Remote Action Request

White Paper

July 2021

Revision 1.0

Document Number: 341431-001US

Notice: This document contains information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure. Intel does not assume any liability for lost or stolen data or systems or any damages resulting from such losses.

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.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document. 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.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

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.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel, the Intel logo, and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others

Contents

Figures

Tables

Revision History

Revision Number	Description	Date
1.0	• Initial release of the document.	July 2021

Glossary

Abbreviation	Description
RAR	Remote Action Request
ILP	Initiator Logical Processor
RLP	Receiving Logical Processor
TLB	Translation Lookaside Buffer
ICR	Interrupt Command Register
MBZ	Must Be Zero

1 Introduction

We are interested in receiving feedback on this feature, which you can provide by visiting this site: www.intel.com/sdm-experimental. We will read your feedback and reply to comments and questions.

Remote Action Request (RAR) is introduced in Intel Architecture as a model-specific feature to speed up inter-processor operations by moving parts of those operations from software (OS, App) to hardware (the IA core).

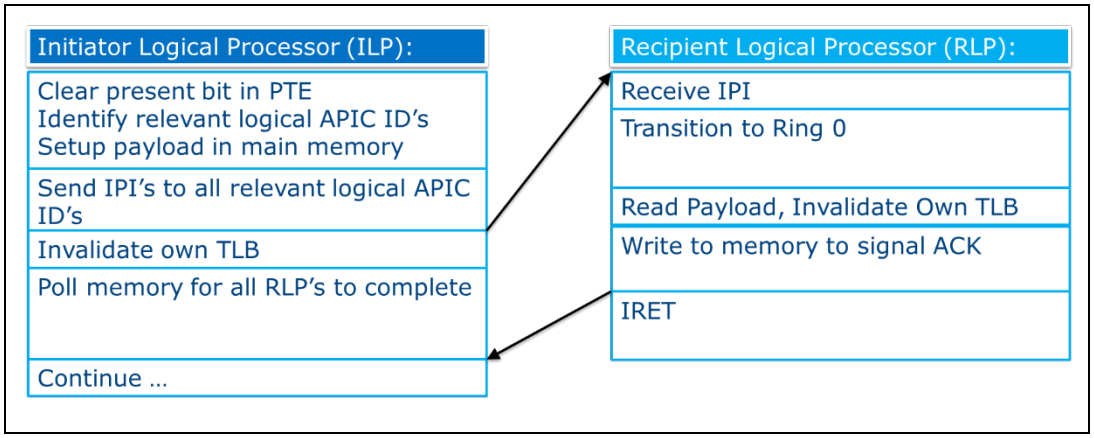
Note: This document is relevant for future processors based on the Sapphire Rapids microarchitecture only. These processors have a CPUID Signature DisplayFamily_DisplayModel value of 06_8FH. If additional processors support RAR in the future, the processor names will be added to future revisions of this document.

RAR is used for speeding up remote TLB shootdowns and allowing them to be serviced while a long instruction is executing on the remote processor or when interrupts are disabled on that processor. RAR is architected to allow for future expansion.

1.1 Overview of Software-based TLB Shootdown

Figure 1-1 shows an example of a TLB shootdown process in an existing software-based protocol. In this case, there is a need to remove a page mapping from the page table, so software running on the initiating logical processor (ILP) starts by clearing the present bit in the page-table-entry (PTE). It then sets up a payload in memory using some predefined memory format and sends an IPI to the LP's that require the TLB invalidation. The recipient logical processor (RLP) transitions into ring 0 after receiving the interrupt and the interrupt handler performs the actions specified by the ILP. After completion of the action, the interrupt handler acknowledges the completion by writing an ACK value to a predefined memory location. The ILP software can then read this location and detect that the protocol has completed. If by the time this process completed no page fault for this page has occurred, the ILP software can safely remove the mapping of the page from the page table.

Figure 1-1. Software-based TLB Shootdown



1.2 Overview of RAR-based TLB Shootdown

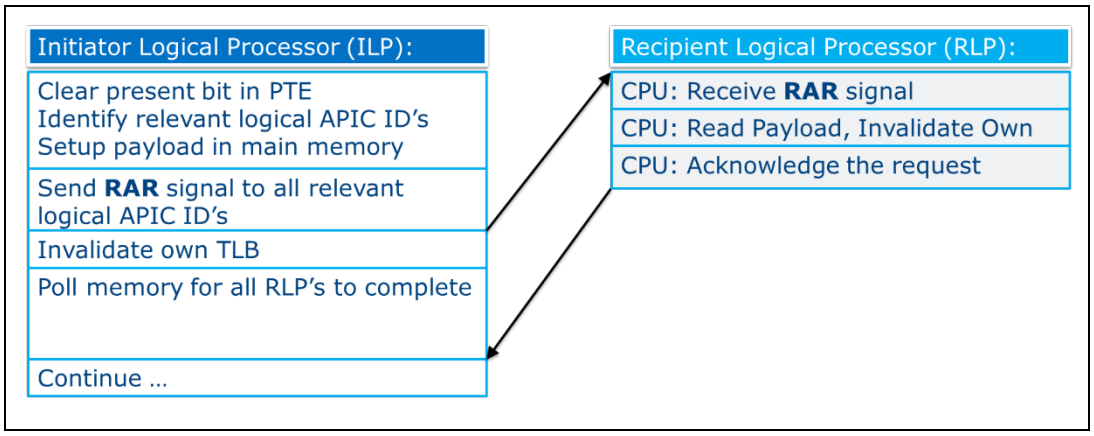
Figure 1-2 shows an example of the same TLB shootdown process shown above but using a RAR-based protocol where a hardware operation replaces software on the RLP.

1. Software on the ILP starts similarly by clearing the present bit in the page-table-entry (PTE), or making some other modification to the page tables.
2. It then sets up a payload in memory location, as defined in section 5.1, using a format defined in section 6.
3. The ILP software then indicates which RLPs are participating in the protocol by setting up the action vector as defined in section 5.2.
4. The ILP software sends a RAR signal by writing to the ICR as defined in section 3. The ILP software must ensure all writes to the payload and action vector are globally observed before sending the RAR.

    a. At this point, the ILP may invalidate its own TLB by signaling RAR to itself in order to invoke the RAR handler locally as well.

5. The RLP that receives the RAR performs the actions specified by the ILP in the payload entirely by hardware. After completion of the action, the RLP hardware acknowledges the completion by writing a **RAR_SUCCESS** value to the action vector. The ILP must not modify the value.
6. The ILP software polls this action vector to detect that the RLP has completed the action. If there are multiple RLPs, the ILP must poll multiple action vectors.

Figure 1-2. RAR-based TLB Shootdown



RAR supports several variations of the Remote TLB Shootdown process and additional remote actions that follow the basic RAR nature where software running on the ILP initiates, sets up, and checks completion of the Remote Action Request while the RLP responds to the request in hardware.

1.3 Operating System Setup for Remote Action Request

In order to set up RAR, the operating system must perform the following actions:

1. First, the operating system must detect support for RAR by reading the MSRs specified in section 2.
2. The operating system then allocates physical memory for a shared payload table, and physical memory for an action vector per logical processor as specified in section 5.
3. The operating system must then program all LPs with the addresses of these memory structures and enable RAR on the LPs by using the MSRs specified in section 4.
4. Once all LPs are programmed, the operating system can begin using RAR for TLB shootdown or other actions supported by RAR.

2 Enumeration

2.1 Detection

RAR support, as described in this document, is enumerated via the RAR bit (bit 1) in the IA32_CORE_CAPABILITIES MSR (MSR address 0CFH) on parts with a CPUID Signature DisplayFamily_DisplayModel value of 06_8FH. RAR is a model specific feature and thus the RAR behavior may change on future parts that enumerate IA32_CORE_CAPABILITIES[RAR].

When IA32_CORE_CAPABILITIES.RAR == 0, all RAR MSRs are inaccessible, and any attempt to access these MSRs will result in a #GP.

2.2 RAR_INFO MSR

RAR capabilities are reported through the read-only RAR_INFO MSR. This MSR is a model specific and available only if IA32_CORE_CAPABILITIES.RAR == 1. See section 4 for detailed information on this MSR.

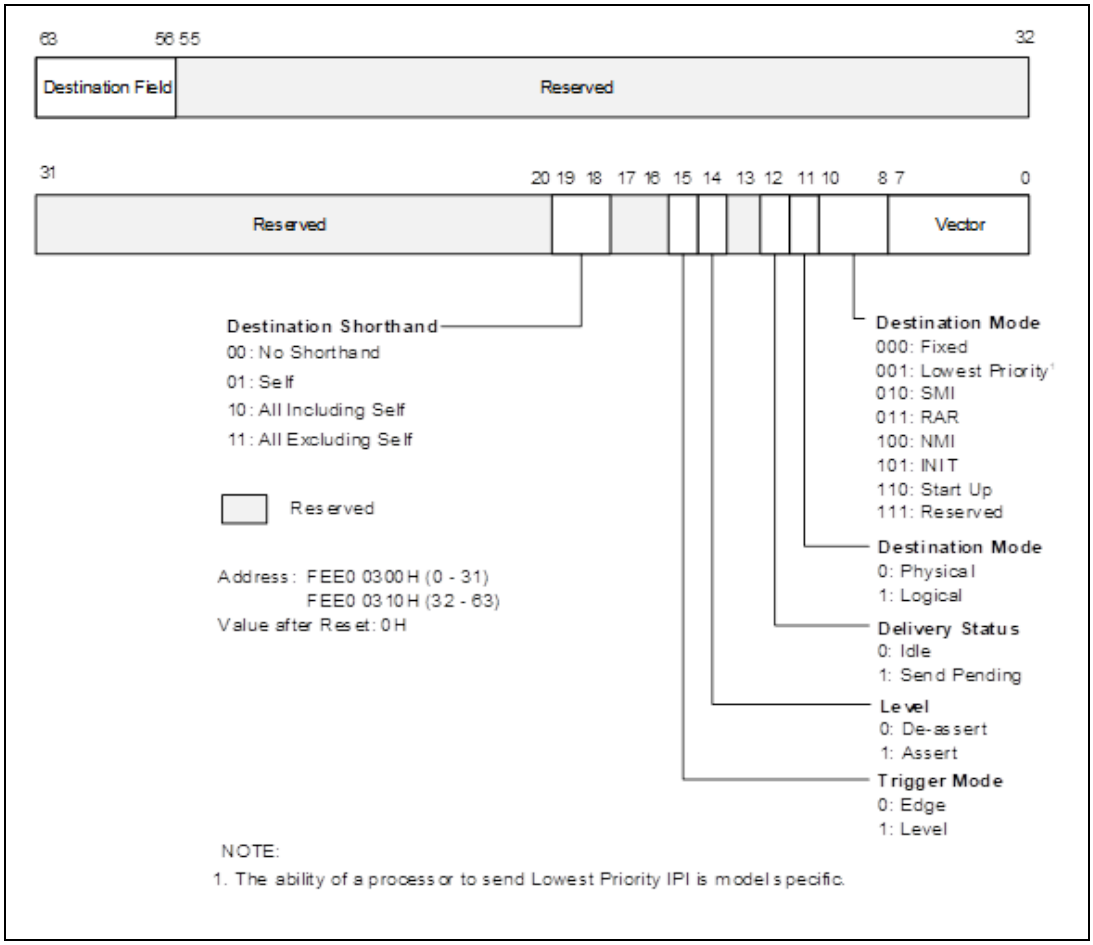
2.3 Enabling

RAR is enabled by setting the Enable bit in the RAR_CONTROL MSR; see section 4 for details. Enabling is possible only when IA32_CORE_CAPABILITIES.RAR == 1.

3 Signaling a Remote Action Request

Signaling a RAR is done similar to sending an INTR, by writing to the Interrupt Command Register (ICR). Figure 3-1 shows the Interrupt Command Register with RAR availability added.

Figure 3-1. Interrupt Command Register (ICR) with RAR Availability



RAR uses the previously reserved delivery mode (0b011), and can be set up as follows:

- **Destination:** Can be any.
- **Vector:** Must be zero[1] (other values are reserved for future use). Writing a different value will be signaled to the Error Status Register (ESR) with bit "Send Illegal Vector" and will be rejected (ICR Reject).
- **Trigger mode:** Ignored (always Edge).
- **Level:** Ignored (always 1).

[1] On some implementations, a vector value of 1 will not be rejected, but will be ignored by the receiving LP.

The ability to use RAR delivery mode is only available on processors when IA32_CORE_CAPABILITIES.RAR == 1.

4 Remote Action Request MSRs

All MSRs below are non-accessible when IA32_CORE_CAPABILITIES.RAR == 0. Reserved fields must be zero; an attempt to write a non-zero value results in a #GP.

Table 4-1. Remote Action Request MSRs

Register Address	MSR Name and Bit	Scope	MSR and Bit Description
EDH	RAR_CONTROL	Thread	RAR Control (R/W)
	63:32		Reserved
	31		ENABLE: RAR events are recognized. When RAR is not enabled, RARs are dropped.
	30		IGNORE_IF: Allow RAR servicing at the RLP regardless of the value of RFLAGS.IF.
	29:0		Reserved
EEH	RAR_ACTION_VECTOR	Thread	Pointer to RAR Action Vector (R/W)
	63:MAXPHYADDR		Reserved
	MAXPHYADDR-1:6		VECTOR_PHYSICAL_ADDRESS: Pointer to the physical address of the 64B aligned RAR action vector.
	5:0		Reserved
EFH	RAR_PAYLOAD_TABLE_BASE	Thread	Pointer to Base of RAR Payload Table (R/W)
	63:MAXPHYADDR		Reserved
	MAXPHYADDR-1:12		TABLE_PHYSICAL_ADDRESS: Pointer to the base physical address of the 4K aligned RAR payload table.
	11:0		Reserved
F0H	RAR_INFO	Thread	Read Only RAR Information (RO)
	63:38		Always zero.
	37:32		TableMaxIndex: Maximum supported payload table index.
	31:0		Supported payload type bitmap. A value of 1 in bit position [i] indicates that payload type [i] is supported. For additional details, see section 5.1.

5 Remote Action Request Memory Structures

5.1 Payload Table

The payload table contains RAR payloads. This table is allocated by the operating system in contiguous physical memory and pointed to by each LP's RAR_PAYLOAD_TABLE_BASE MSR. This architecture doesn't preclude the option for the operating system to allocate multiple Payload Tables, one per a subset of LPs and set the RAR_PAYLOAD_TABLE_BASE Logical Processor MSRs accordingly.

The payload table contains 'N' entries of 64 bytes, where N = RAR_INFO.TableMaxIndex + 1. For future processors based on the Sapphire Rapids microarchitecture, the table is fixed size at 4KB, i.e., N = 64 entries.

The operating system can pre-allocate an entry per ILP. Alternatively, if there are more LPs than N, the operating system can let ILPs dynamically look for empty slots in the table.

In a PCID enabled system where software threads are allocated each with a different PCID, the operating system may choose to allocate a separate Payload Table for each RLP, i.e., each RLP is its own 'RAR Island'.

The payload table must be 4KB-aligned and contiguous in physical memory.

On the RLP side, if action [j] in the Action Vector is pending (0 <= j < N), the RLP core accesses the entry at physical address (RAR_Payload_Table_Base_MSR.TablePhysicalAddress << 12) + j * 64.

If the RLP reads an illegal remote action request type, it signals a failure as described in section 5.2.

Each entry defines the payload for a single action; see details below.

Table 5-1. Payload Table Details

Software Available [511:256]	Payload [255:32]							Reserved [31:16] - MBZ	Type [15:8]	Software Available[1] [7:0]
	Linear Address [255:192]	CR3 [191:128]		Reserved [127:43] MBZ	Num Pages [42:37]	Stride [36:35]	Subtype [34:32]	0	0 = Page invalidation	
	Linear Address [255:192]	Ignored [191:128]		Reserved [127:43] MBZ	Num Pages [42:37]	Stride [36:35]	Subtype [34:32]	0	1 = Page invalidation ignore CR3	
	Linear Address [255:192]	Ignored[2] [191:140]	PCID [139:128]	Reserved [127:43] MBZ	Num Pages [42:37]	Stride [36:35]	Subtype [34:32]	0	2 = PCID invalidation	
	Reserved [255:192] MBZ	EPTP [191:128]		Reserved[127:35] MBZ			Subtype [34:32]	0	3 = INVEPT	
	Linear Address [255:192]	Reserved [191:144] MBZ	VPID [143:128	Reserved [127:43] MBZ	Num Pages [42:37]	Stride [36:35]	Subtype [34:32]	0	4 = VPID invalidation	
	Reserved [255:192] MBZ	MSR Data [191:128]		Reserved [127:43] MBZ	MSR index [42:32]			0	5 = MSR Write	
		Future Payload						0	Future Payload	

[15:8] = **Type**. The action request type:

- Type 0 = Page invalidation; invalidate one or more pages.
- Type 1 = Page invalidation; invalidate one or more pages, ignore CR3 match.
- Type 2 = PCID invalidation; invalidate pages associated with a specific PCID.
- Type 3 = EPT invalidate; invalidate pages associated with a specific EPTP.
- Type 4 = VPID invalidation; invalidate pages associated with a specific VPID.
- Type 5 = MSR write; write value to specified MSR.
- Other types reserved for future usages.

[255:32] = **Payload**. The type-specific payload; detailed in the section 6.

[511:256] = Ignored by the CPU.

[1] Can be used by software to indicate "Valid".
[2] These bits can be the upper CR3 bits. CR3 bits other than PCID [11:0] are not used in INVPCID action request type.

5.2 Action Vector

The action vector is a per-RLP 64 Bytes aligned vector of actions, used for ILP-RLP communication of the RAR protocol. It is pointed to by the RAR Action Vector MSR.

Contains N entries of 8 bits, where N = RAR_INFO MSR.TableMaxIndex+1.

A_{N-1} [8N-1:8N-8]	…	A_2 [23:16]	A_1 [15:8]	A_0 [7:0]

Each 8-bit entry j (0 <= j < N) defines the per-RLP status of the j^{th} action request; see details below.

- 0x00 = **RAR_SUCCESS**. Set by ILP to indicate that action **j** can be ignored by RLP. Set by RLP to indicate successful handling of the RAR.
- 0x01 = **RAR_PENDING**. Set by ILP to indicate that action **j** is pending for this RLP. While an action **j** is pending, software must not modify payload **j** as it may be in use by the RLP.
- 0x80 = **RAR_FAILURE**. Set by RLP to indicate unsuccessful handling (e.g., bad action request type in payload table).
- Other values are reserved for future use. Such values are ignored by the RLP.

6 Remote Action Payloads

6.1 RAR Invalidation Payloads

- The RAR event is a serializing event, ensuring all subsequent memory operations will not use the invalidated cached translations.
- Any invalidation resets the state of the monitor hardware as set up by a MONITOR instruction.
- The RAR event ignores the CPL of the RLP, meaning that invalidations can happen when CPL=3.

6.1.1 RAR Address-specific Invalidation Payloads

This section details the common fields of RAR address-specific invalidation payloads (payloads 0, 1, 2 and 4), subtype 0:

- **Stride:** If NumPages > 1, the linear address (bits [255:192]) is incremented by 4K (value 0), 2M (value 1), or 1G (value 2).
    - Remaining values are reserved. Using reserved values will result in failure **regardless of subtype**.
- **Num Pages:** The number of pages to invalidate, minus one. That is, a value of zero indicates just one page.
- **Linear Address:** The address of the page to invalidate (or the first page, if NumPages > 0).
    - The entire invalidation range is checked for canonicity upfront. Using a non-canonical address will result in failure **regardless of subtype** and regardless of addressing mode.
    - Addresses are always calculated in 64-bit mode. If the address range wraps around the 64-bit boundary, it results in a failure **regardless of subtype**.
    - Addresses are always taken as a linear address, that is, from a base of zero.

6.2 Payload Type 0 - Page Invalidation

When the payload type is “Page invalidation”, the RAR handling in the RLP, depending on the payload SubType field, imitates the operation of performing INVLPG instructions corresponding with the “Num Pages” field in the payload, or the TLB invalidation corresponding with “MOV CR3 / CR0”.

In this case, the payload table entry is interpreted as shown below.

Payload [255:32]					
Linear Address [255:192]	CR3 [191:128]	Reserved [127:43] MBZ	Num Pages [42:37]	Stride [36:35]	SubType [34:32]

SubType:

0) Address specific invalidation.

   Invalidate mappings for multiple pages starting at the linear address for current RLP context except global translations.

1) Reserved; using this subtype causes a failure.

2) All-context invalidation including global.

   Invalidate all mappings for current RLP context[1] including global translations.

3) All-context invalidation excluding global.

   Invalidate all mappings for current RLP context excluding global translations.

SubType values 4–7 are reserved. Using these SubType values causes a failure.

CR3: The RLP linear-address space context to be invalidated. If the RLP's CR3[62:12] does not match the CR3 value in the payload, the invalidation request is acknowledged immediately with **RAR_SUCCESS** without any further action.

Caution: Any operating system that uses PCID must use the "PCID invalidation" type and not "Page invalidation". Use of "Page invalidation" works in the current PCID context of the RLP at the time it begins handling the RAR and does not necessarily invalidate the right entry from the TLB.

6.3 Payload Type 1 - Page Invalidation without CR3 Match

When the payload type is "Page invalidation ignore CR3", handling is identical to "Page invalidation Payload Type", except that the CPU does not check for a CR3 match.

6.4 Payload Type 2 - PCID Invalidation

When the payload type is "PCID invalidation", the RAR handling in the RLP imitates the operation of performing INVPCID instructions corresponding with the "Num Pages" field in the payload.

In this case, the table entry is interpreted as shown below.

[1] If VPID is enabled, the context invalidation will occur for VPID 0 only. A VMM that wants to invalidate a specific VPID must use INVVPID.

Payload [255:32]						
Linear Address [255:192]	CR3 upper bits [191:140] Ignored[3]	PCID[3] [139:128]	Reserved [127:43] MBZ	Num Pages [42:37]	Stride [36:35]	SubType [34:32]

SubType:

0) Address specific invalidation.

   Invalidate mappings for multiple pages starting at the linear address associated with PCID[1], except global translations. In some cases, the invalidation may include global-translations or mappings for other linear addresses (or other PCIDs) as well.

1) Single PCID invalidation.

   Invalidate all mappings associated with PCID, except global translations.

2) All-context invalidation including global.

   Invalidate all mappings for all PCIDs, including global translations.

3) All-context invalidation excluding global.

   Invalidate all mappings for all PCIDs, except global translations.

SubType values 4–7 are reserved. Using these SubType values causes a failure.

PCID:

If CR4.PCIDE = 0, a logical processor does not cache information for any PCID other than 000H. In this case, executions with INVPCID subtypes 0 and 1 are allowed only if the PCID value specified in the PCID filed (bits [139:128]) of the INVPCID payload is 000H. Executions with INVPCID subtypes 2 and 3 invalidate mappings only for PCID 000H.

6.5 Payload Type 3 - EPT Invalidation

When the payload type is "EPT invalidation", the RAR handling in the RLP imitates the operation of performing the INVEPT instruction.

This opcode generates a failure if not in VMX operation.

In this case, the table entry is interpreted as shown below.

[1] If VPID is enabled, the context invalidation will occur for VPID 0 only. A VMM that wants to invalidate a specific VPID must use INVVPID.

Payload [255:32]			
Reserved [255:192] MBZ	EPTP [191:128]	Reserved [127:43] MBZ	SubType [34:32]

SubType:

1) Single-context invalidation. The logical processor invalidates all mappings associated with bits 51:12 of the EPT pointer (EPTP) specified in the payload. It may invalidate other mappings as well. IF VM entry with the "enable EPT" VM execution control set to 1 would fail due to the EPTP value, the opcode generates a failure.

2) Global invalidation. The logical processor invalidates mappings associated with all EPTPs.

Other SubType values are reserved. Using these SubType values causes a failure.

EPTP:

EPTP to invalidate. Only bits 51:12 of this field are used.

6.6 Payload Type 4 - VPID Invalidation

When the payload type is "VPID invalidation", the RAR handling in the RLP imitates the operation of performing INVVPID instructions corresponding with the "Num Pages" field in the payload.

This opcode generates a failure if not in VMX operation.

This payload type invalidates all the specified mappings for the indicated VPID(s) regardless of the EPTP and PCID values with which those mappings may be associated.

In this case, the table entry is interpreted as shown below.

Payload [255:32]						
Linear Address [255:192]	Reserved [191:144] MBZ	VPID [143:128]	Reserved [127:43] MBZ	Num Pages [42:37]	Stride [36:35]	SubType [34:32]

SubType:

0) Address specific invalidation.

   Invalidate mappings for multiple pages starting at L_ADDR associated with VPID, except global translations. In some cases, the invalidation may include global-translations or mappings for other linear=address (or other VPIDs) as well.

1) Single context invalidation.

The logical processor invalidates all mappings tagged with the VPID specified in the INVVPID descriptor. In some cases, it may invalidate mappings for other VPIDs as well.

2) All-contexts invalidation.

   The logical processor invalidates all mappings tagged with all VPIDs except VPID 0000H. In some cases, it may invalidate translations with VPID 0000H as well.

3) Single-context invalidation, retaining global translation.

   If the INVVPID type is 3, the logical processor invalidates all mappings tagged with the VPID specified in the INVVPID descriptor except global translations. In some cases, it may invalidate global translations (and mappings with other VPIDs) as well.

Other SubType values are reserved. Using these SubType values causes a failure.

VPID:

VPID to invalidate. Unlike the INVVPID instruction, a VPID value of zero is legal and does not result in a failure.

Linear Address: The address of the page to invalidate (or the first page, if NumPages > 1).

6.7 Payload Type 5 - MSR Write

When detecting this payload type, the RLP attempts to write the value specified in MSR DATA to the HWP_REQUEST MSR, as if performed by a WRMSR instruction. Any condition that would lead to a #GP during WRMSR causes a RAR Failure condition instead, as described in section 5.2. RAR_SUCCESS indicates that the request value has been written into the HWP_REQUEST MSR.

In this case, the table entry is interpreted as shown below.

Payload [255:32]			
Reserved [255:192] MBZ	MSR Data [191:128]	Reserved [127:43] MBZ	MSR index [42:32]

MSR Index:

0 – HWP Request.

All other values are reserved.

MSR DATA:

64 bits of data to write into MSR.

7 RLP Remote Action Request Handling

The RLP begins handling the RAR upon arrival of a RAR signal if RAR_CONTROL.Enable=1.

If the enable bit, RAR_CONTROL.Enable==0, then the RAR is not serviced and the CPU's internal pending RAR indication is cleared.

A RAR event is invoked on the RLP on an instruction boundary if all of the following conditions hold:

1. A RAR event has been detected by the local APIC.
2. RAR is enabled in the RAR_CONTROL MSR.
3. There is no blocking by MOV SS or STI, or by any other condition specified in section 8.4.
4. At least one of the following holds:
    a. IGNORE IF is set in the RAR_CONTROL MSR.
    b. EFLAGS.IF is set.
    c. RLP is in VMX non-root mode.

7.1 Remote Action Request Handling of Multiple Requests

If a RAR event is received and multiple RARs are pending in the action vector, all of them are handled within the same event window sequentially starting from event 0.

7.2 Remote Action Request RLP Handling Flow

The flow that the RLP hardware performs upon receiving a RAR event is as follows:

- Clear the pending RAR indication.
- Ensure all previous stores are globally observed.
- tmp_vector = load 64 bytes from the address MSR RAR_ACTION_VECTOR.”Vector Physical Address” field.
- For each action j in the action vector:

```
If tmp_vector[j] == RAR_PENDING
 Read slot j from payload table pointed by RAR Payload Table MSR
 If payload not supported:
     Write vector[j] = RAR_FAILURE
     // Perform single byte store to vector at offset j
 Else:
    Handle action type according to payload
    Write vector[j] = RAR_SUCCESS
    // Perform single byte store to vector at offset j
Else
 Ignore
End Flow
```

8 Remote Action Request Interaction with Other Features

8.1 RAR Priority Relative to INTR

The operating system can configure the priority of RAR relative to INTR, as specified below.

8.1.1 Behavior if RAR_CONTROL.IGNORE_IF is Clear

RAR is lower priority than any INTR. RAR cannot be serviced when RFLAGS.IF=0.

RAR is only handled when EFLAGS.IF is set.

8.1.2 Behavior if RAR_CONTROL.IGNORE_IF is Set

If IGNORE_IF is set, RAR is handled independent of the RFLAGS.IF value. If RAR and INTR or NMI arrive at the same time, RAR is taken on the first instruction after the CPU completes IDT vectoring, before the first instruction inside the software interrupt handler.

8.1.3 Behavior in VMX Non-Root Mode

In VMX non-root mode, the setting of the MSR RAR_CONTROL.IGNORE_IF or RFLAGS.IF has no effect on the handling of RAR.

8.2 Sleep States

RAR is a wake event from any MWAIT sleep state. The operating system should avoid sending a RAR to a CPU in C6 or higher sleep state as TLBs are invalidated in such sleep states. RAR sent to an RLP in a sleep state with RAR_CONTROL.IGNORE_IF clear and RFLAGS.IF clear may not wake the RLP.

RAR is accepted in halt state. After handling a RAR received while in halt state, the CPU returns to halt state.

8.3 RAR and Intel® SGX

RAR is accepted while inside an enclave.

RAR is guaranteed to generate an asynchronous enclave exit (AEX) if the requested address is inside the current enclave ELRANGE; see section 32.3 of the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3D. Additionally, see chapter 35

of the Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3D for more information on AEX.

RAR may generate an AEX even if the requested address is not inside the current enclave.

8.4 RAR Clear Conditions

RAR pending event is cleared on INIT and GETSEC SENTER/ENTERACCS.

RAR control MSR is reset on GETSEC SENTER/ENTERACCS and SENTER of the RLP, and on some CPU's also by launch of a BIOS Guard module.

8.5 RAR Inhibit States

RAR is inhibited while the processor is in any of the states listed below. A RAR received during these states remains pending. For example, a RAR taken in SMM would remain pending until after the SMI.

- Blocking by MOVSS or STI.
- System management mode (SMM, and SMM entered by dual-monitor).
- Wait-For-SIPI state.
- Shutdown state.
- SENTER sleep state.
- BIOS guard.

8.6 RAR and Intel® PT; Tracing with GPA

RAR may be used to invalidate EPT mappings in a guest while the guest is actively tracing with RTIT to guest physical addresses.

In certain scenarios, a VMExit due to EPT violation/misconfiguration may occur as a result of a RAR delivered to an RLP that is running in non-root mode:

- PT table EPT translations were invalidated or made non-writeable prior to the RAR.
- RAR payload type is INVVPID or INVEPT.
- The modification to the EPT was detected by the RLP during RAR handling.
- The VM guest is actively tracing with PT using PT2GPA mode.

VMM is expected not to modify the EPT addresses of pages currently in use by the guest without first marking the entry as not present and performing TLB invalidation. Performing such an invalidation may result in a write to the page mapped by the stale TLB translation even after the RLP has signaled RAR_SUCCESS.