AMD

Tiered Memory Page Migration Operations Guide

Publication # **58151**	Revision: **0.51**
Issue Date: **May 2023**	

Advanced Micro Devices

© 2023 Advanced Micro Devices, Inc. All rights reserved.

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale.

Trademarks

AMD, the AMD Arrow logo, AMD EPYC, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Contents

Revision History

Date	Revision	Description
March 2023	0.50	Initial public release.
May 2023	0.51	Must Be Zero fields changed to Reserved. Command status field description updated.

Reference Documents

PID	Document Title
55766	Secure Encrypted Virtualization API (SEV)
56860	SEV Secure Nested Paging Firmware ABI Specification (SNP)
57299	AMD Platform Security Processor BIOS Implementation Guide for EPYC Processors
55901	Processor Programming Reference (PPR) for AMD Family 19h Model 11h, Revision B0 Processors

Abbreviations Used

Abbreviation	Definition
TMPM	Tiered Memory Page Migration
FW	Firmware
SW	Software
HV	Hypervisor
OS	Operating System
RMP	Reverse Map Table
RB	Ring Buffer
SPA	System Physical Address

Chapter 1 Hypervisor/Operating System (HV/OS) Information

1.1 Purpose

This document describes optional extensions to System Encryption, the Secure Encrypted Virtualization API (SEV) and SEV Secure Nested Paging Firmware ABI Specification (SNP) to facilitate page migration operations. The process of copying page content and updating the virtual to physical mapping is called Page Migration. AMD created the Tiered Memory Page Migration (TMPM) engine to aid in the process.

To determine if Tiered Memory Page Migration (TMPM) support is available, the HV/OS (through its TMPM driver) can interrogate ENGINE_READY bit in the PM_Status register found in section 2.4.

TMPM requires IOMMU enablement in the system.

Tiered memory systems consist of two or more classes of memory with different performance characteristics. Software can optimize application performance by moving frequently accessed data into a faster memory tier while keeping less frequently accessed data in slower memory. Moving memory content between tiers to optimize system performance can be handled transparently by leveraging the system's virtual memory support as illustrated below:

- Software monitors memory access behaviors to identify frequently accessed pages in slow memory, and infrequently accessed pages in fast memory
- Software decides when the content of virtual memory pages should be migrated between physical memory tiers to optimize performance
- Software causes the memory content of the physical page(s) to be copied into the new memory tier and IOMMU virtual->physical page table mappings to be updated to point to the new physical location of the data

The TMPM can be used to off-load page migration and it also solves a two key challenges:

- TMPM can pause DMA traffic to a page while the page content is copied and the IOMMU page table entry is updated. This allows memory pages to migrated safely even if they are actively being used by an IO device.
- TMPM is a trusted agent that can migrate pages owned by a Guest that is leveraging AMD's Secure Encrypted Virtualization or Secure Nested Paging features.

1.2 Page Migration Use Cases

As stated above, the page migration commands described in this document allow software to migrate pages from one set of system physical addresses to another while maintaining the same guest physical address (GPA). These commands enable two unique capabilities that are not directly available to software running on the x86 CPU or to regular I/O devices in the system such as SDXI:

1. Migration of pages that may be concurrent targets of DMA operations
2. Migration of guest pages that are protected by Secure Nested Paging (SNP) that Hypervisor or Operating System (HV/OS) cannot otherwise access

The page migration commands allow for the movement of a group of pages. Processing a group of similar page movements within a single command is far more efficient than using the general command batching system to initiate a number of commands that each move a single page.

1.3 Page Migration with Concurrent DMA

The PM_PAGE_MOVE_IO command allows for the migration of pages from one system physical location to another while the page is the concurrent target of DMA operations. This requires coordination with the IOMMUs. In order to utilize this commands, software must provide the location of the relevant IOMMU host Page Table Entry (hPTE) within the command.

1.4 SNP Hypervisor Migration Flow

The PM_PAGE_MOVE_IO command allows the hypervisor to migrate Hypervisor state pages in an SNP-enabled system. The source and destination pages must be in the Hypervisor state.

After migration, the source and destination pages are returned in the Hypervisor state.

In a non-SNP-enabled system, default pages can also be copied. They are returned in the default state.

1.5 SNP Guest Migration Flows

The PM_PAGE_MOVE_GUEST command allows the hypervisor to migrate pages that belong to a Guest in an SNP system. The source pages must be in the Guest-Valid or Guest-Invalid state while the destination pages must be in the Pre-Migration state.

This command can also replace the PAGE_MOVE command in the SNP ABI. However, it is up to software to ensure that if a request is made on a 2M page, the page is made up of physically contiguous 4k pages.

After migration, the source pages are returned in the Pre-Migration state. The destination pages take on the original state of the corresponding source pages.

1.6 Pre-Migration Reverse Map Table State

A new Pre-Migration state is defined to facilitate page migration of SNP Guest pages. This state is similar to Guest-Invalid state except the RMP contains a Platform Specific ASID value (PS_ASID_VAL - available from the TMPM driver). and the GPA value is ignored. The PS_ASID_VAL is never associated with a Guest. A guest may not PVALIDATE a Pre-Migration page to turn it into a Guest-Valid page.

Hypervisor or Operating System (HV/OS) shall use the PS_ASID_VAL as input to the RMPUpdate command.

Software shall use RMPUpdate to convert a Pre-Migration state page back into a Hypervisor state page.

1.7 Multi-Page Commands

TMPM commands support the movement of multiple pages. There is a PM_PAGE_MOVE_GUEST command that can move multiple pages similar to multiple calls to the SNP_PAGE_MOVE command, if desired.

Up to 128 pages may be migrated using a multi-page command. The pages are described in a Page Migration List composed of up to 128 Page Migration Entries whose formats are described with each command. In general, there is no requirement for the pages described in the Page Migration list to be adjacent to each other. Multiple guests may be referenced within a single list. It is required that there be no order-dependence between entries in a Page Migration list. It is also required that there be no order-dependence between commands containing multiple Page Migration lists.

Page Migration Entries are updated with Page Migration Status to provide per-page status for the operation. If the overall command returns with PM_PARTIAL_SUCCESS status, software must check the Page Migration Status Entries to determine whether individual page movements were successful or not. If the overall command returned with PM_SUCCESS, then all of the pages were migrated. Any other return code indicates that none of the page migrations were executed.

The Page Migration Status Entry for a page may indicate the corresponding page could not be migrated but no error occurred. This may occur, for example, if the RMP entry associated with the page is unavailable due to a concurrent RMPUPDATE command. Software may re-attempt the page migration once the conflict is resolved. The status code(s) indicating a retry may be re-attempted is: PM_RMP_NOTEXCLUSIVE.

The Page Migration List must start at a 4KB system physical address regardless of the number of entries in the list.

1.8 Performance

Testing by AMD has shown that there are certain guidelines that, if adhered to, can yield better performance. Regardless of the TMPM performance, system activity can decrease TMPM performance due to intersecting use of the data fabric, DMA, and IOMMU activity.

AMD has discovered that commands with 16 or more pages have significantly higher bandwidth than smaller numbers of pages. Maximum performance is achieved when done with commands having 128 pages moved at a time.

The command input queue (a ring buffer) works best if commands to TMPM arrive quickly enough to keep the ring buffer ahead of the engine’s capability to empty the buffer. Additionally, it is advised to not request interrupts on every command, but rather to request an interrupt at the end of a sequence of commands, similar to a pacing flag. That can be done by using the NOP command or by requesting an Interrupt on Complete from any single command.

TMPM can interrupt on a ring buffer low-water mark being reached to let software refill the queue prior to it becoming empty. This can help keep the queue more full and the engines continually operating.

1.9 Non-Migratable Data Structures

The following data structures do not support migration:

- All IOMMU memory data structures that are accessed using SPA such as the IOMMU host page table data structures, IOMMU command buffer, IOMMU event log etc.
- IOMMU page tables used to map the vIOMMU Private Address Space
- vIOMMU pinned backing storage memory buffers
- The Ring Buffer and any argument pages

Hypervisor or Operating System (HV/OS) is responsible for not issuing page migration commands targeting these structures.

1.10 IOMMU Mapping Requirements

For the PM_PAGE_MOVE_IO command, source pages must be DMA-mapped using IOMMU host page tables. Only one IOMMU host page table may be used to DMA-map any system physical page being migrated with this command.

To support the PM_PAGE_MOVE_IO command, an IOMMU host page table shall be constructed to map the memory containing the IOMMU host page tables used to map migratable memory. Within this document, the virtual address space associated with this page table shall be referred to as the host page table virtual address (HPTVA).

The TMPM engine may modify IOMMU host page table entries in a non-atomic manner. Host software must avoid concurrent modifications to referenced IOMMU host page table entries during TMPM operations. Additionally, the TMPM engine may set the host Access and Dirty bits in IOMMU host page table entries referenced during TMPM operations. This can be done by pausing ring buffer processing or careful consideration of command creation so that software hPTE operations do not overlap TMPM hPTE operations.

1.11 Reloading the TMPM firmware

The Page Migration feature has firmware that can be reloaded. This cannot take place while any operations are in flight and the HV/OS is expected to pause operations, tear down the ring buffer, and then issue the reload command using the BIOS to PSP mailbox protocol described in the AMD Platform Security Processor BIOS Implementation Guide for EPYC Processors, publication #57299. (Summarized below) As a matter of best practice, the PAGEMIGRATION_EN bits in the IOMMU's should be disabled during the reload.

If the ring buffer is still initialized, the firmware will return an error (PM_MX_INVALID_RELOAD_REQUEST). The PSP will return an error reflecting that to the caller.

The firmware provided for reload must be the same or a greater version. If not, the ASP will not reload the firmware and an error will be returned indicating a version issue.

If there is any other reload issue that prevents the engine from being reloaded, the ASP will return an error indicating the engine failed to reload. Any driver or software utilizing the features of the TMPM firmware is reminded to always check the ENGINE_READY status bit before attempting to communicate with the firmware.

1.11.1 Distinguishing between EPYC family reload capabilities

The methodology for TMPM reload varies slightly depending on which EPYC family reload is being done. To distinguish between families, the TMPM driver should execute the PM_GET_CAPABILITIES command (see section 5.1). The output from that command includes a field called FW_VER_Major. Version convention has that field representing the EPYC family, although not in a 1:1 match.

EPYC Family 19h is represented by a FW_VER_Major value of 71 decimal.

Reload, therefore, works as follows:

- The TMPM driver will attempt to execute the PM_GET_CAPABILITIES command.
- If that fails, TMPM update is not available.
- If it succeeds, the driver will look at the FW_VER_Major field of the returned capabilities data
    - That number represents the SOC on which this FW is running

- The driver will use that data to drive which execution path to use for TMPM reload

Instructions for TMPM reload for differing EPYC Family values follow.

1.11.2 EPYC Family 19h TMPM Reload details

The software client will place the updated firmware image in system memory and build a reload firmware header, also in system memory as defined below.

Table 1. Reload Firmware Header

Byte Offset	Bits	In/Out	Name	Description
00h	31:0	In	TotalSize	Total Size of MBOX_BUFFER (including this field) in bytes (Current implementation is 20 (0x14) bytes.
04h	31:0	Out	Status	Command Execution Status
08h	31:0	In	Image_addr_lo	Lower 32 bits of the Physical address of the firmware image
0ch	31:0	In	Image_addr_hi	Upper 32 bits of the Physical address of the firmware image
10h	31:0	In	Image_size	Size of the loaded image
14h	31:0	In	Reserved	Unused

Simplified Algorithm for sending this command (see register table below). Access to the ASP registers is via a PCI driver

1. Client polls on Ready flag of Command/Status Register to be zero (i.e. loop until set to 1)
2. Write the physical address of the Reload Firmware Header into the CmdRspBufAddr_Lo and CmdRspBufAddr_Hi registers.
3. Write a 0x70 into the Command/Status Register[Command ID] field and write 0 to all other fields and update the Command/Status Register.
4. Loop on Ready flag of Command/Status Register while 0 (command in progress)
5. At this point, the Status field of the Reload Firmware Header is valid and, combined with the Status in the Command/Status Register, reflects the Reload operations' status.

Table 2. PSP Register Definitions

Register Name	Bits	Name	Description
Command/Status Register	31	Ready	Set by the target to indicate the mailbox interface state.

			0 — Not ready to handle commands (or handling previous command) 1 — ready to handle next command
	30	Recovery	Set by the target to indicate that the host has to execute FW recovery sequence
	29	ResetRequired	Set by the target to indicate that the host must execute warm reset if
	28:27	-	Reserved
	23:16	CommandID	0x5F for Reload TMPM FW
	15:0	Status	Set by the PSP to indicate the execution status of the last command
CmdRspBufAddr_Lo	31:0	Addr_Lo	Lower 32 bits of physical address of Command/Response Buffer
CmdRspBufAddr_Hi	31:0	Addr_Hi	Upper 32 bits of physical address of Command/Response Buffer

1.12 Debugging the Page Migration Environment

There can be several challenges to development of the TMPM feature. The Ring Buffer offers the HV/OS a way to send large numbers of commands to the TMPM function in the SOC. Due to TMPM being able to process commands asynchronously, there may be times when the software developers need/want to send single commands (comprised of 1-X migration entries) to the TMPM function in order to ensure that the software environment is working as intended. To do this, the software should not advance the Tail pointer more than one command entry away from the Head pointer. That way, when the queue empties (from the TMPM engine perspective) processing will pause (an interrupt for queue empty can be optionally created).

1.13 Restrictions

The TMPM hardware can only move data from one 4k aligned page to another 4k aligned page therefore all system physical addresses (SPAs) must be 4k aligned. The software should ensure that it is not expecting data to be moved in any other way.

This does not apply to SPAs used to represent host page table entries as they can be 8 byte aligned.

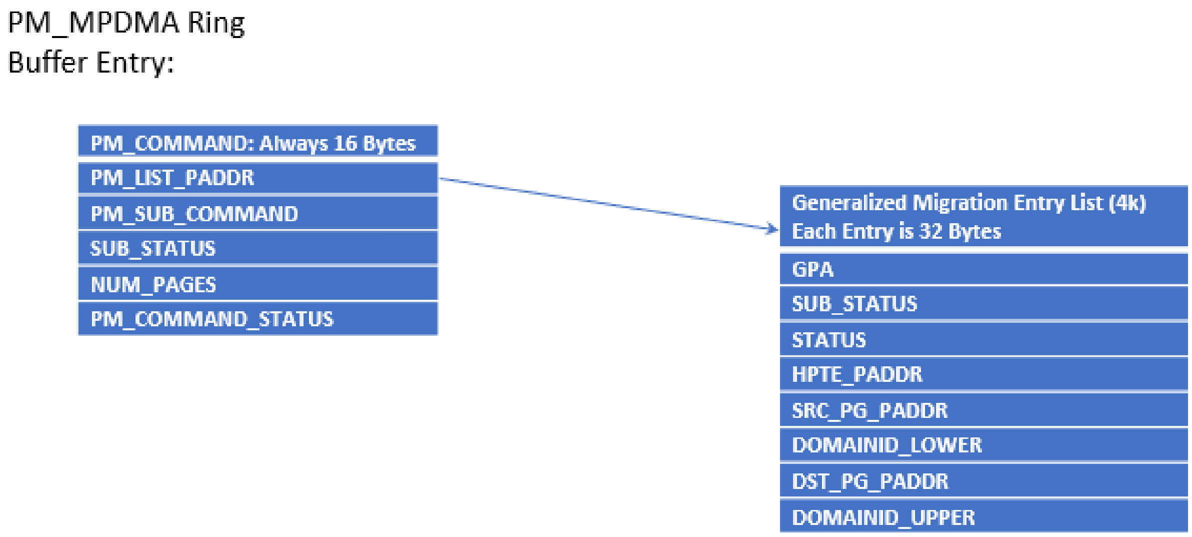
Chapter 2 Command Overview

2.1 Page Migration Commands

TMPM Commands are used to direct the migration of memory pages. TMPM commands are sent to a page migration ring buffer (separate from the SEV/SNP Ring buffer) via an OS visible interface. Status is returned in the memory of each TMPM command.

The Ring Buffer (RB) must use pages of type DEFAULT or HV_FIXED (see 2.5.2). Argument pages can be DEFAULT or HV_FIXED or HV pages. The software is not expected to move an HV page that is used as an argument page and the FW will obtain exclusive access to that page while using it to prevent such errant assignments.

Each TMPM Command contains a parameter page in its PM_LIST_PADDR. A parameter page is a 4k block of memory consisting of 1-128 Migration Entries (ME's). The drawing below provides a representative view of Page Migration commands and parameter pages.



2.2 A Note About Physical Addresses

System Physical Addresses (SPA's) can be 46 bit or 52 bit, depending on whether system encryption is enabled. Each entry is sized for 52 bits to work in any mode as appropriate. The bottom 12 bits of an address must always be zero (4k address alignment), except for Host Page Table Entry physical addresses, which are 8 byte aligned so have the bottom 3 bits zeroed out. If a

2M page is passed in, the bottom 21 bits must always be zero. Any SPA not meeting these conditions will result in an error.

2.3 Page Migration Ring Buffer

The Page Migration Ring Buffer is set up by client software via a driver. Similar to the way the Ring Buffer works for SEV/SNP, the software should set up one or more pages for TMPM commands. A single page can hold 256 commands of 16 bytes each.

If InterruptOnCompletion and/or InterruptOnError is requested in a given command and one or both has occurred, the TMPM engine will indicate that in the command. If interrupts for either are enabled, the Page Migration dispatcher will provide an interrupt to the driver. Once that is done, it will continue processing commands.

2.4 TMPM Engine Mailbox Layout

The TMPM engine has registers that are used to communicate between the software and the TMPM function. The C2PMSG register number is given in the first column in parentheses. Registers are addressed in the TMPM driver as base + (register number * 4).

See table below (brief descriptions of the fields) and section 2.5(further explanation of the fields and their usage).

C2PMSG_x	Bits	Dir	Description
PM_RBCtl (0)	31:6	-	Reserved
	5	In	CLEAR_INT_ON_THRESH – when written as a 1, clear the interrupt on queue threshold bit
	4	In	CLEAR_INT_ON_EMPTY – when written as a 1, clears the interrupt on queue empty bit
	3	In	CLEAR_IN_ON_COMPLETE – when written as a 1, clears the interrupt on complete bit
	2	In	CLEAR_INT_ON_ERR – when written as a 1, clears the interrupt on error bit
	1	In	DRIVER_INITIALIZED – signals the firmware that the driver has finished its initialization and the firmware should evaluate the configuration
	0	In	PAUSE – when written as a 1, the queue stops running. In-flight commands complete

PM_ReadPtr (1)	31:16	Out	PS_ASID_VAL - Platform Specific ASID VALue used for RMPUpdate cmds
	15:0	Out	QReadPtr – the firmware updated location into the RB of the last updated command
PM_WritePtr (2)	31:16	-	Reserved
	15:0	In	QWritePtr - this is moved by software when one or more new command(s) go into the buffer
PM_RBCData (3)	31:10	-	Reserved
	9	In	IntOnThresh – interrupt when the queue reaches QThreshold
	8	In	IntOnEmpty – unconditionally interrupt when queue empties
	7:0	In	NUM_PAGES - Page Migration Ring Buffer size in 4k pages
PM_RBSPALOW (4)	31:0	In	Low 32 bits of Ring Buffer starting address
PM_RBSPAHI (5)	31:0	In	High 32 bits of Ring Buffer starting address
PM_RBCfg (6)	31:16	-	Reserved. Must be zero.
	15:0	In	QThreshold – Qsize in entries below which an interrupt may be optionally generated
PM_Status (7)	31	Out	TOGGLE - Toggles between 0 and 1 when commands are completed – see section 2.5.1 for details
	30	Out	QThreshIntStat – the number of remaining queue entries is below the configured threshold QFreeIntStat – the queue is empty.
	29	Out	QFreeIntStat – the queue is empty.
	28	Out	IntOnComplt – a command with InterruptOnComplete caused an interrupt
	27	Out	IntOnError – a command with InterruptOnError caused an interrupt
	26	Out	RBWritePtr_Err (value greater than RB Size) – see section 2.5.6
	25	Out	RBMem_Err – the command could not be read from the RB

	24	Out	RB_Terminated – the TMPM firmware terminated the RB forcibly (see section 2.5.3)
	23	Out	GET_CAPABILITIES_SUPPORTED
	22-7	-	Reserved
	6	Out	RBMem_Type_Valid – see 2.5.1 step 11
	5	Out	QCmdPtr_Valid
	4	Out	PM_RBCfg_Valid
	3	Out	PM_RBCData_Valid
	2	Out	PAUSED – command processing is paused if read as a 1
	1	Out	DRIVER_INIT_COMPLETE – see 2.5.1 step 11.
	0	Out	ENGINE_READY - MPDMA Engine Ready Flag (0 - not ready, 1 ready)

2.5 Page Migration Driver

The (HID = AMDI0095) can be used by the OS to bind a driver to the TMPM device defined in the ACPI namespace (ASL code). The resources associated with the TMPM device are declared by resource objects within the device definition in ACPI.

Additionally, a page migration driver needs to enable a bit called PAGEMIGRATION_EN in all available IOMMU's. This is in the IOMMU register named IOMMU_MMIO_CNTRL_1 as described in the AMD Processor Programming Reference. This initialization needs to be done only once, although if the driver is re-initialized, re-writing the bits is harmless.

The TMPM function consists of a single interface provided to the operating system via a set of mailbox registers (described below). That interface is capable of passing commands to multiple execution units, that is, commands can be executed in parallel. That capability is supported by using a ring buffer (RB) interface that can accept many commands to be processed as quickly as the hardware/firmware can do so. It also leads to the need for a set of rules around driver operation.

Like all ring buffers, the Read pointer is under control of the firmware and is set to the start of the RB upon initialization. The Write pointer is controlled by the software client (that is, a driver). Since several commands can be processed simultaneously, execution order between commands is not guaranteed. The only ways the driver knows a command is complete is the Read Pointer has been moved past it by the firmware. (It is strongly recommended that the PM_COMMAND_STATUS field is zero when sent to the RB as the firmware uses only non-zero return codes).

2.5.1 General Interface Notes

NOTE: The PM_Status register contains a bit called TOGGLE. Before sending a command to the RB_Ctl register, the software sending the command should note the state of the TOGGLE bit. When it changes, the command is complete and the contents of PM_Status are valid. TOGGLE is only used for reflecting acceptance and completion of changes to the RB_Ctl register.

Changing the configuration/address of the ring buffer during operation is not allowed

2.5.2 PM Device Initialization

To initialize the device, the driver (command by its client) needs to do the following:

1. Check the PM_Status[ENGINE_READY] == 1. This indicates that the TMPM Engine has initialized its internal firmware and is ready for external interaction (including driver initialization).
2. If PM_Status[ENGINE_READY] == 0, then go to step 1
3. If PM_Status[DRIVER_INIT_COMPLETE] == 1 then shut down the driver (see below).
4. Set the client provided system physical address (SPA) for the ring buffer (mailboxes PM_RBSPALOW and PM_RBSPAHI). This must be 4k-page aligned and consist of 1-255 physically contiguous pages. If the SPA is not 4k-aligned or not a valid SPA, QCmdPtr_Valid will not be set when the command completes.
5. Set the configuration data for queue behavior
    a. PM_RBCData must have NUM_PAGES set to a value between 1 and 255. If not, RBCData_Valid will not be set when the DRIVER_INIT_COMPLETE bit is set.
    b. PM_RBCfg may have a value (in entries) below which the driver wishes to receive an interrupt indicating the ring buffer has reached a “low water mark”. A value of 0 indicates no interrupt will be generated, regardless of the value of IntOnThresh. If the value is greater than the number of commands the ring buffer can hold, RBCfg_Valid will not be set when the DRIVER_INIT_COMPLETE bit is set.
6. Set PM_WritePtr to zero
7. Hook the IOAPIC interrupt (if used)
8. Set PM_RBCtl[DRIVER_INITIALIZED] to 1. This causes a signal to the PM Engine and the configuration to be validated. Setting this bit to 1 when DRIVER_INIT_COMPLETE is 1 is ignored.
9. Optionally, PM_RBCtrl[PAUSE] can be set to 1 at the same time to initialize the ring buffer in paused mode.
10. The driver should spin on PM_Status[PM_DRIVER_INIT_COMPLETE] to be set to 1. The bits PM_RBCfg_Valid, PM_RBCData_Valid and PM_RBSPA_Valid and RBMem_Type_Valid will indicate what was not valid, if any. If any are not 1, the driver is not correctly initialized and should be reinitialized with proper data. It is up to the driver/client to understand that TMPM firmware can use DEFAULT pages if SNP_INIT has never been called and only HV_FIXED pages if SNP_INIT has ever been called (that is, the RMP has been set up).

11. The driver must be able to report the PS_ASID_VAL to the software that called it – typically the HV/OS - upon request. PS_ASID_VAL is available in PM_ReadPtr[31:16] after PM_Status[PM_DRIVER_INIT_COMPLETE] is 1.

2.5.3 PM Device Shutdown

1. Driver sets PM_RBCtl[PAUSE] = 1.
2. Driver waits until PM_WritePtr and PM_ReadPtr are equal **and/or** PM_Status[PAUSED] is 1.
3. Driver sets PM_RBCtrl[DRIVER_INITIALIZED] to 0.
4. Driver waits for PM_Status[PM_DRIVER_INIT_COMPLETE] to go to 0.
5. The driver performs any needed cleanup, including unhooking the interrupt, processing the status of commands not yet evaluated in the queue and figuring out what to do with unprocessed commands in the queue.

The driver is now ready to be removed, reconfigured, or left idle, depending on client needs. If the driver sets DRIVER_INITIALIZED to 0 without either PAUSED==1 or HEAD == TAIL, any inflight commands will complete and processing commands will cease. PAUSED will be set to 1 and page migration will be paused. This is an indeterminate state for the driver and commands should not be resumed without shutting down the driver correctly.

NOTE: the TMPM firmware, under certain limited circumstances can forcibly terminate the ring buffer configuration and mark the ring buffer uninitialized. The firmware will clear the RB configuration and set RB_Terminated and fire an interrupt to the driver so it can know that the RB was terminated by the firmware.

2.5.4 PM Driver Pause/Resume

Ring Buffer processing can be paused/resumed by the software client performing a write operation to set PM_RBCtl[PAUSE] to 1. Because the execution units may not be done with their in-flight command(s), the firmware will not take further commands from the ring buffer and allow in-flight commands to complete. The client can determine that the pause has occurred when PM_Status[PAUSED] is equal to 1. No further ring buffer processing will take place until the software Client writes a zero to the PM_RBCtl[Pause] bit at which point ring buffer processing resumes.

The firmware also has a PauseOnError bit that can be set per command that results in the queue being paused as described above.

2.5.5 Ring Buffer Operation

Client software manipulates the queue (after configuration) by adding entries and writing the RBTail register which triggers an interrupt for the dispatcher to start servicing commands. If command processing is under way, the dispatcher notes the new tail and continues processing. Commands to the TMPM function are described in section 3.1 (Command Format).

The client software above the driver is responsible for looking through the completed commands to determine which command(s) caused any reported interrupt.

2.5.6 Interrupts

There are six TMPM interrupt sources that share a single x86 interrupt signal. Four sources can be enabled individually, however RBWritePtr_Err and RBMem_Err are not controllable.

- RBWritePtr_Err will be set when the software writes a value to the RBTail register that is outside the defined size of the ring buffer.
    - In this error case firmware will
        - not accept further commands from the queue
        - allow in-flight commands to complete
        - pause the queue (i.e. set the PAUSE bit)
        - set RBWritePtr_Err to 1
        - signal x86 by triggering the shared interrupt
    - The software may choose to terminate the ring buffer, or restart the queue after correcting the underlying issue (adjusting the QWritePtr value and clearing the RBWritePtr_Err and PAUSE bit).
- RBMem_Err is set when an error is encountered during a read of the command from the ring buffer.
    - In this error case firmware will:
        - Ensure MCA actions will be taken to report the error
        - not accept any further commands from the queue
        - allow the in-flight commands to finish (if possible)
        - pause the queue (i.e. set the PAUSE bit)
        - signal x86 by triggering the shared interrupt
    - The software should terminate the ring buffer and reinitialize it.
- QfreeIntStat is set to indicate that the Ring Buffer is empty. It is set and an interrupt triggered, if enabled, when PM_WritePtr == PM_ReadPtr. It is cleared when PM_WritePtr is next incremented, indicating a command has been placed on the queue. This is enabled at the queue level by setting IntOnEmpty and cannot be changed during operation unless the ring buffer is re-configured. This is an immediate interrupt and reflects the exact RB status.
- QthreshIntStat is set to indicate that the Ring Buffer has only PM_RBCfg[Qtheshold] items remaining. It is only triggered once, but the status bit is left set until the Ring Buffer has more entries than PM_RBCfg[Qthreshold]. This is enabled at the queue level and cannot be changed during operation unless the ring buffer is re-configured. This is an immediate interrupt and reflects the exact RB status when it is issued.
- IntOnError, when set, indicates that one or more commands had requested that an interrupt be generated if a command returned an error (anything but PM_SUCCESS).
    - This is an informative interrupt and processing continues unless this is the last command in the Ring Buffer.
    - Software can request this for each command or only for certain commands as it prefers.

  - It is up to the software to pause processing with the PM_RBCtl[PAUSE] bit if it wants to stop processing to manage errors. The processing has stopped when PM_Status[PAUSED] is true.
  - The interrupt is issued immediately.
  - This interrupt is only issued once (in the event multiple commands request it) until cleared.
- IntOnComplt, when set, indicates that one or more commands had requested that an interrupt be generated when that command completes.
  - This is an informative interrupt and processing continues unless this is the last command in the Ring Buffer.
  - It is up to the software to pause processing with the PM_RBCTL[PAUSE] bit if it wants to stop processing to manage completions. The processing has stopped when PM_STATUS[PAUSED] is true.
  - The interrupt is issued immediately.
  - This interrupt is enabled within a command (allowing software to form a loose barrier).
  - This interrupt is only issued once (in the event multiple commands request it) until cleared.

If software desires precise halting of the ring buffer, software must ensure that the command to halt on is the last command in the queue. If precise error halting is desired, then only one command should be placed in the queue at a time. This is due to the distribution of commands to multiple execution units, for parallelization and higher performance.

Software can clear the interrupt status bits by writing the PM_RBCtl[CLEAR_INTS] bit to 1. This will only be accepted when there are no more entries in the ring buffer to process or PM_Status[PAUSED] is true. If processing resumes interrupt generation will also resume. Note that PM_RBTail can be incremented while processing is paused.

2.6 SPA (System Physical Address) Encoding

In systems enabling SMEE or SME-MK, encryption information is encoded into the SPAs passed from software to the page migration firmware. The following table describes how software must encode SPA addresses used in page migration commands and page migration parameter blocks.

SMEE	SME-MK	SPA Encoding
Off	Off	SPA[51:0] = System Physical Address
Off	On	SPA[51:46] = Host Encryption Key ID. 0 = Unencrypted SPA[45:0] = System Physical Address
On	Off	SPA[51] = C-bit 0 = Unencrypted

		1 = Encrypted SPA[46:0] = System Physical Address*

*Note: The actual system physical address space size may be less than 52 bits. Refer to CPUID 8000_001F[EBX] for more details.

When SMEE is enabled, software may indicate the use of encrypted memory by setting the C-bit to 1, otherwise the memory is treated as unencrypted. TMPM commands and parameters specify whether an address field is intended to utilize Hypervisor memory or Guest memory. Firmware will select the appropriate encryption key id if encryption is selected.

SPAs must be well-formed and in the proper location (i.e. in appropriate address ranges).

Chapter 3 The TMPM Commands

3.1 Command Format

Any reserved or fields marked Out in the In/Out column should be set to zero upon submission by the software.

Byte Offset	Bits	In/Out	Name	Description
00h	63:52	-	Reserved	Reserved
	51:12	In	PM_LIST_PADDR	Bits 51:12 of the SPA of the page migration list. The list is composed of NUM_PAGES+1 Page Migration List Entries.
	11:0	-	Reserved	Reserved
08h	31	In	INT_ON_COMPLT	Interrupt on Completion of this command
	30	In	INT_ON_ERR	Interrupt if this command has an error
	29	In	PAUSE_ON_ERROR	Automatically pauses the queue if an error occurs. Allows any in-flight commands to complete.
	28	-	Reserved	Reserved
	27:16	In	NUM_PAGES	Number of list entries - 1
	15:8	-	Reserved	Reserved
	7:0	In	PM_SUB_COMMAND	One of the Page Migration commands
0ch	31	Out	DoneInt	Set to 1 if this command caused an interrupt when it completed
	30	Out	ErrInt	Set to 1 if this command caused an interrupt due to an error
	29:12	-	Reserved	Reserved
	11:8	Out	SUB_STATUS	Per the status table
	7:0	Out	PM_COMMAND_STATUS	Overall Command result - per status table. – if PM_PARTIAL_SUCCESS is

				returned, each ME in the list pointed to by PM_LIST_PADDR must be examined to determine individual page error status. If PM_SUCCESS is returned, individual ME status is not filled out.

3.2 PM_SUB_COMMAND IDs

Command Name	**Value**	**Description**
PM_GET_CAPABILITIES	00H	This command returns a set of capabilities of the firmware. Please see Section 5.1
PM_NOOP	01H	Takes no action. Always returns success
PM_PAGE_MOVE_IO	02H	This command moves the contents of one or more pages in a non-SNP system to memory that is either unencrypted or encrypted using a hypervisor owned key. If SNP is or has run, this command moves the contents of one or more Hypervisor-owned pages within the system physical address space in a system.
PM_PAGE_MOVE_GUEST	03H	This command moves the contents of one or more pages belonging to an SNP guest within the system physical address space.

Chapter 4 TMPM Statuses

Command status for page migration operations will be made available to the client in the command entry in the ring buffer and in the migration or update lists. There are two entries – a status field and a sub-status field (intended to help isolate where an error code may have been encountered).

4.1 Command and list entry status

PM_MPDMA Status Codes	Value	Description
PM_SUCCESS	F0h	PM value for success
PM_INVALID_PLATFORM_STATE	01h	Platform not in INIT state
PM_INVALID_PAGE_PADDR	02h	Invalid pointer to the ME
PM_INVALID_NUM_PAGES	03h	Invalid number of pages (0-255 for ME lists, up to 4095 for UE lists)
PM_MEM_POISONED	04h	POISON was returned by the hardware
PM_INVALID_PAGE_STATE	05h	Page is in the wrong state
PM_INVALID_PAGE_SIZE	06h	A page was not the correct size (RMP check)
PM_RMP_NOTEXCLUSIVE	07h	Indicates that migration of a page did not succeed due to an inability to gain exclusive access to an RMP entry. Software may re-attempt migration.
PM_INVALID_GUEST	08h	The guest is invalid
PM_INVALID_GPA_PADDR	09h	Invalid Guest Physical Address
PM_INVALID_HPTE_PADDR	0Ah	Invalid HPTE Physical Address
PM_INVALID_COMMAND	0Bh	Command in command buffer is invalid
PM_INVALID_SRC_PG_PADDR	0Ch	Invalid Source Page Physical Address
PM_INVALID_DST_PG_PADDR	0Dh	Invalid Destination Physical Page Address
PM_INVALID_GCTX_PG_PADDR	0Eh	Invalid Guest Context Page Physical Address
PM_INVALID_RMP_PADDR	0Fh	Invalid RMP Page Address
	10h	Reserved for AMD Internal use only
PM_MEM_ABORTED	11h	Memory Operation was aborted

Reserved	12h	-
PM_VERSION_ERROR	13h	The command does not match the current version of the TMPM firmware.
PM_INVALID_PM_LIST_ADDR	14h	Invalid Pointer to PM_PARAMETER_BLOCK
PM_ADDRESSES_MISMATCH	15h	Two addresses that should match do not
PM_PARTIAL_SUCCESS	16h	Command completed partially successfully, please check individual ME statuses
-	17h	Reserved
PM_INVALID_OVERLAP	18h	Invalid attempt to create 2 overlapping pages (ex. 2MB page on top of existing 4KB page)
PM_HW_MEM_ERR	19h	A HW error reading or writing memory was encountered that didn’t lead to an error listed in the algorithms. The SUB-STATUS field may contain more information.
PM_MX_INVALID_RELOAD_REQUEST	84h	A request to reload the TMPM FW was made while PM commands are executing. The calling SW should quiesce the command queue.

4.2 Sub Statuses – valid only if STATUS != PM_SUCCESS

SUB_STATUS	**Value**	**Used By**
SUB_STATUS_UNUSED	0	Not applicable as a SUB_STATUS
PM_VALIDATE	1	The error reported was detected during address validation
PM_ACCESS	2	The error reported was detected during address reference

Chapter 5 Page Migration Parameter Blocks

In the following command descriptions, and reference to RMP_ENFORCE means that SNP has been initialized and an RMP table has been created. Even if SNP has been shutdown, the RMP table remains and RMP_ENFORCE will remain true.

5.1 PM_GET_CAPABILITIES

This command is intended to be used to query the capabilities and version(s) of the firmware. The use case surrounding it suggests that it be marked with interrupt on completion to advise the HV/OS so any functional/performance decisions can be made.

All input fields except PM_LIST_ADDR, PM_SUB_COMMAND, InterruptOnError and InterruptOnComplete field are ignored on input.

On output the page pointed to by PM_LIST_ADDR has this structure placed in it:

Byte Offset	Bits	Name
0	32:16	CAP_Version
	15:0	CAP_Length
4	31:24	FW_VER_Major
	23:16	FW_VER_Minor
	15:0	Reserved
8	31:24	max_spec_major
	23:16	max_spec_minor
	15:8	min_spec_major
	7:0	min_spec_minor
12	31:5	Reserved (0)
	4	1 = RELOAD supported
	3	1 = PM_NOOP command supported
	2	1 = PM_PAGE_MOVE_GUEST command supported
	1	1 = PM_PAGE_MOVE_IO command supported
	0	1 = PM_GET_CAPABILITIES command supported

16 - 4095	0xFFF	Reserved

5.2 PM_PAGE_MOVE_IO

There are 1-128 Migration Entries described in the table below to be processed per PM_PAGE_MOVE_IO command.

This command moves the contents of one or more pages and may be used in both SNP-enabled and non-SNP-enabled systems. In SNP-enabled systems, this command may only be used on Hypervisor-owned pages. In SNP-enabled systems, the memory may be either unencrypted or encrypted using a hypervisor key. When used in SEV-enabled systems, only unencrypted and host/hypervisor encrypted memory may be targeted.

This command is capable of moving pages that may be actively in use by DMA devices. This command will directly update the IOMMU host PTEs to point to the new system physical page locations.

For non-SNP-enabled systems, if ATS devices have access to the pages and they are not configured for secure ATS, software must modify each of the IOMMU host PTEs associated with the migrating pages by setting hPTE.PMS=1. Software must then issue ATS invalidations to the IOMMU for the affected pages followed by completing a completion wait synchronization command. These steps must occur prior to calling PM_PAGE_MOVE_IO. The firmware will set hPTE.PMS=0 for each successfully migrated page prior to completing PM_PAGE_MOVE_IO.

The CPU must be prevented from accessing the pages to be moved. Software may make the pages non-present in the CPU host page tables followed by performing a CPU TLB shootdown, prior to calling PM_PAGE_MOVE_IO. After the completion of PM_PAGE_MOVE_IO, software is responsible for updating the physical addresses in the CPU host page tables to point to the new locations of the successfully migrated pages.

If SNP is running or has been initialized, the source and destination pages contained within the page migration list must all be in the Hypervisor state. After migration, both the source and destination pages remain in the Hypervisor state.

In order to migrate a 2M page, software shall smash it into 4K pages prior to performing migration. This command does not support direct migration of 2M pages.

Byte Offset	Bits	In/Out	Name	Description
00h	63:52	-	Reserved	Reserved
	51:12	in	SRC_PG_PADDR	bits 51:12 of the Source Page System Physical Address

	11:4	-	Reserved	Reserved
	3:0	in	DOMAINID_UPPER	Bits 15:12 of the Domain ID for the IOMMU hPTE
08h	63:52	-	Reserved	Reserved
	51:12	in	DST_PG_PADDR	bits 51:12 of the Destination Page System Physical Address
	11:0	in	DOMAINID_LOWER	Bits 11:0 of the Domain ID for the IOMMU hPTE
10h	63:52	-	Reserved	Reserved
	51:3	in	HPTE_PADDR	bits 51:3 of the Source Host Page Table Entry System Physical Address
	2:0	-	Reserved	Reserved
18h	63:60	Out	PTE-ERR	PTE Error code for this page move
	59:56	Out	PTE-SUBERR	PTE Sub-Error code for this page move
	55:52	-	Reserved	Reserved
	51:12	in	GPA	bits 51:12 of the Guest Physical Address associated with SRC_PG_PADDR
	11:8	Out	SUB_STATUS	Per the status table
	7:0	Out	STATUS	ME result - per status table if command status is not PM_SUCCESS, otherwise not used

Firmware executes these actions for each page migration entry list entry. Status is returned through the Page Migration List Entry STATUS and SUB_STATUS field.

1. The firmware checks that SRC_PADDR and DST_PADDR are valid SPAs. If not, the firmware returns PM_INVALID_SRC_PG_PADDR or PM_INVALID_DST_PG_PADDR.
2. The firmware checks that the Page Address in the Host Page Table Entry (hPTE) points to the SPA of the source page. If not, the firmware returns PM_ADDRESSES_DONT_MATCH.
3. If RMP_ENFORCE == FALSE
    a. The firmware checks that the hPTE is in the correct state for this operation. If not, PM_INVALID_PAGE_STATE is returned.
4. If RMP_ENFORCE == TRUE

    a. The firmware checks that the SRC_PADDR and DST_PADDR refer to either Hypervisor or Default pages. If not, firmware returns PM_INVALID_PAGE_STATE.

5. Firmware marks the hPTE for migration.
6. Firmware issues an IOMMU TLB invalidate using the provided GPA and DomainID
7. Firmware then issues an IOMMU TLB Sync and waits for completion.
8. If RMP_ENFORCE == TRUE
    a. If SRC_PADDR is not a Default page, firmware tries to obtain exclusive access to the source RMP. If this is not successful firmware returns PM_RMP_NOTEXCLUSIVE.
    b. If SRC_PADDR is a Hypervisor page, firmware marks the page for migration
    c. If DST_PADDR is not a Default page, firmware tries to obtain exclusive access to the destination RMP. If this is not successful, firmware returns PM_RMP_NOTEXCLUSIVE.
    d. If SRC_PADDR is within a Hypervisor page, firmware checks that the source RMP indicates a 4KB page size. If not, firmware releases exclusive access to the source and destination RMPs, and returns PM_INVALID_PAGE_SIZE.
    e. If DST_PADDR is a Hypervisor page, firmware checks that the destination RMP indicates a 4KB page size. If not, firmware releases exclusive access to the source and destination RMPs, and returns PM_INVALID_PAGE_SIZE.
    f. If DST_PADDR is a Hypervisor page, firmware marks the page for migration.
    g. Firmware issues a CPU TLB invalidate to the GPA .
    h. Firmware issues an IOMMU TLB invalidate to the GPA .
    i. Firmware issues a CPU+IOMMU TLB Sync and waits for completion.
    j. If SRC_PADDR is a Hypervisor page, firmware marks the page for migration.
9. Firmware sets hPTE.PMS to indicate active migration
10. Firmware then programs all IOMMUs to flush DMA and waits for completion.
11. The firmware copies the source page into the destination page maintaining proper encryption as needed
12. Firmware programs the hPTE to point to the destination SPA
13. If RMP_ENFORCE==TRUE
    a. If DST_PADDR is a Hypervisor page, firmware releases exclusive access to the destination RMP .
    b. If SRC_PADDR is a Hypervisor page, firmware releases exclusive access to the destination RMP .
14. Firmware clears the hPTE from migration use.
15. Firmware returns PM_SUCCESS.

5.3 PM_PAGE_MOVE_GUEST

There are 1-128 Migration Entries described in the table below to be processed per PM_PAGE_MOVE_GUEST command.

This command moves the contents of one or more pages belonging to an SNP guest within the system physical address space. This command may be used as a faster replacement for the SNP

PAGE_MOVE command. However, if software sends in a 2M page, that page must consist of physically contiguous 4k pages, as it does now when the SEV/SNP PAGE_MOVE command or software is required to smash it into 4K pages prior to performing migration. The command does not support direct migration of non-physically contiguous 2M pages. Moreover, 2M pages must be 2M aligned.

DMA to the migrating pages is not supported. If there is concurrent DMA to the page during the migration process, the DMA may be aborted.

The CPU must be prevented from accessing the pages to be moved, generally by making them non-present in the CPU host page table.

The source pages must be in the Guest-Valid or Guest-Invalid state. The destination pages must be in the Pre-Migration state. After migration, the destination pages take on the state of their corresponding source page. The source pages are returned in the Pre-Migration state.

Byte Offset	Bits	In/Out	Name	Description
00h	63:52	-	Reserved	Reserved
	51:12	in	SRC_PG_PADDR	bits 51:12 of the Source Page System Physical Address
	11:0	-	Reserved	Reserved
08h	63:52	-	Reserved	Reserved
	51:12	in	DST_PG_PADDR	bits 51:12 of the Destination Page System Physical Address
	11:0	-	Reserved	Reserved
10h	63:52	-	Reserved	Reserved
	51:12	in	GCTX_PG_PADDR	bits 51:12 of the Guest Context Page System Physical Address
	11:1	-	Reserved	Reserved
	0	in	PAGE_SIZE	0 indicates a 4k page, 1 indicates a physically contiguous 2M page
18h	63:60	Out	PTE-ERR	PTE Error code for this page move
	59:56	Out	PTE-SUBERR	PTE Sub-Error code for this page move
	55:52	-	Reserved	Reserved
	11:8	Out	SUB_STATUS	Per the status table

	7:0	Out	STATUS	ME result - per status table if command status is not PM_SUCCESS, otherwise not used

1. Firmware executes these actions for each page migration entry list entry. Status is returned through the Page Migration List Entry STATUS field.
2. RMP_ENFORCE must be 1, otherwise return PM_INVALID_PLATFORM_STATE.
3. The firmware checks that both SRC_PADDR and DST_PADDR are not in Default page state. If not, the firmware returns PM_INVALID_PAGE_STATE.
4. Firmware checks that GCTX_PADDR is a valid SPA. If not, the firmware returns PM_INVALID_GCTX_PADDR.
5. The firmware then checks that the page at GCTX_PADDR is in the Context state. If not, the firmware returns PM_INVALID_GUEST.
6. Firmware tries to obtain exclusive access to the source RMP. If this is not successful firmware returns PM_RMP_NOTEXCLUSIVE.
7. Firmware tries to obtain exclusive access to the destination RMP. If this is not successful, firmware releases exclusive access to the source RMP and firmware returns PM_RMP_NOTEXCLUSIVE.
8. The firmware checks that the source and destination RMPs both indicate the same size page sizes. If not, the firmware releases exclusive access to the source and destination RMPs, and returns PM_INVALID_PAGE_SIZE.
9. The firmware checks that the source page is in the Guest-Valid, or Guest-Invalid state. If not, firmware releases exclusive access to the source and destination RMPs and returns PM_INVALID_PAGE_STATE.
10. The firmware checks the destination page is in the Pre-Migration state. If not, firmware releases exclusive access to the source and destination RMPs, and returns PM_INVALID_PAGE_STATE
11. Firmware issues a CPU TLB invalidate to the source page using source RMP.GPA and RMP.ASID.
12. Firmware issues a CPU TLB Sync and waits for completion.
13. Firmware copies the data from the source page to the destination page.
14. Firmware updates the destination RMP to reflect the migration.
15. Firmware updates the source RMP to reflect the migration.
16. Firmware returns PM_SUCCESS.

5.4 PM_NOOP

This command is intended to be used to mark the end of strings of page migration commands. Notably, it can be marked as interrupt on completion to advise the software that a long sequence of commands has been completed.

All input fields except the PM_SUB_COMMAND and the InterruptOnError and InterruptOnComplete field are ignored on input.

5.5 Command Output

On output, the Page Migration Command Format changes the information found in section 3.1 to include:

Byte Offset	Bits	In/Out	Name	Description
00h	63:0	-	Reserved	Reserved
08h	31:8	-	Reserved	Reserved
	7:0	in	PM_SUB_COMMAND	One of the Page Migration commands
0ch	31	Out	DoneInt	Set if this command caused an interrupt when it completed
	30	Out	ErrInt	Set if this command caused an interrupt due to an error
	29:12	-	reserved	Reserved
	11:8	Out	SUB_STATUS	Per the status table
	7:0	Out	PM_COMMAND_STATUS	Overall Command result - per status table. – if PM_PARTIAL_SUCCESS is returned, each ME in the list pointed to by PM_LIST_PADDR must be examined to determine individual page error status. If PM_SUCCESS is returned, individual ME status is not filled out.

Chapter 6 Error Handling And Mitigation

There are several kinds of errors that may occur in TMPM command processing. Errors in command format and content are easy to manage and result in the command being rejected. Errors in the migration list entry content are handled similarly and may result in partial success for a given command. This is true across the kinds of commands accepted by the TMPM engine. All of these errors are reported in the commands themselves and/or the migration list entries.

Errors in managing the Ring Buffer itself are reported in the PM_STATUS register. This would include things like an error in addressing the ring buffer memory.

Hardware related errors that occur during operation, where the hardware reports an error in performing an operation will result in those errors reported in the command status and/or via the Machine Check Architecture (MCA). If the TMPM function times out, as detected by a client, such as the page migration driver or software profiler the client should log an MCA, take whatever appropriate error detection/correction it can do and determine the best course of action, such as saving state, terminating VMs or simply not performing additional TMPM commands. A command can operate on up to 128 pages in one batch. Once the batch has begun processing, hardware errors can lead to the entirety of pages in that command being unrecoverable.

Errors result in command status codes being returned indicating failure. However, the state of the memory being operated upon may not be fixable. In that case, the system will eventually cease to function, but allows the software to attempt to save current data and system state prior to needing to reboot.