 for available commands. Disabling a counter by successfully issuing a Disable Perfmon Counter command to the Enhanced Command Interface clears the freeze status for that counter.

11.4.13.13 Performance Monitoring Overflow Status Registers

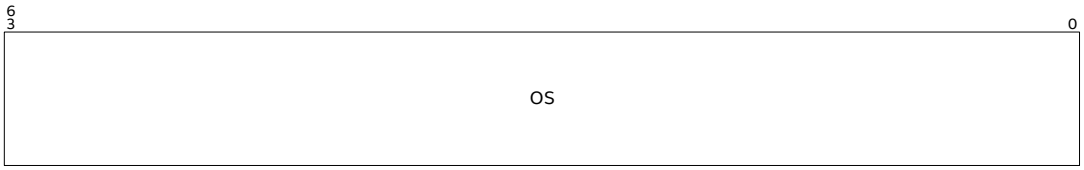


Figure 11-55. Performance Monitoring Overflow Status Registers

Abbreviation	PERFOVFSTS_REG [***m***]
General Description	Register to indicate the overflow status of all the counters. There are ***m*** Overflow status registers to provide 1 bit for each counter indicated by the PERFCAP_REG register. The offset of these registers is given by the Overflow Offset register. These registers are not present for implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	***Overflow Offset*** + (***m*** * 8)

Bits	Access	Default	Field	Description
63:0	RW1C	0h	OS: Overflow Status	This register is a bit vector with one bit per counter. Each bit indicates whether the corresponding performance counter has encountered an overflow condition. • **0:** Counter has not encountered an overflow condition. • **1:** Counter has encountered an overflow condition. Writing a **1** clears the bit.

11.4.13.14 Performance Monitoring Event Capability Register

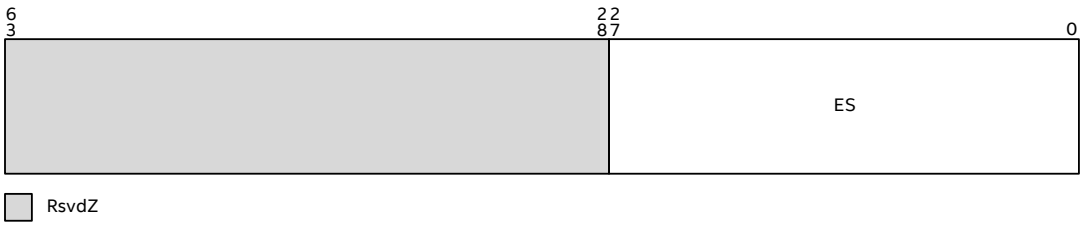


Figure 11-56. Performance Monitoring Event Capability Register

Abbreviation	PERFEVNTCAP_REG [***g***]
General Description	Each of these registers corresponds to an Event Group Index, ***g***, and reports the set of events supported for that Event Group Index. The number of registers corresponds to the number of Event Groups reported in PERFCAP_REG. If the implementation does not support any events in an Event Group Index, the corresponding Events Supported field is 0. The encoding of supported events within each Event Group Index is presented in Table 49. These registers are not present for implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	380h + (***g*** *8)

Bits	Access	Default	Field	Description
63:28	RsvdZ	0h	R: Reserved	Reserved
27:0	RO	X	ES: Events Supported	Indicates the events supported for the corresponding Event Group Index ***g***. Each bit that is 1 indicates that the corresponding event is supported.

11.4.13.15 Performance Monitoring Counter Configuration Registers

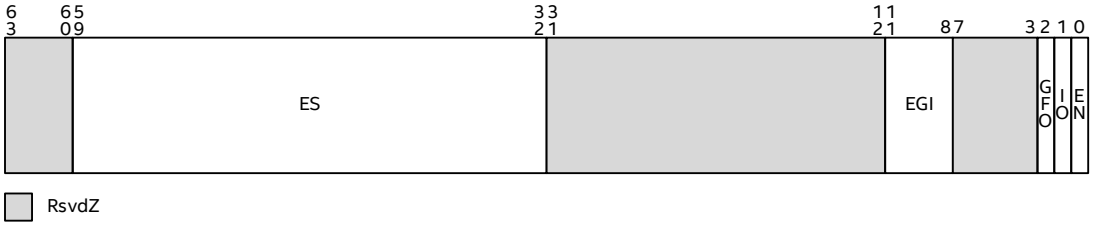


Figure 11-57. Performance Monitoring Counter Configuration Registers

Abbreviation	PERFCNTRCFG_REG [***c***]
General Description	These registers are used to configure the set of events monitored by each counter. They also control interrupt generation behavior and the behavior upon overflow. The number of counter configuration registers corresponds to the number of counter registers indicated in PERFCAP_REG. The default value of these registers is 0. Refer to Table 51 regarding fields within this register with RWL access attributes. These registers are not present for implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	Configuration Offset + (***c*** × 100h)

Bits	Access	Default	Field	Description
63:60	RsvdZ	0h	R: Reserved	Reserved
59:32	RWL	0h	ES: Event Select	Specifies the set of events to be monitored by this counter, corresponding to the Event Group selected. Refer to Section 10.2 for details on how software should program this field.
31:12	RsvdZ	0h	R: Reserved	Reserved
11:8	RWL	0h	EGI: Event Group Index	Specifies the Event Group Index to associate with this counter. Refer to Section 10.2 for details on how software should program this field.
7:3	RsvdZ	0h	R: Reserved	Reserved
2	RWL	0h	GFO: Global Freeze on Overflow	• **0:** No global freeze. • **1:** When an overflow is detected in this register, all counters in the device are frozen. In either case, overflow is recorded in the Overflow Status register. This bit is reserved if Global Freeze on Overflow is not supported for this counter. Refer to Section 10.5 for details.
1	RWL	0h	IO: Interrupt on Overflow	• **0:** No performance monitoring event is generated. • **1:** Generate a performance monitoring event when this counter overflows. An interrupt may be delivered, depending on the configuration of the Performance Interrupt registers. This bit is reserved if Interrupt on Overflow is not supported for this counter. Refer to Section 10.5 for details.

Bits	Access	Default	Field	Description
0	RO	0h	EN: Enable	Hardware updates this bit to indicate if the counter is enabled. • **0:** This counter is disabled. • **1:** This counter is enabled to count events. Software uses the Enable/Disable Perfmon Counter commands through the Enhanced Command Interface to change the enabled status of a counter. See Section 11.4.13 for more details. This field influences Performance Monitoring Registers where RWL and RWLV access attributes apply. Refer to Table 51 for registers affected by this field.

11.4.13.16 Performance Monitoring Requester ID Filter Configuration Registers

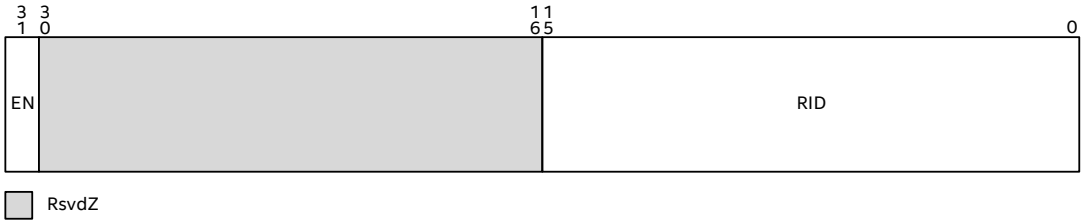


Figure 11-58. Performance Monitoring Requester ID Filter Configuration Registers

Abbreviation	PERFCNTR_RIDFLTR_REG [***c***]
General Description	These registers are used to configure the Requester ID filter for each counter. There exists exactly one instance of this register per counter. Refer to Table 51 regarding fields within this register with RWL access attributes. This register is treated as RsvdZ for implementations reporting Requester ID Filter (RIDFS) as not supported in the Performance Monitoring Capability register. These registers are not present for implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	***Configuration Offset*** + (***c*** × 100h) + 8h

Bits	Access	Default	Field	Description
31	RWL	0h	EN: Enable	• **0:** This filter is ignored. • **1:** Counter will increment only for selected event occurrences where the associated Requester ID matches the value in the RID field.
30:16	RsvdZ	0h	R: Reserved	Reserved
15:0	RWL	0h	RID: Requester ID	The Requester ID value used to filter event occurrences.

11.4.13.17 Performance Monitoring Domain ID Filter Configuration Registers

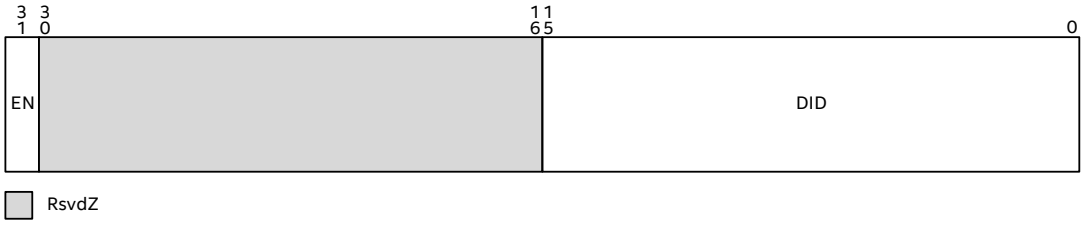


Figure 11-59. Performance Monitoring Domain ID Filter Configuration Registers

Abbreviation	PERFCNTR_DIDFLTR_REG [***c***]
General Description	These registers are used to configure the Domain ID filter for each counter. There exists exactly one instance of this register per counter. Refer to Table 51 regarding fields within this register with RWL access attributes. This register is treated as RsvdZ for implementations reporting Domain ID Filter (DIDFS) as not supported in the Performance Monitoring Capability register. These registers are not present for implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	***Configuration Offset*** + (***c*** × 100h) + Ch

Bits	Access	Default	Field	Description
31	RWL	0h	EN: Enable	• **0:** This filter is ignored. • **1:** Counter will increment only for selected event occurrences where the associated Domain ID matches the value in the DID field.
30:16	RsvdZ	0h	R: Reserved	Reserved
15:0	RWL	0h	DID: Domain ID	The Domain ID value used to filter event occurrences.

11.4.13.18 Performance Monitoring PASID Filter Configuration Registers

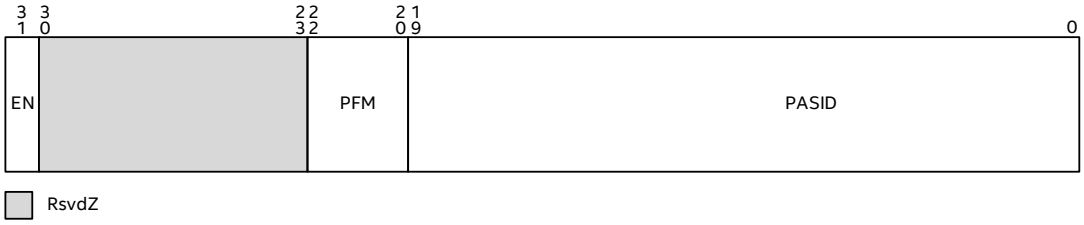


Figure 11-60. Performance Monitoring PASID Filter Configuration Registers

Abbreviation	PERFCNTR_PASIDFLTR_REG [***c***]
General Description	These registers are used to configure the PASID filter for each counter. There exists exactly one instance of this register per counter. Refer to Table 51 regarding fields within this register with RWL access attributes. This register is treated as RsvdZ for implementations reporting PASID Filter (PASIDFS) as not supported in the Performance Monitoring Capability register. These registers are not present for implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	***Configuration Offset*** + (***c*** × 100h) + 10h

Bits	Access	Default	Field	Description
31	RWL	0h	EN: Enable	• **0:** This filter is ignored. • **1:** Counter will increment only when the condition selected by PASID Filter Mode is satisfied.
30:22	RsvdZ	0h	R: Reserved	Reserved
21:20	RWL	0h	PFM: PASID Filter Mode	• **00b:** Occurrences related to requests-with-PASID that match the PASID field • **01b:** Occurrences related to requests-with-PASID with any PASID value • **10b:** Occurrences related to requests-without-PASID • **11b:** Reserved
19:0	RWL	0h	PASID: PASID Value	The PASID value used to filter event occurrences when PFM=00b.

11.4.13.19 Performance Monitoring Address Type Filter Configuration Registers

Figure 11-61. Performance Monitoring Address Type Filter Configuration Registers

Abbreviation	PERFCNTR_ATFLTR_REG [***c***]
General Description	These registers are used to configure the Address Type filter for each counter. There exists exactly one instance of this register per counter. Refer to Table 51 regarding fields within this register with RWL access attributes. This register is treated as RsvdZ for implementations reporting Address Type Filter (ATFS) as not supported in the Performance Monitoring Capability register. These registers are not present for implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	***Configuration Offset*** + (***c*** × 100h) + 14h

Bits	Access	Default	Field	Description
31	RWL	0h	EN: Enable	• **0:** This filter is ignored. • **1:** Counter will increment only for selected event occurrences that matches one or more of the conditions selected in the AT field.
30:5	RsvdZ	0h	R: Reserved	Reserved
4:0	RWL	0h	AT: Address Type	Bitmap selecting requests that match one or more of the following: • **0:** Untranslated requests (AT=00b) • **1:** Translation requests (AT=01b) • **2:** Translated requests (AT=10b) • **3:** Reserved • **4**: HPT prefetch Implementations not supporting one or more types of requests may treat the associated bit in this field as Reserved(0).

11.4.13.20 Performance Monitoring Page Table Level Filter Configuration Registers

Figure 11-62. Performance Monitoring Page Table Level Filter Configuration Registers

Abbreviation	PERFCNTR_PTLFLTR_REG [***c***]
General Description	These registers are used to configure the Page Table Level filter for each counter. There exists exactly one instance of this register per counter. Refer to Table 51 regarding fields within this register with RWL access attributes. This register is treated as RsvdZ for implementations reporting Page Table Level Filter (PTLFS) as not supported in the Performance Monitoring Capability register. These registers are not present for implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	***Configuration Offset*** + (***c*** × 100h) + 18h

Bits	Access	Default	Field	Description
31	RWL	0h	EN: Enable	• **0:** This filter is ignored. • **1:** Counter will increment only for selected event occurrences that match one or more of the conditions selected in the PTL field.
30:5	RsvdZ	0h	R: Reserved	Reserved
4:0	WO	0h	PTL: Page Table Level	Bitmap selecting event occurrences pertaining to one or more of the following page table entries types or page sizes:

Bit Value	SS/FS Entries	HPT Entries
0	PTE (4k Pages)	HPTL1E
1	PDE (2M Pages)	HPTL2E
2	PDPE (1G Pages)	HPTL3E
3	PML4E	HPTL4E
4	PML5E	Reserved

11.4.13.21 Performance Monitoring Counter Capability Register

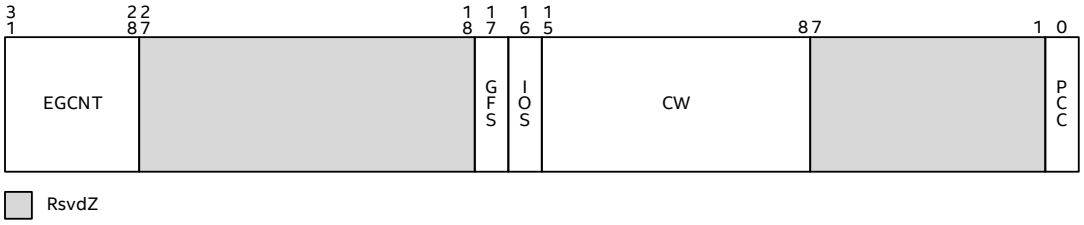


Figure 11-63. Performance Monitoring Counter Capability Register

Abbreviation	PERFCNTRCAP_REG [***c***]
General Description	This set of registers is provided only on implementations that support different sets of events for different counters. They are present only if the Per Counter Capabilities Supported field in PERFCAP_REG is 1. If present, the number of these capability registers corresponds to the number of counters reported in PERFCAP_REG. The values reported in each capability register apply only to the corresponding counter and override the values reported in PERFCAP_REG if PCC is 1. These registers are not present for implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	***Configuration Offset*** + (***c*** × 100h) + 0x80

Bits	Access	Default	Field	Description
31:28	RO	X	EGCNT: Event Group Count	This field indicates the number of corresponding Counter Event Capability Registers describing the Event Groups and Events supported by this counter register. For example, if a given counter can only counts events corresponding to a single Event Group, then this field will be 1 and there will be 1 Counter Event Capability Register.
27:18	RsvdZ	0h	R: Reserved	Reserved
17	RO	X	GFS: Global Freeze on Overflow Support	Indicates whether the freeze capability is supported when an overflow condition occurs. • **0:** Global Freeze on Overflow is not supported. • **1:** Global Freeze on Overflow is supported.
16	RO	X	IOS: Interrupt on Overflow Support	Indicates whether the interrupt capability is supported when an overflow condition occurs. • **0:** Interrupt Capability is not supported. • **1:** Interrupt Capability is supported.
15:8	RO	X	CW: Counter Width	The value of this field represents the number of bits supported for this counter. If the value of this field is N, then the counter is an N-bit counter and the maximum value it can count is 2^N-1.
7:1	RsvdZ	0h	R: Reserved	Reserved
0	RO	X	PCC: Per-Counter Capabilities	• **0:** This counter supports the capabilities reported by PERFCAP_REG and PERFEVNTCAP_REG. • **1:** This counter supports the capabilities described in this register and in the Counter Event Capability Registers.

11.4.13.22 Performance Monitoring Counter Event Capability Registers

Figure 11-64. Performance Monitoring Counter Event Capability Registers

Abbreviation	PERFCNTREVCAP_REG [c,e_c]
General Description	These registers report the Event Groups and Events supported by each counter. They are present only if the Per Counter Capabilities Supported field in PERFCAP_REG is 1. The values reported in each register apply only to the corresponding counter and only if the PCC field in the corresponding PERFCNTRCAP_REG is 1. These registers are not present for implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	***Configuration Offset*** + (c × 100h) + 84h + (e_c × 4)

Bits	Access	Default	Field	Description
31:28	RO	X	EGI: Event Group Index	Indicates the index of the Event Group that can be enabled in this counter register. This field is treated as RsvdZ is the corresponding Per-Counter Capabilities field
27:0	RO	X	ES: Event Support	Indicates the Events supported in this counter register for the Event Group specified by Event Group Index

11.4.13.23 Performance Monitoring Counter Registers

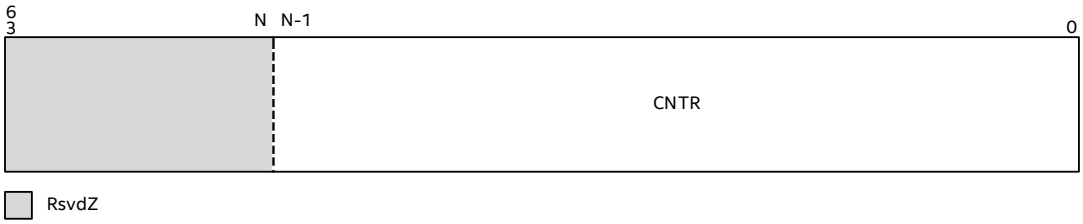


Figure 11-65. Performance Monitoring Counter Registers

Abbreviation	PERFCNTR_REG [***c***]
General Description	Each Counter register is an N-bit counter that is used to count occurrences of configured events. N is the value of the Counter Width field in PERFCAP_REG or PERFCNTRCAP_REG. Behavior of software reads and writes to these registers are described in section Section 10.3. Once written, the counter continues to increment from the written value. A freeze operation causes the counter to stop accumulating further events and to retain its value at the time of freeze. An unfreeze operation allows the counter to resume counting subsequent events. Each counter is equally spaced and aligned based on the Counter Stride field (PERFCAP_REG.CS). Refer to Table 51 regarding fields within this register with RWLV access attributes. These registers are not present for implementations reporting Performance Monitoring (PMS) as not supported in the Extended Capability register.
Register Offset	***Counter Offset*** + (***c*** × Counter Stride)

Bits	Access	Default	Field	Description
63:N	RsvdZ	0h	R: Reserved	Reserved
N-1:0	RWLV	0h	CNTR: Counter Value	N-bit performance counter where N is the value of the Counter Width field in PERFCAP_REG or corresponding PERFCNTRCAP_REG when the Per-Counter Capabilities field is Set.

11.4.14 Enhanced Command Interface

11.4.14.1 Enhanced Command Register

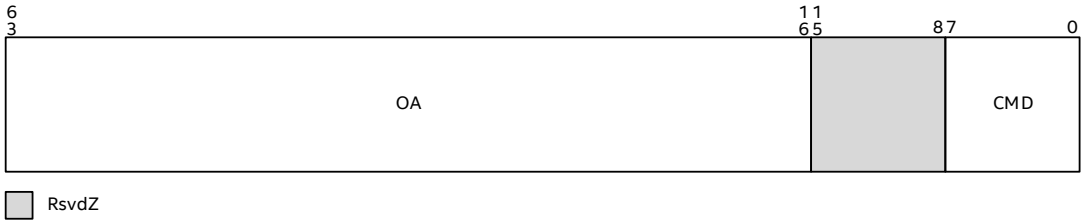


Figure 11-66. Enhanced Command Register

Abbreviation	ECMD_REG
General Description	Register to submit command and operand of enhanced commands to DMA Remapping hardware. 32-bit software must write the upper 32-bits in the operand first, prior to writing the lower 32-bits including the command field. If Operand B is required to issue a command, then the Enhanced Command Extended Operand Register (ECEO_REG) must be written prior to this register. This register is treated as RsvdZ by implementations reporting Enhanced Command Support (ECMDS) as not supported in the Capability register.
Register Offset	400h

Bits	Access	Default	Field	Description
63:16	WO	0h	OA: Operand A	Operand specification is command-specific. See Table 52 below for details. Writes to this field are dropped while the In Progress field in the Enhanced Command Response Register (ECRSP_REG) is Set.
15:8	RsvdZ	0h	R: Reserved	Reserved
7:0	WO	0h	CMD: Command	See Table 52 for details. Writes to this field are dropped while the In Progress field in the Enhanced Command Response Register (ECRSP_REG) is Set. Writes to this field clear the Enhanced Command Response Register (ECRSP_REG) and set the In Progress field of ECRSP_REG.

Table 52. Enhanced Command Descriptions

Command Name	Command (ECMD_REG[7:0])	Operand A ECMD_REG[63:16]	Operand B ECEO_REG[63:0]
Reserved	0	Reserved	Reserved
Enable Perfmon Counter	240	Counter Number	Unused
Disable Perfmon Counter	241	Counter Number	Unused
Reset All Perfmon Counter Configuration	242	Unused	Unused

Command Name	Command (ECMD_REG[7:0])	Operand A ECMD_REG[63:16]	Operand B ECEO_REG[63:0]
Reset All Perfmon Counter Values	243	Unused	Unused
Freeze All Perfmon Counters	244	Unused	Unused
Unfreeze All Perfmon Counters	245	Unused	Unused
Reserved	246-255	Reserved	Reserved

11.4.14.2 Enhanced Command Extended Operand Register

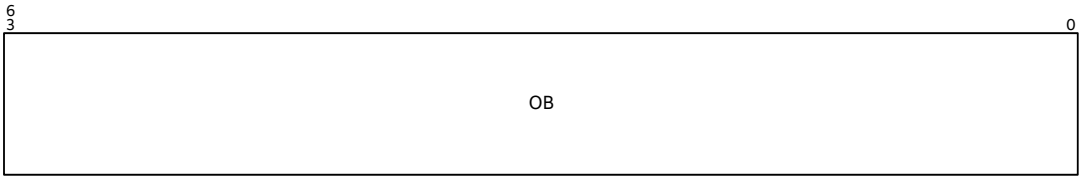


Figure 11-67. Enhanced Command Extended Operand Register

Abbreviation	ECEO_REG
General Description	Register to submit additional operands of enhanced commands to DMA Remapping hardware. This register is treated as RsvdZ by implementations reporting Enhanced Command Support (ECMDS) as not supported in the Capability register.
Register Offset	408h

Bits	Access	Default	Field	Description
63:0	WO	0h	OB: Operand B	Operand specification is command-specific. See Table 52 for details. Writes to this field are dropped while the In Progress field in the Enhanced Command Response Register (ECRSP_REG) is Set.

11.4.14.3 Enhanced Command Response Register

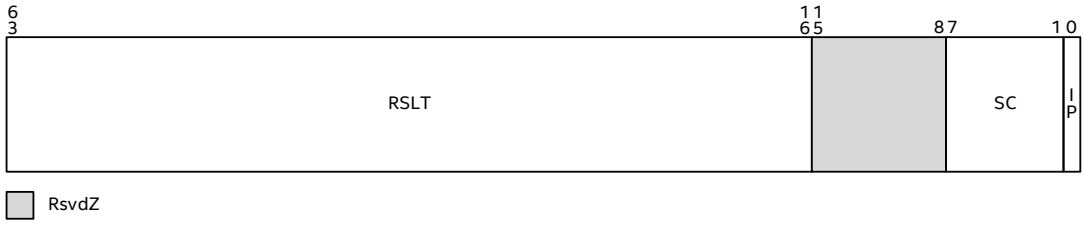


Figure 11-68. Enhanced Command Response Register

Abbreviation	ECRSP_REG
General Description	Register to report status and results for enhanced commands submitted to DMA Remapping hardware. This register is treated as RsvdZ by implementations reporting Enhanced Command Support (ECMDS) as not supported in the Capability register.
Register Offset	410h

Bits	Access	Default	Field	Description
63:16	RO	0h	RSLT: Result	Result specification is command-specific. See Table 53 for details.
15:8	RsvdZ	0h	R: Reserved	Reserved.
7:1	RO	0h	SC: Status Code	• **0:** Command successful without any error. • **1:** Undefined command. • **2:** Command Abort due to in-flight command from GCMD/PMR Registers. • **3-15:** Reserved. • **16-127:** All other Status Codes are command-specific. See Table 53 below for details. Usage of any command that is reported as unsupported by the Enhanced Command Capability Register will return Undefined Command.
0	RO	0h	IP: In Progress	• **0:** Command has been completed. • **1:** Command is in progress.

Table 53. Enhanced Command Response Descriptions

Command Name	Command (ECMD_REG[7:0])	Command Specific Status Code (ECRSP_REG[7:1])	Result (ECRSP_REG[63:16])
Reserved	0	N/A	Reserved
Enable Perfmon Counter	240	• 16: Enable counter failed due to error checks.	Reserved
Disable Perfmon Counter	241	• 16: Disable counter failed due to invalid counter number	Reserved

Command Name	Command (ECMD_REG[7:0])	Command Specific Status Code (ECRSP_REG[7:1])	Result (ECRSP_REG[63:16])
Reset All Perfmon Counter Configuration	242	N/A	Reserved
Reset All Perfmon Counter Values	243	N/A	Reserved
Freeze All Perfmon Counters	244	N/A	Reserved
Unfreeze All Perfmon Counters	245	N/A	Reserved
Reserved	246-255	N/A	Reserved

11.4.14.4 Enhanced Command Status Registers

11.4.14.4.1 Enhanced Command Status Register 0

Figure 11-69. Enhanced Command Status Register 0

Abbreviation	ECSTS_REG_0
General Description	Register reporting hardware state of DMA Remapping hardware for enhanced commands issued through the Enhanced Command Interface. This register is treated as RsvdZ by implementations reporting Enhanced Command Support (ECMDS) as not supported in the Capability register.
Register Offset	420h

Bits	Access	Default	Field	Description
63:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.14.4.2 Enhanced Command Status Register 1

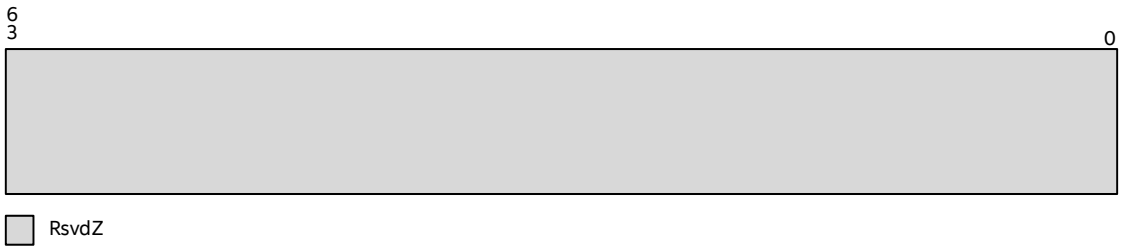


Figure 11-70. Enhanced Command Status Register 1

Abbreviation	ECSTS_REG_1
General Description	Register reporting hardware state of DMA Remapping hardware for enhanced commands issued through the Enhanced Command Interface. This register is treated as RsvdZ by implementations reporting Enhanced Command Support (ECMDS) as not supported in the Capability register.
Register Offset	428h

Bits	Access	Default	Field	Description
63:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.14.5 Enhanced Command Capability Registers

11.4.14.5.1 Enhanced Command Capability Register 0

Figure 11-71. Enhanced Command Capability Register 0

Abbreviation	ECCAP_REG_0
General Description	Register specifying the enhanced commands supported by DMA Remapping hardware. This register covers enhanced commands 0-63. This register is treated as RsvdZ by implementations reporting Enhanced Command Support (ECMDS) as not supported in the Capability register.
Register Offset	430h

Bits	Access	Default	Field	Description
63:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.14.5.2 Enhanced Command Capability Register 1

Figure 11-72. Enhanced Command Capability Register 1

Abbreviation	ECCAP_REG_1
General Description	Register specifying the enhanced commands supported by DMA Remapping hardware. This register covers enhanced commands 64-127. This register is treated as RsvdZ by implementations reporting Enhanced Command Support (ECMDS) as not supported in the Capability register.
Register Offset	438h

Bits	Access	Default	Field	Description
63:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.14.5.3 Enhanced Command Capability Register 2

Figure 11-73. Enhanced Command Capability Register 2

Abbreviation	ECCAP_REG_2
General Description	Register specifying the enhanced commands supported by DMA Remapping hardware. This register covers enhanced commands 128-191. This register is treated as RsvdZ by implementations reporting Enhanced Command Support (ECMDS) as not supported in the Capability register.
Register Offset	440h

Bits	Access	Default	Field	Description
63:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.14.5.4 Enhanced Command Capability Register 3

Figure 11-74. Enhanced Command Capability Register 3

Abbreviation	ECCAP_REG_3
General Description	Register specifying the enhanced commands supported by DMA Remapping hardware. This register covers enhanced commands 192-255. This register is treated as RsvdZ by implementations reporting Enhanced Command Support (ECMDS) as not supported in the Capability register.
Register Offset	448h

Bits	Access	Default	Field	Description
63:54	RsvdZ	0h	R: Reserved	Reserved.
53	RO	X	UFCNTS: Unfreeze All Perfmon Counters Support	• **0:** Unfreeze All Perfmon Counters Command not supported • **1:** Unfreeze All Perfmon Counters Command supported
52	RO	X	FCNTS: Freeze All Perfmon Counters Support	• **0:** Freeze All Perfmon Counters Command not supported • **1:** Freeze All Perfmon Counters Command supported
51	RO	X	RCNTS: Reset All Perfmon Counter Values Support	• **0:** Reset All Perfmon Counter Values not supported • **1:** Reset All Perfmon Counter Values supported
50	RO	X	RCFGS: Reset All Perfmon Counter Configuration Support	• **0:** Reset All Perfmon Counter Configuration Command not supported • **1:** Reset All Perfmon Counter Configuration Command supported
49	RO	X	DCNTS: Disable Perfmon Counter Support	• **0:** Disable Perfmon Counter Command not supported • **1:** Disable Perfmon Counter Command supported
48	RO	X	ECNTS: Enable Perfmon Counter Support	• **0:** Enable Perfmon Counter Command not supported • **1:** Enable Perfmon Counter Command supported
47:0	RsvdZ	0h	R: Reserved	Reserved.

11.4.15 RDT Configuration Register

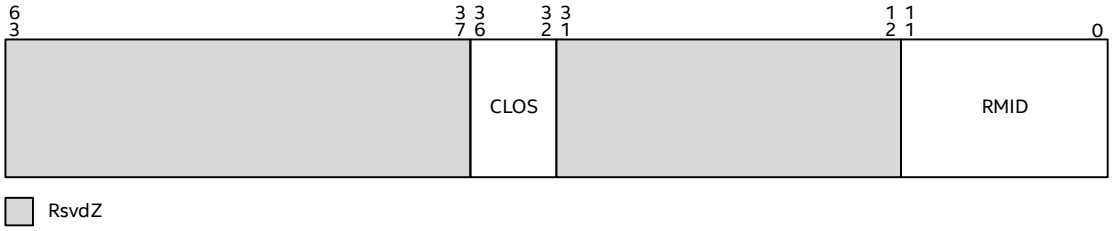


Figure 11-75. RDT Configuration Register

Abbreviation	RDTC_REG
General Description	Register for software control of RMID and CLOS associated with accesses to memory by remapping hardware. Refer to documentation of Intel® Resource Director Technology for usage of this register. Software must determine the maximum supported values of each field in this register. Values programmed to this register above the maximum supported values result in undefined behavior. This register is treated as RsvdZ by implementations reporting RDT Configuration Support (RDTS) as not supported in the Extended Capability register.
Register Offset	DC0h

Bits	Access	Default	Field	Description
63:37	RsvdZ	0h	R: Reserved	Reserved.
36:32	RW	X	CLOS: Class of Service	This field determines the class of service associated with accesses to system memory by remapping hardware.
31:12	RsvdZ	0h	R: Reserved	Reserved.
11:0	RW	X	RMID: Resource Monitoring ID	This field determines the resource monitoring ID associated with accesses to system memory by remapping hardware.

11.4.16 Virtual Command Interface

11.4.16.1 Virtual Command Register

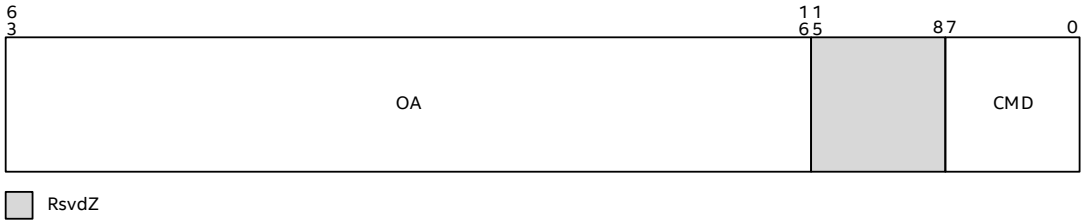


Figure 11-76. Virtual Command Register

Abbreviation	VCMD_REG
General Description	Register to submit command and operand of virtual commands to virtual DMA Remapping hardware. This register is treated as RsvdZ by implementations reporting Virtual Command Support (VCS) as not supported in the Extended Capability register.
Register Offset	E00h

Bits	Access	Default	Field	Description
63:16	RW	X	OA: Operand A	Operand specification is command-specific. See Table 54 below for details. Writes to this field are dropped while the In Progress field in the Virtual Command Response Register (VCRSP_REG) is Set.
15:8	RsvdZ	0h	R: Reserved	Reserved
7:0	RW	X	CMD: Command	See Table 54 for details. Writes to this field are dropped while the In Progress field in the Virtual Command Response Register (VCRSP_REG) is Set. Write to this field clear the Virtual Command Response Register (VCRSP_REG) and set the In Progress field of the VCRSP_REG.

Table 54. Virtual Command Descriptions

Command Name	Command (VCMD_REG[7:0])	Operand A VCMD_REG[63:16]	Operand B VCMD_EO_REG[63:0]
Null Command	0	Unused	Unused
Allocate PASID	1	Unused	Unused
Free PASID	2	63:36 - Unused 35:16 - PASID	Unused
Reserved	3 - 255	Reserved	Reserved

11.4.16.2 Virtual Command Extended Operand Register

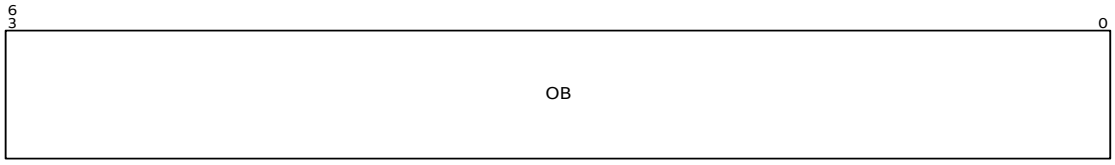


Figure 11-77. Virtual Command Register

Abbreviation	VCMD_EO_REG
General Description	Register to submit additional operands of virtual commands to virtual DMA Remapping hardware. This register is treated as RsvdZ by implementations reporting Virtual Command Support (VCS) as not supported in the Extended Capability register.
Register Offset	E08h

Bits	Access	Default	Field	Description
63:0	RW	X	OB: Operand B	Operand specification is command-specific. See Table 54 for details. Writes to this field are dropped while the In Progress field in the Virtual Command Response Register (VCRSP_REG) is Set.

11.4.16.3 Virtual Command Response Register

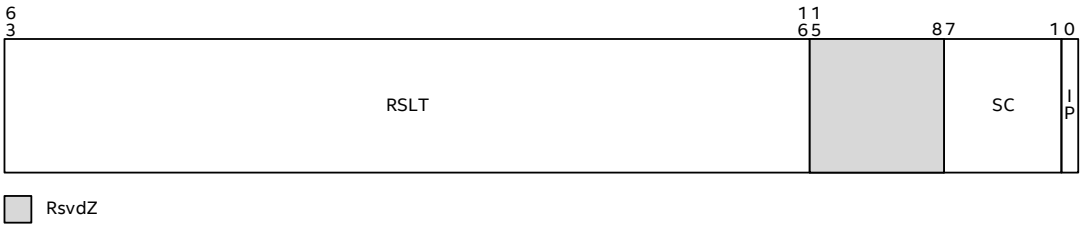


Figure 11-78. Virtual Command Response Register

Abbreviation	VCRSP_REG
General Description	Register to report status and results for virtual commands submitted to virtual-DMA Remapping hardware. This register is treated as RsvdZ by implementations reporting Virtual Command Support (VCS) as not supported in the Extended Capability register.
Register Offset	E10h

Bits	Access	Default	Field	Description
63:16	RO	0h	RSLT: Result	Result specification is command-specific. See Table 55 for details.
15:8	RsvdZ	0h	R: Reserved	Reserved.
7:1	RO	0h	SC: Status Code	• 0: Command successful without any error. • 1: Undefined command. • 2-15: Reserved. • 16-127: All other Status Codes are command-specific. See Table 55 below for details. • Usage of any command that is reported as unsupported by the Virtual Command Capability Register will return Undefined Command.
0	RO	0h	IP: In Progress	• 0: Command has been completed. • 1: Command is in progress.

Table 55. Virtual Command Response Description

Command Name	Command (VCMD_REG[7:0])	Command Specific Status Code (VCRSP_REG[7:1])	Result (VCRSP_REG[63:16])
Null Command	0	N/A	• 63:16 - RsvdZ
Allocate PASID	1	16: No PASID available	• 63:36 - RsvdZ • 35:16 - PASID The contents of this field are treated as reserved when the status code is non-zero.
Free PASID	2	16: Invalid PASID	• 63:16 - RsvdZ
Reserved	3 - 255	N/A	• 63:16 - RsvdZ

11.4.16.4 Virtual Command Capability Register

Figure 11-79. Virtual Command Capability Register

Abbreviation	VCCAP_REG
General Description	Register specifying the virtual commands supported by virtual DMA Remapping hardware. This register is treated as RsvdZ by implementations reporting Virtual Command Support (VCS) as not supported in the Extended Capability register.
Register Offset	E30h

Bits	Access	Default	Field	Description
63:3	RsvdZ	0h	R: Reserved	Reserved.
2	RO	X	FPS: Free PASID Support	• 0: Free PASID commands are not supported. • 1: Free PASID commands are supported. If Set, software must use the Virtual Command Register interface to free PASIDs. Hardware implementations reporting Process Address Space ID Support (PASID) field in the Extended Capability Register as Clear also report this field as Clear.
1	RO	X	PAS: PASID Allocation Support	• 0: PASID allocate commands are not supported. • 1: PASID allocate commands are supported. If Set, software must use the Virtual Command Register interface to allocate PASIDs. Hardware implementations reporting Process Address Space ID Support (PASID) field in the Extended Capability Register as Clear also report this field as Clear.
0	RO	X	NS: Null Command Support	• 0: Null commands are not supported. • 1: Null commands are supported.

Appendix A Snoop and Memory Type for Various Structures

Translation Structure Operation	Snoop	Memory Type
DMA Remapping Structures		
Read of Root-table/ scalable-mode Root-table	ECAP.C	WB
Read of Context-table/ scalable-mode Context-table	ECAP.C	WB
Read of scalable-mode PASID-directory	ECAP.C	WB
Read of scalable-mode PASID-table	ECAP.C	WB
Read of first-stage table	PASID-table-entry.PWSNP	WB
Atomic update of first-stage table	1	WB
Read of second-stage table	• Legacy mode: ECAP.C • Scalable mode: PASID-table-entry.PWSNP	• Legacy mode: WB • Scalable mode: WB
Atomic update of second-stage table	1	WB
Read of page-frame	Table 6	Table 9
Interrupt Remapping Structures		
Read of Interrupt Remap Table	ECAP.C	WB
Atomic update of Posted Interrupt Descriptor	1	WB
Command Queues		
Read of Invalidation Queue	1	WB
Write of Page Request Queue	1	WB
Status Write from Invalidation Wait Descriptor (*inv_wait_dsc*)	1	WB