SEV Secure Nested Paging Firmware ABI Specification

Publication # **56860**	Revision: **0.8**
Issue Date: **August 2020**	

Advanced Micro Devices

Specification Agreement

This Specification Agreement (this “Agreement”) is a legal agreement between Advanced Micro Devices, Inc. (“AMD”) and “You” as the recipient of the attached AMD Specification (the “Specification”). If you are accessing the Specification as part of your performance of work for another party, you acknowledge that you have authority to bind such party to the terms and conditions of this Agreement. If you accessed the Specification by any means or otherwise use or provide Feedback (defined below) on the Specification, You agree to the terms and conditions set forth in this Agreement. If You do not agree to the terms and conditions set forth in this Agreement, you are not licensed to use the Specification; do not use, access or provide Feedback about the Specification.

In consideration of Your use or access of the Specification (in whole or in part), the receipt and sufficiency of which are acknowledged, You agree as follows:

1. You may review the Specification only (a) as a reference to assist You in planning and designing Your product, service or technology (“Product”) to interface with an AMD product in compliance with the requirements as set forth in the Specification and (b) to provide Feedback about the information disclosed in the Specification to AMD.

2. Except as expressly set forth in Paragraph 1, all rights in and to the Specification are retained by AMD. This Agreement does not give You any rights under any AMD patents, copyrights, trademarks or other intellectual property rights. You may not (i) duplicate any part of the Specification; (ii) remove this Agreement or any notices from the Specification, or (iii) give any part of the Specification, or assign or otherwise provide Your rights under this Agreement, to anyone else.

3. The Specification may contain preliminary information, errors, or inaccuracies, or may not include certain necessary information. Additionally, AMD reserves the right to discontinue or make changes to the Specification and its products at any time without notice. The Specification is provided entirely “AS IS.” AMD MAKES NO WARRANTY OF ANY KIND AND DISCLAIMS ALL EXPRESS, IMPLIED AND STATUTORY WARRANTIES, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, TITLE OR THOSE WARRANTIES ARISING AS A COURSE OF DEALING OR CUSTOM OF TRADE. AMD SHALL NOT BE LIABLE FOR DIRECT, INDIRECT, CONSEQUENTIAL, SPECIAL, INCIDENTAL, PUNITIVE OR EXEMPLARY DAMAGES OF ANY KIND (INCLUDING LOSS OF BUSINESS, LOSS OF INFORMATION OR DATA, LOST PROFITS, LOSS OF CAPITAL, LOSS OF GOODWILL) REGARDLESS OF THE FORM OF ACTION WHETHER IN CONTRACT, TORT (INCLUDING NEGLIGENCE) AND STRICT PRODUCT LIABILITY OR OTHERWISE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

4. Furthermore, AMD’s products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD’s product could create a situation where personal injury, death, or severe property or environmental damage may occur.

5. You have no obligation to give AMD any suggestions, comments or feedback (“Feedback”) relating to the Specification. However, any Feedback You voluntarily provide may be used by AMD without restriction, fee or obligation of confidentiality. Accordingly, if You do give AMD Feedback on any version of the Specification, You agree AMD may freely use, reproduce, license, distribute, and otherwise commercialize Your Feedback in any product, as well as has the right to sublicense third parties to do the same. Further, You will not give AMD any Feedback that You may have reason to believe is (i) subject to any patent, copyright or other intellectual property claim or right of any third party; or (ii) subject to license terms which seek to require any product or intellectual property incorporating or derived from Feedback or any Product or other AMD intellectual property to be licensed to or otherwise provided to any third party.

6. You shall adhere to all applicable U.S., European, and other export laws, including but not limited to the U.S. Export Administration Regulations (“EAR”), (15 C.F.R. Sections 730 through 774), and E.U. Council Regulation (EC) No 428/2009 of 5 May 2009. Further, pursuant to Section 740.6 of the EAR, You hereby certifies that, except pursuant to a license granted by the United States Department of Commerce Bureau of Industry and Security or as otherwise permitted pursuant to a License Exception under the U.S. Export Administration Regulations ("EAR"), You will not (1) export, re-export or release to a national of a country in Country Groups D:1, E:1 or E:2 any restricted technology,

software, or source code You receive hereunder, or (2) export to Country Groups D:1, E:1 or E:2 the direct product of such technology or software, if such foreign produced direct product is subject to national security controls as identified on the Commerce Control List (currently found in Supplement 1 to Part 774 of EAR). For the most current Country Group listings, or for additional information about the EAR or Your obligations under those regulations, please refer to the U.S. Bureau of Industry and Security's website at http://www.bis.doc.gov/.

7. If You are a part of the U.S. Government, then the Specification is provided with "RESTRICTED RIGHTS" as set forth in subparagraphs (c) (1) and (2) of the Commercial Computer Software-Restricted Rights clause at FAR 52.227-14 or subparagraph (c) (1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7013, as applicable.

8. This Agreement is governed by the laws of the State of California without regard to its choice of law principles. Any dispute involving it must be brought in a court having jurisdiction of such dispute in Santa Clara County, California, and You waive any defenses and rights allowing the dispute to be litigated elsewhere. If any part of this agreement is unenforceable, it will be considered modified to the extent necessary to make it enforceable, and the remainder shall continue in effect. The failure of AMD to enforce any rights granted hereunder or to take action against You in the event of any breach hereunder shall not be deemed a waiver by AMD as to subsequent enforcement of rights or subsequent actions in the event of future breaches. This Agreement is the entire agreement between You and AMD concerning the Specification; it may be changed only by a written document signed by both You and an authorized representative of AMD.

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.

Linux is a registered trademark of Linus Torvalds.

Contents

List of Tables

Revision History

Date	Revision	Description
August 2020	0.8	**Updates and Additions:** • Updated Section 3.2 Platform State Machine. • Updated Table 11. Command Identifiers. • Updated Section 7.3 Attestation. • Updated Table 21. ATTESTATION_REPORT Structure. • Updated Section 8.3.2 Actions for SNP_PLATFORM_STATUS. • Updated Table 36. Layout of the STRUCT_PLATFORM_STATUS Structure. • Added Section 8.4 SNP_SET_REPORTED_TCB. • Updated Section 8.5.2 Actions for SNP_INIT. • Updated Table 40. Status Codes for SNP_INIT. • Updated Section 8.10.2 Actions for SNP_DF_FLUSH. • Updated Table 50. Status Codes for SNP_DF_FLUSH. • Updated Section 8.11.2 Actions for SNP_SHUTDOWN.
April 2020	0.7	Initial public release.

Chapter 1 Introduction

1.1 Purpose

The purpose of this document is to provide details of the Platform Security Processor (PSP) firmware support for the Secure Nested Paging (SEV-SNP) enhancement to SEV. The PSP exposes a set of functions to the hypervisor for guest lifecycle management of SNP-enabled guests.

1.2 Scope

This document describes the software interface for the functions supported by the PSP for SNP VM management. It does not describe the x86 CPU or System-on-Chip (SOC) hardware support for SNP. While certain sections of this document may describe potential hypervisor usage of the firmware ABI, this document is not intended to prescribe any specific use or hypervisor architecture. Please refer to [APM] for the x86 ISA mechanisms related to SEV-SNP and to the whitepaper [SNP-WP] for a high-level description of SEV-SNP and the features it provides.

1.3 Intended Audience

The intended audience of this document is hypervisor developers, kernel developers, and security architects. Hypervisor developers supporting SNP will need to use the firmware functions described herein for VM lifecycle management. Additionally, kernel developers and security architects will need to use the guest message functions to perform secure attestation, key management, and migration.

1.4 References

Table 1. External References

Reference	Document
APM	AMD64 Architecture Programmer’s Manual (Volumes 1–5) #s 24592, 24593, 24594, 26568, and 26569
PPR	Processor Programming Reference
SNP-WP	AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More
SEV	Secure Encrypted Virtualization API, #55766

Chapter 2 Data Structures and Encodings

This section describes data structures that are common to multiple commands.

2.1 Metadata Entries (MDATA)

Table 2 describes a metadata entry within a metadata page. Metadata entries describe security attributes of pages that have been swapped out. When pages are swapped back in, the firmware uses the metadata entries to ensure the SNP security properties are not violated.

Table 2. Layout of the MDATA Structure

Byte Offset	Bits	Name	Description
00h	63:0	SOFTWARE_DATA	Software available data supplied by the hypervisor.
08h	63:0	IV	Initialization vector used to encrypt the swapped-out page.
10h	127:0	AUTH_TAG	Authentication tag of the swapped-out page.
20h	63:12	GPA	Bits 63:12 of the gPA of the swapped-out page.
	11:5	-	Reserved.
	4	PAGE_SIZE	Indicates the size of the swapped-out page. If set to 0, the page is 4 KB. If set to 1, the page is 2 MB.
	3	METADATA	Indicates that the swapped-out page is a metadata page.
	2	VMSA	Contains RMP.VMSA of the page at the time the page was swapped out.
	1	PAGE_VALIDATED	Contains RMP.Validated of the page at the time the page was swapped out.
	0	VALID	Indicates this metadata entry is valid.
28h	31:24	VMPL3	The permission mask RMP.VMPL3 of the page at the time the page was swapped out.
	23:16	VMPL2	The permission mask RMP.VMPL2 of the page at the time the page was swapped out.
	15:8	VMPL1	The permission mask RMP.VMPL1 of the page at the time the page was swapped out.
	7:0	VMPL0	The permission mask RMP.VMPL0 of the page at the time the page was swapped out.
2Ch	31:0	-	Reserved.
30h	63:0	-	Reserved.
38h	63:0	-	Reserved.

2.2 TCB_VERSION

The TCB version is a version string that represents the version of the firmware. The TCB version is described in Table 3.

Table 3. Structure of the TCB_VERSION Version String

Bits	Field	Description
63:56	MICROCODE	Lowest current patch level of all the cores.
55:48	SNP	Version of the SNP firmware Security Version Number (SVN) of SNP firmware.
47:16	-	Reserved.
15:8	TEE	Current PSP OS version SVN of PSP operating system.
7:0	BOOT_LOADER	Current bootloader version SVN of PSP bootloader.

2.3 VCEK

The Versioned Chip Endorsement Key (VCEK) is a root key derived from chip-unique secrets and the current TCB version. The VCEK can be computed for previous versions of the TCB, allowing for migrations of secrets from previous version to the current version.

2.4 Invalid Physical Address (PADDR_INVALID)

The value PADDR_INVALID represents an invalid value for sPA and gPA fields in this specification. PADDR_INVALID is defined as two’s-complement –1 with a width of the field to which it is assigned. Note that 0h is a valid sPA and gPA.

Chapter 3 Platform Management

Before SNP VMs can be launched, the platform must be properly configured and initialized. Platform initialization is accomplished via the SNP_INIT command, which verifies that SNP has been enabled across all CPUs and configured correctly. Further, the platform contains a state machine that restricts which commands may be executed at certain times throughout execution.

3.1 Feature Detection and Enablement

On initialization, the SNP_INIT command will check that the SEV-SNP feature is available and globally enabled. See [APM] Volume 2, Section 15.36, for information on feature detection and enablement.

3.2 Platform State Machine

The SNP firmware may exist in two states: UNINIT and INIT. Certain commands may be executed only in each of these states.

Table 4. Commands Available in Each State

State	Encoding	Description	Allowed Platform Commands
UNINIT	0h	The platform is uninitialized. This is the reset state of the PSP firmware.	SNP_INIT SNP_PLATFORM_STATUS DOWNLOAD_FIRMWARE GET_ID
INIT	1h	The platform is initialized	All SNP commands except SNP_INIT, DOWNLOAD_FIRMWARE

Chapter 4 Guest Management

The lifecycles of SNP-enabled guests are managed through the guest management ABI functions. SNP-enabled guests are identified via their guest context pages, and may be launched, attested, migrated, etc. via the appropriate ABI calls. An SNP-enabled guest is created by first allocating a context page, then activating the guest on a specific ASID, and then adding an initial set of plaintext pages into the guest address space. After the guest has begun execution, it may request attestation reports, derived keys, and assist in scenarios such as live migration directly through a trusted channel with the PSP firmware.

4.1 Guest Context

The guest context (represented as GCTX throughout this specification) contains all the information, keys, and metadata associated with the guest that the firmware tracks to implement the SEV and SNP features. The guest context is specified in Table 5.

Table 5. Fields of the Guest Context (GCTX)

Field	Migrated?	Description
ASID	No	The ASID that the guest's keys are installed on, if at all.
State	Yes	The current state of the guest.
MsgCount0	Yes	The number of guest messages that the firmware has sent to or received from VMPL0.
MsgCount1	Yes	The number of guest messages that the firmware has sent to or received from VMPL1.
MsgCount2	Yes	The number of guest messages that the firmware has sent to or received from VMPL2.
MsgCount3	Yes	The number of guest messages that the firmware has sent to or received from VMPL3.
Policy	Yes	The guest's security policy.
MA	No	The migration agent of the guest, if the guest is associated with a migration agent.
LD	Yes	The launch digest context used to measure the guest during the launch command flow.
OEK	Yes	The offline encryption key associated with this guest.
VEK	No	The VM encryption key used to encrypt the guest's memory.
VMPCK0, VMPCK1, VMPCK2, VMPCK3	Yes	The VM communication keys.

Field	Migrated?	Description
VMRK	Yes	The VM root key provided by the MA at guest launch or guest import.
HostData	Yes	Host data provided by the hypervisor during guest launch. This firmware includes this value in all attestation reports for this guest.
IDBlockEn	Yes	Indicates whether an ID block was associated with the guest.
IDBlock	Yes	The associated ID block, if any.
IDKeyDigest	Yes	The ID key digest, if any.
AuthorKeyEn	Yes	Indicates whether an Author key signed the ID key.
AuthorKeyDigest	Yes	The Author key digest, if any.
ReportID	Yes	Attestation report ID.
RootMDEntry	Yes	The root metadata entry.
IMD	Yes	The measurement of the Incoming Migration Image (IMI).
IMIEn	No	Indicates whether the current launch flow is an IMI migration or not. Used only when the guest is in the GSTATE_LAUNCH state.

The firmware stores the guest context in a page donated by the hypervisor. The hypervisor donates the page through the SNP_GCTX_CREATE command and reclaims it with SNP_PAGE_RECLAIM command. Because the guest context page is in the Context state (see Chapter 5 for details on the page state machine), the hypervisor cannot write to the page. The firmware prevents the hypervisor from reading from the page by encrypting the guest context.

4.2 Guest State Machine

The commands that can be successfully issued for a guest are restricted according to an internal guest state machine. The guest state machine ensures that commands are executed in the correct order. The current guest state is stored in GCTX.State.

Table 6. Guest State Definition

State	Encoding	Description	Allowed Guest Commands
GSTATE_INIT	0h	The initial state of the guest.	SNP_LAUNCH_START SNP_GUEST_REQUEST (VM_IMPORT) SNP_PAGE_RECLAIM SNP_DECOMMISSION

State	Encoding	Description	Allowed Guest Commands
GSTATE_LAUNCH	1h	The guest is being launched.	SNP_GCTX_CREATE SNP_LAUNCH_UPDATE SNP_LAUNCH_FINISH SNP_ACTIVATE SNP_DECOMMISSION SNP_PAGE_RECLAIM SNP_PAGE_MOVE SNP_PAGE_SWAP_OUT SNP_PAGE_SWAP_IN SNP_PAGE_UNSMASH
GSTATE_RUNNING	2h	The guest is currently running.	SNP_ACTIVATE SNP_DECOMMISSION SNP_PAGE_RECLAIM SNP_PAGE_MOVE SNP_PAGE_SWAP_OUT SNP_PAGE_SWAP_IN SNP_PAGE_UNSMASH SNP_GUEST_REQUEST

Table 7. Guest State Transitions

Command	Start State	End State
SNP_LAUNCH_START	GSTATE_INIT	GSTATE_LAUNCH
SNP_LAUNCH_FINISH	GSTATE_LAUNCH	GSTATE_RUNNING
VM_ABSORB	GSTATE_LAUNCH	GSTATE_RUNNING
VM_IMPORT	GSTATE_INIT	GSTATE_RUNNING

4.3 Guest Policy

The firmware associates each guest with a guest policy that the guest owner provides. The firmware restricts what actions the hypervisor can take on this guest according to the guest policy. The policy also indicates the minimum firmware version to for the guest.

The guest owner provides the guest policy to the firmware during launch. The firmware then binds the policy to the guest. The policy cannot be changed throughout the lifetime of the guest. The policy is also migrated with the guest and enforced by the destination platform firmware.

The guest policy is an 8-byte structure with the fields shown in Table 8.

Table 8. Guest Policy Structure

Bit(s)	Name	Description
63:20	-	Reserved. MBZ.
19	DEBUG	0: Debugging is disallowed. 1: Debugging is allowed.
18	MIGRATE_MA	0: Association with a migration agent is disallowed. 1: Association with a migration agent is allowed.
17	-	Reserved. Must be one.
16	SMT	0: SMT is disallowed. 1: SMT is allowed.
15:8	ABI_MAJOR	The minimum ABI major version required for this guest to run.
7:0	ABI_MINOR	The minimum ABI minor version required for this guest to run.

The policy bits for a given guest are referenced with the format POLICY.<FLAG_NAME>. For instance, the flag indicating that SMT is allowed is referred to as POLICY.SMT.

4.4 Guest Activation

The processor associates each guest memory transaction with the Address Space Identifier (ASID) specified in the guest's VMCB. The ASID of a guest selects the key used by the memory controller to encrypt that guest's memory. The hypervisor must inform the firmware on which ASID it will execute the guest with using the VMRUN instruction. The firmware then installs the guest's VEK in the key slot associated with that ASID. To inform the firmware of the guest-ASID binding, the hypervisor calls SNP_ACTIVATE.

All guest data in the caches and data fabric write buffers are unencrypted. Guests with different ASIDs have logically separate caches. However, guests with the same ASID share cache lines. To ensure that a previously decommissioned guest's data are not accessible to a new guest, SNP_ACTIVATE will require that the caches are invalidated and that the data fabric write buffers are flushed. In this case, the hypervisor must first invoke WBINVD on all cores. Following the WBINVD completion, the hypervisor must invoke the SNP_DF_FLUSH command. This ensures that no plaintext data owned by another guest exist in the caches or in the write buffers before activation.

The hypervisor can activate a guest on a subset of core complexes using SNP_ACTIVATE_EX. If a guest is activated on a core complex, the hypervisor may execute the guest with that ASID on only that core complex.

Guest activation must always occur before any memory is assigned to the guest by the hypervisor using the RMPUPDATE instruction.

4.5 Launching a Guest

The hypervisor starts an SNP guest by launching the guest. The hypervisor uses the commands SNP_LAUNCH_START, SNP_LAUNCH_UPDATE, and SNP_LAUNCH_FINISH to launch the guest.

SNP_LAUNCH_START begins the launch process. Through this command, the firmware initializes a cryptographic digest context used to construct the measurement of the guest. If the guest is expected to be migrated, SNP_LAUNCH_START also binds a Migration Agent (MA) to the guest. (See 4.11 for further information about migration.)

SNP_LAUNCH_UPDATE inserts data into the guest's memory. The firmware extends the cryptographic digest context with the data to bind the measurement of the guest with all operations that the hypervisor took on the guest's memory contents.

SNP_LAUNCH_UPDATE can insert two special pages into the guest's memory: the secrets page and the CPUID page. The secrets page contains encryption keys used by the guest to interact with the firmware. Because the secrets page is encrypted with the guest's memory encryption key, the hypervisor cannot read the keys. The CPUID page contains hypervisor provided CPUID function values that it passes to the guest. The firmware validates these values to ensure the hypervisor is not providing out-of-range values.

SNP_LAUNCH_FINISH finalizes the cryptographic digest and stores it as the measurement of the guest at launch. This measurement is a critical part of the guest's attestation report produced by the firmware. This command also takes identity keys to be associated with guest used as part of the attestation report. For further information about the identity bloc, see 4.6. For attestation, see 4.9.

After SNP_LAUNCH_FINISH completes successfully, the hypervisor may invoke VMRUN on the x86 CPU to execute the guest.

4.6 Identity Block

As part of the input to the SNP_LAUNCH_FINISH command, the hypervisor may provide an optional data structure called the identity block. The identity block contains the expected launch digest of the guest, information uniquely identifying the guest, the guest policy bitfield, and a signature by the guest owner. The provided launch digest is checked against the computed launch digest, and the provided policy is checked against the policy used to launch the guest. The identifying information is stored in the guest context to be used during key derivation and attestation. Finally, the firmware will check that the signature is valid.

The firmware stores the keys used to sign the identity block in the guest context. Attestation reports for the guest contain the public keys to reflect the binding of the guest to the guest owner. A guest owner that sees its public keys in the attestation report knows that the launch process used an identity block provided by that guest owner to validate the guest.

4.7 Decommissioning a Guest

The hypervisor may decommission a guest by calling SNP_DECOMMISSION on the guest context page. The firmware prevents the hypervisor from running a decommissioned guest by marking the guest's ASID as unusable. Further, the firmware transitions the guest context page to a Firmware page, thus rendering the context page unusable.

4.8 Guest Messages

During the launch sequence, a special secrets page may be inserted that contains VM Platform Communication Keys (VMPCKs) that may be used by the guest to send and receive secure messages to the PSP. Guests encrypt messages as described in the SNP_GUEST_REQUEST function before presenting the encrypted payload to the hypervisor. The hypervisor in turn calls SNP_GUEST_REQUEST and returns the result (also encrypted with the VMPCK) to the guest. Guest messages are used for getting attestation reports, derived keys, handling migration, and other uses.

4.9 Remote Attestation

Guests may ask the PSP to generate an attestation report on their behalf via a SNP_GUEST_REQUEST call. The guest may ask for an attestation report at any time and multiple reports can be generated. When the guest asks for a report, it supplies 512 bits of arbitrary data to be included in the report. The resulting report will contain this data, identity information about the guest (from the launch sequence), migration, and policy information. The report is signed by VCEK, a chip-unique key specific to the current TCB version.

Guests may supply attestation reports to 3rd parties to establish trust. The 3rd party should verify the authenticity of the report based on its signature. A successful signature verification proves that the 512 bits of guest data supplied in the report came from the guest whose identity is described. For instance, this may be used to securely associate a public key with a particular VM instance.

4.10 Guest Keys

Guests may ask the PSP to derive keys for them based on various information via a SNP_GUEST_REQUEST call. Keys are either rooted in a VM Root Key (VMRK) that is supplied as part of the launch flow (and migrates with the guest), or in the VCEK, which is machine specific. When asked for a key, the PSP uses a key derivation function (KDF) to generate the requested key based on the root value and additional parameters. Certain pieces of guest information are always mixed into the derived key while others may be optionally mixed when requested by the guest. Keys may be used to seal information to the identity of the guest, or for other purposes.

4.11 Migration

Migration is supported in the SNP architecture through Migration Agents (MAs). A Migration Agent is itself an SNP VM that is bound to the primary VM during the launch process. A VM may be associated only with a single MA, but a single MA may manage multiple primary VMs. The MA is responsible for supplying the VMRK during the launch process and for enforcing the guest migration policy.

The MA is considered part of the guest VM’s TCB. Consequently, when a guest generates an attestation report, the report includes information about the MA associated with the guest (if one exists). A 3rd party verifying the attestation report of a guest should also verify the report of the guest’s MA.

The hypervisor may migrate a guest with or without the assistance of the guest. 4.12 describes how a hypervisor migrates with the assistance of the guest. When the hypervisor wishes to migrate a guest without the assistance of the guest, it first swaps all guest memory and associated metadata pages using the SNP_PAGE_SWAP_OUT command (see Chapter 5 for additional details). Swapped pages are encrypted using the OEK (Offline Encryption Key). Because each swapped page must be associated with a metadata entry, eventually there will be a single metadata page remaining after all other pages are swapped. When the hypervisor swaps this page, it can choose to store its metadata entry in the special RootMDEntry field in the guest context.

After all the guest memory is swapped, the hypervisor asks the MA to perform the VM_EXPORT function via SNP_GUEST_REQUEST. This function sends the context page of the guest to be migrated to the MA via an encrypted channel. At this point, the primary VM is no longer runnable.

The MA sends the VM context to a trusted location, such as a MA on a new machine. The mechanism that the MA uses to transfer this data and enforce security on it is outside the scope of this document.

In a typical scenario, a MA will have started on the destination machine to receive the guest context information. After the hypervisor creates a guest context (as described earlier) it may ask the MA to perform the VM_IMPORT function (via SNP_GUEST_REQUEST), which installs the provided guest context on the new machine. At this point, the hypervisor may proceed with swapping in guest memory (via SNP_PAGE_SWAP_IN) and begin executing the guest.

The use of an MA is optional and SNP guests may be started without a MA. Guests that are started without a MA may not be exported and therefore cannot be migrated without shutting themselves down.

4.12 Guest Assisted Migration

If the guest has an Initial Migration Image (IMI), the guest may assist the hypervisor during the migration process to increase migration throughput. An IMI is software measured during the guest launch process that can reconstruct a guest on the receiving platform from pages it is sent by the sending guest.

On launch, a subset of the pages launched may be marked as part of the IMI. The launch process measures the IMI separately into the Initial Migration Digest (IMD) and is stored in the guest context. To start a migration operation, the cloud provider performs a modified launch flow on the receiving platform. This launch flow differs from normal launch in two important ways:

- Only the IMI pages are launched via SNP_LAUNCH_UPDATE
- SNP_LAUNCH_FINISH is replaced by the absorb guest message

The absorb guest message takes a guest context exported by the sending machine using the export guest message. The absorb message differs from the import message mainly by overwriting the IMI context with the incoming guest context. However, the absorb message requires that the launch digest of the IMI matches the IMD of the migrated guest. This ensures that the receiving IMI is exactly the IMI that was launched with the guest.

When the guest is exported on the sending platform for the purpose of guest assisted migration, the guest remains runnable. This allows the guest to send its own memory contents to the IMI.

Chapter 5 Page Management

5.1 Page Security Attributes

The Reverse Map Table (RMP) is a structure that resides in DRAM and maps system physical addresses (sPAs) to guest physical addresses (gPAs). There is only one RMP for the entire system, which is configured using x86 model specific registers (MSRs). See [APM] volume 2, Section 15.36, for details.

Each RMP entry is indexed by the sPA the page. The RMP, combined with all guests' nested page tables, creates a global one-to-one mapping between sPAs and gPAs. That is, the RMP ensures that a page cannot be mapped into multiple guests at once, and it cannot be mapped multiple times into a single guest at once.

The RMP also contains various security attributes of each that are managed by the hypervisor through hardware-mediated and firmware-mediated controls. The fields of an RMP entry are described in [APM] volume 2, Section 15.36.3.

5.2 Page States

A page's state is completely determined by the fields in the page's RMP entry. Specifically, the page state depends on the Assigned, Validated, ASID, Immutable, GPA, and VMSA RMP entry fields. Table 9 enumerates and defines each of the page states. Note that (-) in a cell indicates that the page state is not dependent on that field.

Table 9. Page State Definitions

Page State	Assigned	Validated	ASID	Immutable	GPA	VMSA
Hypervisor	0	0	0	0	-	-
Reclaim	1	0	0	0	-	-
Firmware	1	0	0	1	0	0
Context	1	0	0	1	0	1
Metadata	1	0	0	1	>0	-
Pre-Guest	1	0	>0	1	-	-
Guest-Invalid	1	0	>0	0	-	-
Pre-Swap	1	1	>0	1	-	-
Guest-Valid	1	1	>0	0	-	-
Default	See discussion below.					

A Hypervisor page is used by the hypervisor for its normal execution. SNP places no restrictions on the use of Hypervisor pages for purposes outside of managing SNP guests. A Default page is a page that does not have an RMP entry. Pages do not have an RMP entry if the sPA indexes to an entry past the end of the RMP table—that is, past RMP_END. Default pages have the same access permissions as a Hypervisor page but cannot be transitioned to any other page state.

Pages in the Firmware state are owned by the firmware. Because the RMP.Immutable bit is set, the hypervisor cannot write to Firmware pages nor alter the RMP entry with the RMPUPDATE instruction. A Firmware page is used by the hypervisor to donate writeable memory to the firmware to operate on. Such pages may be used to output data to the hypervisor, or to transition into a special page state, such as Metadata pages or Context pages.

When an immutable page is returned to the hypervisor by the firmware, the page is transitioned into the Reclaim page state. The Reclaim page state can then be transitioned to other non-immutable pages by the hypervisor using RMPUPDATE.

A Context page is a firmware-owned page that contains all context information of a guest. The format of the Context page is implementation specific. The content of a Context page is encrypted and integrity protected so that the hypervisor cannot not read or write to it.

A Metadata page is a firmware-owned page that contains the metadata of a swapped-out page. Metadata pages have a well-defined format. The firmware converts a Firmware into a Metadata page by making the GPA field non-zero.

A Guest-Invalid page has been donated to the guest but has not yet been validated by the guest. If the hypervisor wishes to have the firmware operate on them, the hypervisor transitions the page into a Pre-Guest page.

Similarly, a Guest-Valid page has been donated to the guest, and the guest has validated the page. If the hypervisor wishes to have the firmware operate on them, the hypervisor transitions the page into a Pre-Swap page.

5.3 Page State Transitions

The only ways in which a page can transition between states are by invoking the RMPUPDATE and PVALIDATE instructions or by issuing firmware commands described in this specification. The hardware and firmware mediate all page state transitions to ensure that only secure state transitions occur.

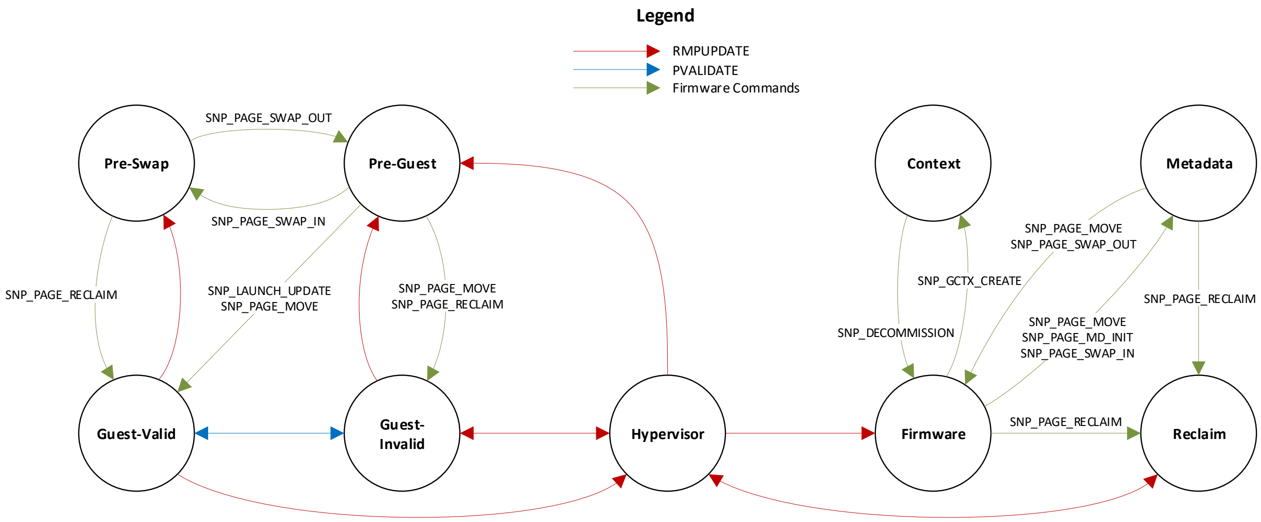

Figure 1. SNP Page State Machine

Red edges in Figure 1 represent hypervisor actions. Blue edges represent guest actions. Green edges represent firmware commands specified in this document. Note that the some transitive RMPUPDATE edges are omitted for clarity.

Actions that trigger a page state transition are depicted in Figure 1. The following subsections describe the transitions in further detail. Notably, the following subsections do not describe the Default page state because Default pages cannot transition to other page states.

5.3.1 RMPUPDATE

The RMPUPDATE instruction may be used by the hypervisor to alter the RMP entries of pages. This allows the hypervisor to directly alter the state of most pages.

Notably, RMPUPDATE can invalidate a page but cannot validate a page. This means that the hypervisor cannot produce pages in the Pre-Swap or Guest-Valid states without assistance from a guest or from the PSP firmware.

Also, RMPUPDATE cannot affect the page state of an immutable page. A hypervisor can produce pages in the Pre-Guest or Pre-Swap states with RMPUPDATE. However, once in those states, the hypervisor must rely on the PSP firmware to transition them.

5.3.2 PVALIDATE

The PVALIDATE instruction may be used by a guest to alter the Validated flag of a page. This allows a guest to signal to the hardware and firmware that the page at a specified gPA is validated—that is, the guest expects the hardware and firmware to protect the integrity of the page.

Because PVALIDATE can be executed only within the guest, PVALIDATE can operate only on pages addressable within the guest's physical address space. Further, Pre-Guest, and Pre-Swap pages have their RMP.Immutable flags equal to 1, which prevents the guest from transitioning them.

5.3.3 Page Management Commands

The hypervisor can invoke the commands described in this specification to manage memory without violating the security provided by SNP.

The hypervisor must perform these actions using RMPUPDATE. This restriction allows RMPUPDATE to mediate all re-assignments of pages so that the appropriate TLB and cache operations happen.

5.3.4 Launch Commands

The SNP_GCTX_CREATE command transitions a page from the Firmware state to the Context state. This is the only way a Context page can be created.

The launch commands, specifically SNP_LAUNCH_UPDATE, take unencrypted guest pages and convert them into encrypted pages. In doing so, this command also transitions the launched pages to Guest-Valid pages.

5.3.5 Guest Request Commands

Neither the SNP_GUEST_REQUEST command itself nor any of the guest messages alter the state of the pages passed to it.

5.3.6 Platform Commands

SNP_INIT initializes the state of all pages within the system by initializing the RMP. Other platform commands do not alter the state of any pages.

5.3.7 SEV Legacy Commands

The behavior of the SEV-legacy commands is altered when the SNP firmware is in the INIT state. In this case, the SEV-legacy commands require any page that the SEV-legacy command writes to be a Firmware or Default page.

5.4 Metadata Entries

A metadata entry contains security attributes associated with a swapped-out page. A Metadata page can describe three types of swapped-out pages: Data pages, Metadata pages, or VMSA pages. Each page type determines how the metadata entry is constructed.

Table 10 describes the contents of a metadata entry. All references to RMP fields or addresses refer to the attributes of the page at the time the hypervisor swapped it out.

Table 10. Contents of Metadata Entries for Swapped-Out Data Pages, VMSA Pages, and Metadata Pages

Field	Data Page	VMSA Page	Metadata Page
SOFTWARE_DATA	Software-provided data	Software-provided data	Software-provided data
IV	Initialization vector	Initialization vector	Initialization vector
AUTH_TAG	Authentication tag	Authentication tag	Authentication tag
PAGE_SIZE	RMP.Page_Size	RMP.Page_Size	RMP.Page_Size
VALID	1	1	1
METADATA	0	0	1
VMSA	0	1	0
GPA	gPA of the page	gPA of the page	PADDR_INVALID
PAGE_VALIDATED	RMP.Validated	RMP.Validated	0
VMPL0	RMP.VMPL0 if VMPLs are enabled. 0h otherwise.	RMP.VMPL0 if VMPLs are enabled. 0h otherwise.	0h
VMPL1	RMP.VMPL1 if VMPLs are enabled. 0h otherwise.	RMP.VMPL1 if VMPLs are enabled. 0h otherwise.	0h
VMPL2	RMP.VMPL2 if VMPLs are enabled. 0h otherwise.	RMP.VMPL2 if VMPLs are enabled. 0h otherwise.	0h
VMPL3	RMP.VMPL3 if VMPLs are enabled. 0h otherwise.	RMP.VMPL3 if VMPLs are enabled. 0h otherwise.	0h
Reserved fields	0h	0h	0h

The hypervisor may request that the firmware place data into SOFTWARE_DATA for its own purposes. The firmware never interprets this field. Because the hypervisor can read the metadata entries in Metadata pages, the hypervisor can use SOFTWARE_DATA for its own bookkeeping purposes.

An entry with VALID set to 0h is invalid and does not refer to any swapped-out page. When VALID is 0, the firmware does not interpret any other fields of the entry.

Chapter 6 Mailbox Protocol

Software on the x86 CPUs communicate with the PSP through a set of MMIO registers, referred to as mailbox registers. This ABI used the mailbox protocol defined in Chapter 4 of [SEV]. This ABI adds new commands and status codes, which extend the SEV mailbox protocol. These command and status codes are described in the following sections.

6.1 Command Identifier

This ABI adds many new commands to be handled by the mailbox protocol. Table 11 summarizes the additional commands and their identifiers. See the command definitions for further details.

Table 11. Command Identifiers

Command	ID	Description
SNP_INIT	81h	Initialize platform for SNP.
SNP_SHUTDOWN	82h	Un-initialize platform for SNP.
SNP_PLATFORM_STATUS	83h	Query platform information.
SNP_DF_FLUSH	84h	Flush data fabric buffers.
SNP_DECOMMISSION	90h	Destroy a guest context.
SNP_ACTIVATE	91h	Assign an ASID to a guest.
SNP_GUEST_STATUS	92h	Query guest information.
SNP_GCTX_CREATE	93h	Create a guest context.
SNP_GUEST_REQUEST	94h	Process a guest request.
SNP_ACTIVATE_EX	95h	Assign an ASID to a guest on select cores.
SNP_LAUNCH_START	A0h	Begin to launch a new guest.
SNP_LAUNCH_UPDATE	A1h	Add memory to a launching guest.
SNP_LAUNCH_FINISH	A2h	Complete launching a guest.
SNP_DBG_DECRYPT	B0h	Decrypt guest memory for debugging.
SNP_DBG_ENCRYPT	B1h	Encrypt guest memory for debugging.
SNP_PAGE_SWAP_OUT	C0h	Swap a page out of guest memory.
SNP_PAGE_SWAP_IN	C1h	Swap a page into guest memory.
SNP_PAGE_MOVE	C2h	Move a Memory page.
SNP_PAGE_MD_INIT	C3h	Initialize a Metadata page.
SNP_PAGE_RECLAIM	C7h	Clear the immutable bit on a page.
SNP_PAGE_UNSMASH	C8h	Convert a sequence of 4 k pages into a 2 MB page.

SNP_SET_REPORTED_TCB	C9h	Set the system wide ReportedTcb value

6.2 Status Codes

This ABI introduces several new status codes to the mailbox protocol. Table 12 summarizes the additional status codes added by this ABI.

Table 12. Status Codes

Status	Code	Description
INVALID_PAGE_SIZE	19h	The RMP page size is incorrect.
INVALID_PAGE_STATE	1Ah	The RMP page state is incorrect.
INVALID_MDATA_ENTRY	1Bh	The metadata entry is invalid.
INVALID_PAGE_OWNER	1Ch	The page ownership is incorrect.
AEAD_OFLOW	1Dh	The AEAD algorithm would have overflowed.

Chapter 7 Guest Messages

Guest messages provide the guest a mechanism to communicate with the PSP without risk from a malicious hypervisor who wishes to read, alter, drop, or replay the messages sent. A guest may issue requests of firmware via the SNP_GUEST_REQUEST command. This command constructs a trusted channel between the guest and the PSP firmware. The hypervisor cannot alter the messages without detection nor read the plaintext of the messages.

The firmware constructs the channel using a Virtual Machine Platform Communication key (VMPCK). Each guest has four VMPCKs, which the firmware generates and provides to the guest in a special secrets page as part of the guest launch process (see SNP_LAUNCH_UPDATE in Section 8.13 details). Only the guest and the firmware possess the VMPCKs.

Each message contains a sequence number per VMPCK. The sequence number is incremented with each message sent. Messages sent by the guest to the firmware and by the firmware to the guest must be delivered in order. If not, the firmware will reject subsequent messages by the guest when it detects that the sequence numbers are out of sync.

Each message is protected with an Authenticated Encryption with Associated Data algorithm (AEAD), namely AES-256 GCM.

Details on how to send a message via the SNP_GUEST_REQUEST command can be found in Section 8.22.

7.1 CPUID Reporting

Note: This guest message may be removed in future versions as it is redundant with the CPUID page in SNP_LAUNCH_UPDATE (see Section 8.13).

The firmware provides a service to the guest to validate CPUID function values provided by the hypervisor. This ensures that CPUID function values provided by the hypervisor are within range of the hardware. To use this service, the guest constructs an MSG_CPUID_REQ message.

The guest constructs an MSG_CPUID_REQ message as defined in Table 13. This message contains an array of CPUID function structures as defined in Table 13. The guest fills the structure with the information the guest received from the CPUID instruction from the hypervisor.

The message contains enough space for COUNT_MAX function structures, but only COUNT function structures are valid. COUNT_MAX is 64.

Table 13. MSG_CPUID_REQ Structure

Byte Offset	Bits	Name	Description
00h	31:0	COUNT	Number of CPUID functions to validate. Must be less than COUNT_MAX.
04h	31:0	-	Reserved. Must be zero.
08h	63:0	-	Reserved. Must be zero.
10h		CPUID_FUNCTION[]	COUNT_MAX number of CPUID_FUNCTION records. Only the first COUNT records are valid.

Table 14. CPUID_FUNCTION Structure

Byte Offset	Bits	Name	Description
00h	31:0	EAX_IN	EAX input parameter to CPUID.
04h	31:0	ECX_IN	ECX input parameter to CPUID.
08h	63:0	XCR0_IN	XCR0 at the time of CPUID execution.
10h	63:0	XSS_IN	IA32_XSS MSR at the time of CPUID execution.
18h	31:0	EAX	EAX output parameter of CPUID.
1Ch	31:0	EBX	EBX output parameter of CPUID.
20h	31:0	ECX	ECX output parameter of CPUID.
24h	31:0	EDX	EDX output parameter of CPUID.
28h	63:0	-	Reserved. Must be zero.

The firmware returns an MSG_CPUID_RSP message as defined in Table 15. The message contains the same CPUID function structures that may be altered by the firmware. The firmware will alter the function structure only when the hypervisor promised a feature that is not truly available. However, the firmware will never indicate features exists that the hypervisor did not report—even if those features are present. This allows the hypervisor to advertise a subset of the features of the platform.

If the firmware encounters a CPUID function that is not architecturally defined for the current platform, the firmware sets the EAX_IN, ECX_IN, and XCR0_IN fields of the function to 0xFFFFFFFF. This indicates to the guest that the CPUID function is not known to the firmware and thus the firmware is unable to confirm the function’s value.

Table 15. MSG_CPUID_RSP Structure

Byte Offset	Bits	Name	Description
00h	31:0	STATUS	The status of key derivation operation. 0h: Success. 16h: Invalid parameters.
04h	31:0	COUNT	Number of CPUID functions that have been validated.
08h	63:0	-	Reserved.
10h		CPUID_FUNCTION[]	COUNT_MAX number of CPUID_FUNCTION records. Only the first COUNT records are valid.

7.2 Key Derivation

The guest can ask the firmware to provide a key derived from a root key. This key may be used by the guest for any purpose it chooses, such as sealing keys or communicating with external entities.

The data that the firmware mixes into the derived key is described in Table 16. The firmware unconditionally mixes some of the fields into the key while the guest may optionally select and even supply other data to mix into the key.

Table 16. Data Mixed into the Derived Guest Key

Data	Description	Mix Type	Provided in Message
VCEK/VMRK	VCEK or VMRK of the guest. The guest selects which of the keys is used.	Always	No
VMPL	The VMPL selected by the guest.	Always	Yes
Host Data	The host data provided at launch.	Always	No
ID key/ Author key	The author key provided at launch. If an author key was not provided, then the firmware uses the ID key instead.	Always	No
Guest Field Selection	A bitmask describing which of the fields in this table are mixed into the key. This covers the guest-selectable fields as well as other field selection done by the firmware.	Always	Yes
TCB Version	The TCB version selected by the guest.	Optional	Yes
Guest SVN	SVN of the guest.	Optional	Yes
Measurement	The measurement of the guest at launch.	Optional	No
Family ID	The family ID provided at launch.	Optional	No

Data	Description	Mix Type	Provided in Message
Image ID	The image ID provided at launch.	Optional	No
Guest Policy	The guest policy provided at launch.	Optional	No

Table 17 describes the MSG_KEY_REQ message structure that the guest sends to the firmware to request a derived key.

Table 17. MSG_KEY_REQ Message Structure

Byte Offset	Bits	Name	Description
0h	31:1	-	Reserved. Must be zero.
	0	ROOT_KEY_SELECT	Selects the root key to derive the key from. 0 indicates VCEK. 1 indicates VMRK.
4h	31:0	-	Reserved. Must be zero.
8h	63:0	GUEST_FIELD_SELECT	Bitmask indicating which data will be mixed into the derived key. See Table 16 for the structure of this bitmask.
10h	31:0	VMPL	The VMPL to mix into the derived key. Must be greater than or equal to the current VMPL.
14h	31:0	GUEST_SVN	The guest SVN to mix into the key. Must not exceed the guest SVN provided at launch.
18h	63:0	TCB_VERSION	The TCB version to mix into the derived key. Must not exceed the current TCB version.

The MSG_KEY_REQ described in Table 17 describes the MSG_REQ message structure that the guest sends to the firmware to request a derived key. GUEST_FIELD_SELECT indicates which guest-selectable fields will be mixed into the key that is described in Table 18.

Table 18. Structure of the GUEST_FIELD_SELECT Field

Bits	Field	Description
63:6	-	Reserved. Must be zero.
5	TCB_VERSION	Indicates that the guest-provided TCB version string will be mixed into the key.
4	GUEST_SVN	Indicates that the guest-provided SVN will be mixed into the key.
3	MEASUREMENT	Indicates the measurement of the guest during launch will be mixed into the key.
2	FAMILY_ID	Indicates the family ID of the guest will be mixed into the key.
1	IMAGE_ID	Indicates that the image ID of the guest will be mixed into the key.
0	GUEST_POLICY	Indicates that the guest policy will be mixed into the key.

The firmware returns the MSG_KEY_RSP message defined Table 19 to the guest.

Table 19. MSG_KEY_RSP Message Structure

Byte Offset	Bits	Name	Description
00h	31:0	STATUS	The status of key derivation operation. 0h: Success. 16h: Invalid parameters.
04h–1Fh		-	Reserved.
20h	255:0	DERIVED_KEY	The requested derived key if STATUS is 0h.

7.3 Attestation

The guest can request that the firmware construct an attestation report. External entities can use an attestation report to assure the identity and security configuration of the guest.

A guest requests an attestation report by constructing an MSG_REPORT_REQ as specified in Table 20. The message contains data provided by the guest in REPORT_DATA to be included into the report; the firmware does not interpret this data.

Table 20. MSG_REPORT_REQ Message Structure

Byte Offset	Bits	Name	Description
00h	511:0	REPORT_DATA	Guest-provided data to be included into the attestation report
40h	31:0	VMPL	The VMPL to put into the attestation report. Must be greater than or equal to the current VMPL and at most three.
44h–5Fh		-	Reserved. Must be zero.

The guest may generate attestation reports for VMPLs that are greater than or equal to the current VMPL. The desired VMPL is provided by the guest in the request message.

Upon receiving a request for an attestation report, the firmware constructs the report according to Table 21.

The firmware generates a report ID for each guest that persists with the guest instance throughout its lifetime. In each attestation report, the report ID is placed in REPORT_ID. If the guest has a migration agent associated with it, the REPORT_ID_MA is filled in with the report ID of the migration agent.

The firmware signs the attestation report with its VCEK. The firmware uses the system wide ReportedTcb value as the TCB version to derive the VCEK. This value is set by the hypervisor. The firmware guarantees that the ReportedTcb value is never greater than the installed TCB version.

Table 21. ATTESTATION_REPORT Structure

Byte Offset	Bits	Name	Description
00h	31:0	VERSION	Version number of this attestation report. Set to 1h for this specification.
04h	31:0	GUEST_SVN	The guest SVN.
08h	63:0	POLICY	The guest policy. See Table 8 for a description of the guest policy structure.
10h	127:0	FAMILY_ID	The family ID provided at launch.
20h	127:0	IMAGE_ID	The image ID provided at launch.
30h	31:0	VMPL	The request VMPL for the attestation report.
34h	31:0	SIGNATURE_ALGO	The signature algorithm used to sign this report. 102h indicates ECDSA P-384 with SHA-384. All other encodings are reserved.
38h	63:0	PLATFORM_VERSION	The install version of firmware.
40h	63:0	PLATFORM_INFO	Information about the platform. See Table 22.
48h	31:1	-	Reserved. Must be zero.
	0	AUTHOR_KEY_EN	Indicates that the digest of the author key is present in AUTHOR_KEY_DIGEST. Set to the value of GCTX.AuthorKeyEn.
4Ch	31:0	-	Reserved. Must be zero.
50h	511:0	REPORT_DATA	Guest-provided data.
90h	383:0	MEASUREMENT	The measurement calculated at launch.
C0h	255:0	HOST_DATA	Data provided by the hypervisor at launch.
E0h	383:0	ID_KEY_DIGEST	SHA-384 digest of the ID public key that signed the ID block provided in SNP_LANUNCH_FINISH.
110h	383:0	AUTHOR_KEY_DIGEST	SHA-384 digest of the Author public key that certified the ID key, if provided in SNP_LAUNCH_FINSIH. Zeroes if AUTHOR_KEY__EN is 1.
140h	255:0	REPORT_ID	Report ID of this guest.

Byte Offset	Bits	Name	Description
160h	255:0	REPORT_ID_MA	Report ID of this guest's migration agent.
180h	63:0	REPORTED_TCB	Reported TCB version used to derive the VCEK that signed this report.
188h – 19Fh		-	Reserved.
1A0h-1DFh	511:0	CHIP_ID	Identifier unique to the chip.
1E0h-29Fh		-	Reserved.
2A0h-49Fh		SIGNATURE	Signature of this report. The format of the signature is described in Table 23.

Table 22. Structure of the PLATFORM_INFO Field

Byte Offset	Bits	Name	Description
0h	63:1	-	Reserved.
	0	SMT_EN	Indicates that SMT is enabled in the system.

Table 23. Format for an ECDSA P-384 with SHA-384 Signature (Used when SIGNATURE_ALGO is 102h)

Byte Offset	Bits	Name	Description
000h	575:0	R	R component of this signature
048h	575:0	S	S component of this signature
090h–1FFh		-	Reserved.

The firmware constructs an MSG_REPORT_RSP message containing the generated attestation report as defined in Table 24.

Table 24. MSG_REPORT_RSP Message Structure

Byte Offset	Bits	Name	Description
00h	31:0	STATUS	The status of key derivation operation. 0h: Success. 16h: Invalid parameters.
04h	31:0	REPORT_SIZE	Size in bytes of the report.
08h–1Fh		-	Reserved.
20h		REPORT	The attestation report generated by the firmware.

7.4 VM Export

When the hypervisor wishes to migrate a guest, it sends a request to that guest’s migration agent. The migration agent then sends the PSP a request message to export the guest’s data to the migration agent. The format of this request is defined in Table 25.

Table 25. MSG_EXPORT_REQ Message Structure

Byte Offset	Bits	Name	Description
00h	63:12	GCTX_PADDR	Bits 63:12 of the sPA of the guest context page for the guest to be exported
	11:0	-	Reserved. Must be zero.
08h	31:1	-	Reserved. Must be zero.
	0	IMI_EN	Indicates that an IMI is used to migrate the guest.
0Ch	31:0	-	Reserved. Must be zero.

The firmware checks that GCTX_PADDR is a valid sPA. If not, the firmware returns a status of INVALID_ADDRESS.

The firmware checks that GCTX_PADDR is a Context page. The firmware checks that GCTX.MA of the guest matches the sPA of the migration agent’s guest context page. If either check fails, the firmware returns a status of INVALID_GUEST.

The firmware checks that the guest to be exported is in the GSTATE_RUNNING state. If not, the firmware returns INVALID_GUEST_STATE.

The firmware responds with a MSG_EXPORT_RSP message containing the guest context defined in Table 26. The size of the payload is such that HDR_SIZE + MSG_SIZE is 4096. That is, the message fills a 4 KB page.

Table 26. MSG_EXPORT_RSP Message Structure

Byte Offset	Bits	Name	Description
00h	31:0	STATUS	The status of the attestation request. 0h: Success 16h: Invalid parameters
04h	31:0	GCTX_SIZE	Size in bytes of the guest context stored in GCTX
08h	31:0	GCTX_VERSION	Version of the GCTX field. Set to 1h for this ABI version.
0Ch–1Fh		-	Reserved.
20h–2AFh		GCTX	Guest context. See Table 27 for the format of this field.

Table 27. GCTX Field Structure

Byte Offset	Bits	Name	Description
000h	383:0	LD	See 4.1 for description of this field.
030h	255:0	OEK	
050h	255:0	VMPCK0	
070h	255:0	VMPCK1	
090h	255:0	VMPCK2	
0B0h	255:0	VMPCK3	
0D0h	255:0	VMRK	
0F0h	255:0	HostData	
110h	383:0	IDKeyDigest	
140h	383:0	AuthorKeyDigest	
170h	255:0	ReportID	
190h	383:0	IMD	
1C0h	31:0	MsgCount0	
1C4h	31:0	MsgCount1	
1C8h	31:0	MsgCount2	
1CCh	31:0	MsgCount3	
1D0h		RootMDEntry	See 5.1 for description of this field. If IMI_EN is set, then this field is set to 0h.
210h	61:2	-	Reserved. Must be zero.
	1	IDBlockEn	See 4.1 for description of this field.
	0	AuthorKeyEn	See 4.1 for description of this field.
218h	63:0	Policy	See 4.1 for description of this field.
220h	7:0	State	See 4.1 for description of this field.
228h	63:0	-	Reserved. Must be zero.
230h–28Fh		IDBlock	See 8.1 for the description of this field and Table 51 for the format of the field.

If IMI_EN message parameter is 0, the firmware makes the exported guest unable to run on this platform.

If IMI_EN message parameter is 1, the firmware allows the exported guest to continue running on this platform. The IMI within the guest is expected to make itself not runnable after it has completed migration.

If IMI_EN message parameter is 1, the firmware does not export the RootMDEntry. Instead, it writes 0h to the RootMDEntry field.

7.5 VM Import

When the hypervisor wishes to receive a migrated guest from another system, it first constructs a guest context with SNP_GCTX_CREATE. The hypervisor then passes the new guest context sPA to the migration agent. The migration agent then sends the PSP a request message to import the guest's data to the migration agent. The format of this request is defined in Table 28.

A hypervisor should ensure that all pages of the guest have been swapped out before invoking this command. The RootMDEntry in the guest contest should contain the root metadata entry of the guest that covers all pages of the guest.

Table 28. MSG_IMPORT_REQ Message Structure

Byte Offset	Bits	Name	Description
00h	63:12	GCTX_PADDR	Bits 63:12 of the sPA of a page donated to the firmware by the hypervisor to contain the guest context.
	11:0	-	Reserved. Must be zero.
08h	31:0	GCTX_SIZE	Size in bytes of the guest context stored in GCTX
0Ch	31:0	GCTX_VERSION	Version of the GCTX field. Set to 1h for this ABI version.
10h–1Fh		-	Reserved. Must be zero.
20h–2AFh		INCOMING_GCTX	Incoming guest context. See Table 27 for the format of this field.

The firmware checks that GCTX_PADDR is a valid sPA. If not, the firmware returns the status INVALID_ADDRESS. The firmware then checks that GCTX_PADDR is a Context page. If not, the firmware returns the status INVALID_GUEST.

The firmware checks that the guest is in the GSTATE_INIT state. If not, the firmware returns INVALID_GUEST_STATE.

The firmware checks that RootMDEntry of the incoming guest context has its VALID field set to 1. If not, the firmware returns INVALID_MDATA_ENTRY.

The firmware copies the incoming guest context into the context page at GCTX_PADDR. The firmware then sets the fields of the guest context page according to Table 29.

Table 29. Guest Context Initialized by the MSG_IMPORT_REQ Guest Message

Field	Value
MA	The GCTX_PADDR of the migration agent that sent this message.
IMIEn	0

The firmware transitions the guest to the GSTATE_RUNNING state.

The firmware responds with a message containing the status of the import. The response message is defined in Table 30.

Table 30. MSG_IMPORT_RSP Message Structure

Byte Offset	Bits	Name	Description
0h	31:0	STATUS	Status of the import operation
4h–Fh		-	Reserved.

7.6 VM Absorb

When an IMI is used to accelerate guest migration, a migration agent imports the new guest using the MSG_ABSORB_REQ message. This message requests that, after the hypervisor has launched the IMI, the firmware replace the guest's context with the context migrated from another machine.

The migration agent sends the firmware an MSG_ABSORB_REQ message as described in Table 31.

Table 31. MSG_ABSORB_REQ Message Structure

Byte Offset	Bits	Name	Description
00h	63:12	GCTX_PADDR	Bits 63:12 of the sPA of a page donated to the firmware by the hypervisor to contain the guest context.
	11:0	-	Reserved. Must be zero.
08h	31:0	IN_GCTX_SIZE	Size in bytes of the guest context stored in GCTX.
0Ch	31:0	IN_GCTX_VERSION	Version of the GCTX field. Set to 1h for this ABI version.
10h–1Fh		-	Reserved.
20h–28Fh		IN_GCTX	Incoming guest context. See Table 26 for the format of this field.

The firmware checks that GCTX_PADDR is a valid sPA. If not, the firmware returns the status INVALID_ADDRESS. The firmware then checks that GCTX_PADDR is a Context page. If not, the firmware returns the status INVALID_GUEST.

The firmware checks that the guest is in the GSTATE_LAUNCH state. The firmware also checks that GCTX.IMIEn is 1. If either check fails, the firmware returns the status INVALID_GUEST_STATE.

The firmware checks that the IN_GCTX.IMD is equal to GCTX.LD. If not, the firmware returns the status BAD_MEASUREMENT.

The firmware checks that it supports the IN_GCTX_VERSION and that the IN_GCTX_SIZE is compatible with this version. If not, the firmware returns the status INVALID_PARAM.

The firmware checks that RootMDEntry of the incoming guest context has its VALID field set to 0. If not, the firmware returns INVALID_MDATA_ENTRY.

The firmware overwrites the guest context at GCTX_PADDR with the guest context in the IN_GCTX field. The firmware then sets the state of the guest to the GSTATE_RUNNING state.

The firmware responds with a message containing the status of the import. The response message is defined in Table 32.

Table 32. MSG_ABSORB_RSP Message Structure

Byte Offset	Bits	Name	Description
0h	31:0	STATUS	Status of the absorb operation
4h–Fh		-	Reserved. Must be zero.

7.7 VMRK Message

During launch, the migration agent of the guest sends the VMRK to use for the guest. It must be encrypted with the migration agent’s VMPCK0. If not, the firmware returns INVALID_PARAM.

The structure of the VMRK message is defined in Table 33.

Table 33. Structure of the MSG_VMRK_REQ Guest Message

Byte Offset	Bits	Name	Description
0h	63:12	GCTX_PADDR	Bits 63:12 of the sPA of a page donated to the firmware by the hypervisor to contain the guest context.
	11:0	-	Reserved. Must be zero.
4h–1Fh		-	Reserved. Must be zero.

Byte Offset	Bits	Name	Description
20h	255:0	VMRK	A VMRK generated by a migration agent.

The firmware checks that GCTX_PADDR is a valid sPA. If not, the firmware returns the status INVALID_ADDRESS. The firmware then checks that GCTX_PADDR is a Context page. If not, the firmware returns the status INVALID_GUEST.

The firmware checks that the guest is in the GSTATE_LAUNCH state. The firmware also checks that GCTX.IMIEn is 0. If either check fails, the firmware returns the status INVALID_GUEST_STATE.

The firmware checks that GCTX.MA of the guest matches the GCTX_PADDR of the migration agent—that is, the guest sending the MSG_VMRK_REQ message. If not, the firmware returns the status INVALID_GUEST.

The firmware installs the VMRK into the guest's GCTX.VMRK.

The firmware responds with a message containing the status. The response message is defined in Table 34.

Table 34. MSG_VMRK_RSP Message Structure

Byte Offset	Bits	Name	Description
0h	31:0	STATUS	Status of the VMRK operation.
4h–Fh		-	Reserved.

Chapter 8 Command Reference

8.1 DOWNLOAD_FIRMWARE

This command allows the hypervisor to install new SNP firmware newer than the currently active firmware. This command is a legacy SEV command and documented in Section 5 of [SEV].

In addition to the checks performed in [SEV], the SNP platform state must be UNINIT. If not, the firmware returns INVALID_PLATFORM_STATE.

8.2 GET_ID

This command returns a unique ID for the system that can be used to obtain a certificate for the VCEK from AMD's Key Distribution Server. This command is a legacy SEV command and documented in Section 5 of [SEV].

In addition to the checks in [SEV], the firmware also checks that, if the SNP firmware state is INIT, the 16 B buffer pointed at by ID_PADDR resides entirely in Hypervisor or Default pages. Otherwise, the firmware returns INVALID_PAGE_STATE.

8.3 SNP_PLATFORM_STATUS

This command returns information about the current status and capabilities of the platform.

8.3.1 Parameters

Table 35. Layout of the CMDBUF_SNP_PLATFORM_STATUS Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	STATUS_PADDR	sPA to write the platform status structure. See Table 36.

8.3.2 Actions

The platform may be in any state when this command is called.

The firmware checks that STATUS_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS.

If the SNP firmware state is INIT, the page must be either a Firmware or Default page. If not, the firmware returns INVALID_PAGE_STATE.

If the platform state is UNINIT, the firmware does not check the state or size of the page.

The following data structure is written to memory at STATUS_PADDR

Table 36. Layout of the STRUCT_PLATFORM_STATUS Structure

Byte Offset	Bits	Name	Description
00h	7:0	API_MAJOR	Major API version.
01h	7:0	API_MINOR	Minor API version.
02h	7:0	STATE	The current platform state, zero extended. See 3.2 for encodings.
03h	7:0	-	Reserved.
04h	31:0	BUILD	Firmware build ID for this API version.
08h	31:0	-	Reserved.
0Ch	31:0	GUEST_COUNT	The number of guests currently managed by the firmware.
10h	63:0	PLATFORM_VERSION	The installed TCB version.
18h	63:0	REPORTED_TCB	The reported TCB version in guest attestation reports. Undefined if SNP is not in the INIT state.

8.3.3 Status Codes

Table 37. Status Codes for SNP_PLATFORM_STATUS

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	The address is invalid for use by the firmware.
INVALID_PARAM	MBZ fields are not zero.
INVALID_PAGE_STATE	The page at STATUS_PADDR is not in the correct RMP page state.

8.4 SNP_SET_REPORTED_TCB

This command sets the system wide ReportedTcb value. This value is used to derive the VCEK used to sign guest attestation reports.

8.4.1 Parameters

Table 38. Layout of the CMDBUF_SNP_PLATFORM_STATUS Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:0	In	REPORTED_TCB	The TCB version string to report in guest attestation reports.

8.4.2 Actions

The platform must be in the INIT state when this command is called.

The firmware checks that the REPORTED_TCB parameter is less than or equal to the installed TCB version. If not, the firmware returns INVALID_PARAM.

The firmware sets the system wide ReportedTcb value to REPORTED_TCB.

8.4.3 Status Codes

Table 39. Status Codes for SNP_PLATFORM_STATUS

Status	Condition
SUCCESS	Successful completion.
INVALID_PARAM	The desired reported TCB string is invalid
INVALID_PLATFORM_STATE	The platform is not in the INIT state

8.5 SNP_INIT

This command validates the platform configuration of the SNP and initializes the firmware.

8.5.1 Parameters

None.

8.5.2 Actions

Before invoking SNP_INIT, software must ensure that no CPUs contain dirty cache lines for the memory containing the RMP.

The firmware checks that the platform is in the UNINIT state. The firmware also checks that SEV-legacy firmware is not already initialized. If either check fails, the firmware returns INVALID_PLATFORM_STATE.

The firmware ensures the following system requirements are met:

- SYSCFG[MemoryEncryptionModEn] must be set to 1 across all cores (SEV must be enabled)
- SYSCFG[SecureNestedPagingEn] must be set to 1 across all cores
- SYSCFG[VMPLEn] must be set to 1 across all cores.
- VM_HSAVE_PA (MSR C001_0117) must be set to 0h across all cores

If any of the above checks fail, the firmware returns INVALID_CONFIG.

The firmware also ensures that the following requirements for the RMP have been met:

- RMP_BASE and RMP_END must be set identically across all cores
- RMP_BASE must be 1 MB aligned
- RMP_END – RMP_BASE + 1 must be a multiple of 1 MB
- RMP is large enough to protect itself

If any of the above checks fail, the firmware returns INVALID_ADDRESS.

The firmware alters the RMP such that:

- Pages of the RMP are in the Firmware state
- All other pages covered by the RMP are in the Hypervisor state

The firmware also initializes any microarchitectural data structures within the RMP. Immediately after completing RMP initialization, the firmware forces a TLB flush across all cores on all sockets.

The firmware marks all encryption capable ASIDs as unusable for encrypted virtualization.

The firmware sets the system wide ReportedTcb value to the installed TCB version. To change this value, SNP_SET_REPORTED_TCB must be invoked after SNP_INIT completes successfully.

The firmware sets the platform state to INIT.

8.5.3 Status Codes

Table 40. Status Codes for SNP_INIT

Status	Condition
SUCCESS	Successful completion.
INVALID_CONFIG	The system is not in a valid configuration that can support SNP.
INVALID_PLATFORM_STATE	The platform is not in the UNINIT state.
INVALID_ADDRESS	RMP_BASE or RMP_END are not valid addresses

8.6 SNP_GCTX_CREATE

This command donates a page from the hypervisor to the firmware to be used to store the guest context.

8.6.1 Parameters

Table 41. Layout of the CMDBUF_SNP_GCTX_CREATE Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	GCTX_PADDR	Bits 63:12 of the sPA of a page donated to the firmware by the hypervisor to contain the guest context.
	11:0	-	-	Reserved. Must be zero.

8.6.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that GCTX_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS.

The firmware checks that the donated context page is in the Firmware state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the donated page is marked as a 4 KB page in the RMP. If not, the firmware returns INVALID_PAGE_SIZE.

The firmware transitions the page to the Context state and initializes the guest context according to Table 42. All other fields within the guest context remain indeterminate until they are initialized through the launch process or through the import process.

Table 42. Guest Context Initialized by the SNP_GCTX_CREATE Command

Field	Value
ASID	Set to 0h indicating that no ASID has been associated with this guest.
State	GSTATE_INIT.
VEK	Generated using a CSRNG.

8.6.3 Status Codes

Table 43. Status Codes for SNP_GCTX_CREATE

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_ADDRESS	The address is invalid for use by the firmware.
INVALID_PARAM	MBZ fields are not zero.
INVALID_PAGE_STATE	The page is not in the Firmware state.
INVALID_PAGE_SIZE	The page is not a 4 KB page.

8.7 SNP_ACTIVATE

This command installs the guest's VEK into the memory controller in the key slot associated with a given ASID.

8.7.1 Parameters

Table 44. Layout of the CMDBUF_SNP_ACTIVATE Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	GCTX_PADDR	Bits 63:12 of the sPA of a page donated to the firmware by the hypervisor to contain the guest context.
	11:0	-	-	Reserved. Must be zero.
08h	31:0	In	ASID	ASID to bind to the guest.

8.7.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that GCTX_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS. The firmware then checks that the page at GCTX_PADDR is in the Context state. If not, the firmware returns INVALID_GUEST.

The firmware checks that the guest is in the GSTATE_LAUNCH state or in the GSTATE_RUNNING state. If not, the firmware returns INVALID_GUEST_STATE.

The firmware checks that ASID is an encryption capable ASID. If not, the firmware returns INVALID_ASID. If the ASID is already assigned to another guest, the firmware returns ASID_OWNED. If the guest is already activated, the firmware returns ACTIVE.

The firmware checks that a DF_FLUSH is not required. If a DF_FLUSH is required, the firmware returns DFFLUSH_REQUIRED. Note that all ASIDs are marked to require a DF_FLUSH at reset.

The firmware checks that there are no pages assigned to the ASID in the RMP. If not, the firmware returns INVALID_CONFIG.

The firmware installs the guest's VEK into the memory controllers in the key slot associated with the given ASID.

8.7.3 Status Codes

Table 45. Status Codes for SNP_ACTIVATE

Status	Condition
SUCCESS	Successful completion.
INVALID_CONFIG	ASID has pages assigned to it already.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_GUEST_STATE	The guest is not in the LAUNCH state.
INVALID_ADDRESS	The address is invalid for use by the firmware.
INVALID_PARAM	MBZ fields are not zero.
INVALID_GUEST	The guest is invalid.
INVALID_ASID	The provided ASID is not an encryption capable ASID.
ASID_OWNED	The ASID is already owned by another guest.
ACTIVE	The guest is already activated.
DFFLUSH_REQUIRED	DF_FLUSH was not invoked before this command.

8.8 SNP_ACTIVATE_EX

This command installs the guest's VEK into the memory controller in the key slot associated with a given ASID on select core complexes. Only hardware threads in the selected core complex may execute the guest. When an ASID is later re-used, WBINVD need be done only on core complexes associated with the guest.

8.8.1 Parameters

Table 46. Layout of the CMDBUF_SNP_ACTIVATE_EX Structure

Byte Offset	Bits	In/Out	Name	Description
00h	31:0	In	EX_LEN	Length of command buffer. 20h for this version.
04h	31:0	-	-	Reserved.
08h	63:12	In	GCTX_PADDR	Bits 63:12 of the sPA of a page donated to the firmware by the hypervisor to contain the guest context.
	11:0	-	-	Reserved. Must be zero.
10h	31:0	In	ASID	The ASID in which the guest should be bound.
14h	31:0	In	NUMIDs	Number of APIC IDs in the ID_PADDR list.
18h	63:0	In	ID_PADDR	Bits 63:0 of the sPA of a list of 32-bit APIC IDs.

8.8.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that GCTX_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS. The firmware then checks that the page at GCTX_PADDR is in the Context state. If not, the firmware returns INVALID_GUEST.

The firmware checks that the guest is in the GSTATE_LAUNCH state or in the GSTATE_RUNNING state. If not, the firmware returns INVALID_GUEST_STATE.

The firmware checks that ASID is an encryption capable ASID. If not, the firmware returns INVALID_ASID. If the ASID is already assigned to another guest, the firmware returns

ASID_OWNED. If the guest is already activated but on a different ASID, the firmware returns ACTIVE.

The firmware checks that a DF_FLUSH is not required. If so, the firmware returns DFFLUSH_REQUIRED. Note that all ASIDs are marked to require a DF_FLUSH at reset.

If the guest is not yet activated, the firmware checks that there are no pages assigned to the ASID in the RMP. If not, the firmware returns INVALID_CONFIG.

The firmware installs the guest's VEK into the memory controllers for the given APIC IDs into the key slot associated with the given ASID. This command can be called multiple times in order to expand the set of CCXs on which the guest may execute.

8.8.3 Status Codes

Table 47. Status Codes for SNP_ACTIVATE_EX

Status	Condition
SUCCESS	Successful completion.

8.9 SNP_DECOMMISSION

This command destroys a guest context. After this command successfully completes, the guest will not long be runnable.

8.9.1 Parameters

Table 48. Layout of the CMDBUF_SNP_DECOMMISSION Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	GCTX_PADDR	Bits 63:12 of the sPA of the guest's context page.
	11:0	-	-	Reserved. Must be zero.

8.9.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that the GCTX_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS.

The firmware checks that the page is a Context page. If not, the firmware returns INVALID_GUEST.

The firmware marks the ASID of the guest as not runnable. Then, the firmware records that each CPU core on each of the CCXs that the guest was activated on requires a WBINVD followed by a single DF_FLUSH command to ensure that all unencrypted data in the caches are invalidated before reusing the ASID. The firmware then transitions the page into a Firmware page.

8.9.3 Status Codes

Table 49. Status Codes for SNP_DECOMMISSION

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_ADDRESS	The address is not valid or is misaligned.
INVALID_PARAM	MBZ fields are not zero.
INVALID_GUEST	The guest is not valid.

8.10 SNP_DF_FLUSH

This command flushes SOC data buffers after CPU caches have been invalidated. After a VM is decommissioned or exported, the hypervisor must execute a WBINVD on the cores that the previous guest was active on before invoking the SNP_DF_FLUSH command. The combination of WBINVD and SNP_DF_FLUSH ensures that all data associated with the previous guest is no longer in any CPU caches.

8.10.1 Parameters

None

8.10.2 Actions

For each core marked for cache invalidation, the firmware checks that the core has executed a WBINVD instruction. If not, the firmware returns WBINVD_REQUIRED. The commands that mark cores for cache invalidation include SNP_DECOMMISSION and the guest request MSG_EXPORT_REQ.

The firmware flushes the write buffers of the data fabric and records that a flush has been performed for all decommissioned ASIDs.

8.10.3 Status Codes

Table 50. Status Codes for SNP_DF_FLUSH

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The firmware is not in the INIT.
WBINVD_REQUIRED	At least one core did not execute a WBINVD instruction before calling this command.

8.11 SNP_SHUTDOWN

This command returns the firmware to an uninitialized state.

8.11.1 Parameters

None

8.11.2 Actions

If the SNP firmware is in the UNINIT state, the firmware returns SUCCESS without taking any further action.

If SEV firmware is not in the UNINIT state, the firmware returns INVALID_PLATFORM_STATE.

If the SNP firmware is in the INIT state, the firmware checks for every encryption capable ASID that the ASID is not in use by a guest and a DF_FLUSH is not required. If a DF_FLUSH is required, the firmware returns DFFLUSH_REQUIRED. The firmware then clears the encryption keys out of the memory controller. Finally, the firmware transitions the platform to the UNINIT state and returns SUCCESS.

Note that the firmware will not automatically reclaim any pages marked as immutable in the RMP. The hypervisor should either reclaim the pages using SNP_PAGE_RECLAIM or should call SNP_INIT afterwards to reset the RMP.

8.11.3 Status Codes

Table 51. Status Codes for SNP_SHUTDOWN

Status	Condition
SUCCESS	Successful completion.

8.12 SNP_LAUNCH_START

This command initializes the flow to launch a guest.

8.12.1 Parameters

Table 52. Layout of the CMDBUF_SNP_LAUNCH_START Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	GCTX_PADDR	Bits 63:12 of the sPA of the guest context page.
	11:0	-	-	Reserved. Must be zero.
08h	63:0	In	POLICY	Guest policy. See Table 7 for a description of the guest policy structure.
10h	63:12	In	MA_GCTX_PADDR	Bits 63:12 of the sPA of the guest context of the migration agent. Ignored if MA_EN is 0.
	11:0	In	-	Reserved. Must be zero.
18h	31:2	-	-	Reserved. Must be zero.
	1	In	IMI_EN	Indicates that this launch flow is launching an IMI for the purpose of guest-assisted migration.
	0	In	MA_EN	1 if this guest is associated with a migration agent. Otherwise 0.

8.12.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that GCTX_PADDR is a valid sPA. If MA_EN is 1, the firmware checks that MA_GCTX_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS. The firmware then checks that GCTX_PADDR is a Context page. If MA_EN is 1, the firmware checks that MA_GCTX_PADDR is a Context pages. If not, the firmware returns INVALID_GUEST.

The firmware checks that the guest is in the GSTATE_INIT state. If not, the firmware returns INVALID_GUEST_STATE.

The firmware checks that the guest's policy is satisfied by checking that the following conditions are met:

- If MA_EN is 1, POLICY.MIGRATE_MA must be 1.

- If MA_EN is 1, then the migration agent must not be migratable—that is, the migration agent itself must not be bound to another migration agent.
- If POLICY.VMPL is 1, then VMPLs must be enabled.
- If POLICY.SMT is 0, then SMT must be disabled.
- POLICY.ABI_MAJOR must be equal the major version of this ABI.
- POLICY.ABI_MINOR must be less than or equal to the minor version of this ABI.

If any of the above conditions are not met, the firmware returns POLICY_FAILURE.

The firmware initializes the guest context with the values defined in Table 53.

Table 53. Guest Context Field Initialization for the Launch Flow

Field	Value
MsgCount0 MsgCount1 MsgCount2 MsgCount3	0h
Policy	Set to POLICY.
MA	Set to MA_GCTX_PADDR if MA_EN is 1. Set to PADDR_INVAL otherwise.
OEK	Generated using a CSRNG.
VMPCK0 VMPCK1 VMPCK2 VMPCK3	Generated using a CSRNG.
VMRK	Generated using a CSRNG. May be replaced by a VMRK guest message from the associated migration agent. See 7.7.
LD	0h
IMD	0h
IDBlockEn	0
IDBlock	0h
IDKeyDigest	0h
AuthorKeyEn	0
AuthorKeyDigest	0h
ReportID	Generated using a CSRNG.
IMIEn	Set to IMI_EN.

The firmware sets the guest state to GSTATE_LAUNCH.

8.12.3 Status Codes

Table 54. Status Codes for SNP_LAUNCH_START

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_ADDRESS	An address was not a valid sPA or properly aligned.
INVALID_PARAM	MBZ fields are not zero.
INVALID_GUEST	The guest is invalid.
INVALID_GUEST_STATE	The guest was not in the GSTATE_INIT state.
POLICY_FAILURE	The guest’s policy was violated.

8.13 SNP_LAUNCH_UPDATE

This command inserts pages into the guest physical address space.

8.13.1 Parameters

Table 55. Layout of the CMDBUF_SNP_LAUNCH_UPDATE Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	GCTX_PADDR	Bits 63:12 of the sPA of the guest context page.
	11:0	-	-	Reserved. Must be zero.
08h	31:5	-	-	Reserved. Must be zero.
	4	In	IMI_PAGE	Indicates that this page is part of the IMI of the guest.
	3:1	In	PAGE_TYPE	Encoded page type. See Table 56.
	0	In	PAGE_SIZE	Indicates page size. 0 indicates a 4 KB page. 1 indicates a 2 MB page.
0Ch	31:0	-	-	Reserved. Must be zero.
10h	63:12	In	PAGE_PADDR	Bits 63:12 of the sPA of the destination page. The page size is determined by PAGE_SIZE.
	11:0	-	-	Reserved. Must be zero.
18h	63:32	-	-	Reserved. Must be zero.
	31:24	In	VMPL3_PERMS	VMPL permission mask for VMPL3. See Table 57 for the definition of the mask.
	23:16	In	VMPL2_PERMS	VMPL permission mask for VMPL1. See Table 57 for the definition of the mask.
	15:8	In	VMPL1_PERMS	VMPL permission mask for VMPL1. See Table 57 for the definition of the mask.
	7:0	-	-	Reserved. Must be zero.

Table 56. Encodings for the PAGE_TYPE Field

Value	Name	Description
00h	-	Reserved.
01h	PAGE_TYPE_NORMAL	A normal data page.
02h	PAGE_TYPE_VMSA	A VMSA page.
03h	PAGE_TYPE_ZERO	A page full of zeroes.
04h	PAGE_TYPE_UNMEASURED	A page that is encrypted but not measured.

Value	Name	Description
05h	PAGE_TYPE_SECRETS	A page for the firmware to store secrets for the guest.
06h	PAGE_TYPE_CPUID	A page for the hypervisor to provide CPUID function values.
All other encodings		Reserved.

Table 57. VMPL Permission Mask

Bit	Field	Description
7:4	-	Reserved. Must be zero.
3	Execute-Supervisor	Page is executable by the VMPL in CPL2, CPL1, and CPL0.
2	Execute-User	Page is executable by the VMPL in CPL3.
1	Write	Page is writeable by the VMPL.
0	Read	Page is readable by the VMPL.

8.13.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that GCTX_PADDR and PAGE_PADDR are valid sPAs. If not, the firmware returns INVALID_ADDRESS. The firmware checks that if PAGE_SIZE is 1, then PAGE_PADDR is 2 MB aligned. If this check fails, the firmware returns INVALID_ADDRESS.

The firmware checks that GCTX_PADDR is a Context page. If not, the firmware returns INVALID_GUEST.

The firmware checks that the guest is in the GSTATE_LAUNCH state. If not, the firmware returns INVALID_GUEST_STATE.

The firmware also checks that the page at PAGE_PADDR is Pre-Guest page. If not, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the guest is activated—that is, it has an assigned ASID. If not, the firmware returns INACTIVE.

The firmware checks that the ASID of the destination page indicated by the RMP matches the ASID of the guest. If not, the firmware returns INVALID_PAGE_OWNER.

The firmware checks that the destination page size indicated by the RMP matches the page size indicated by the PAGE_SIZE parameter. If not, the firmware returns INVALID_PAGE_SIZE.

The firmware checks that if GCTX.IMIEn is 1, then IMI_PAGE is also 1. If not, then the firmware returns INVALID_PARAM.

The firmware checks that if VMPLs are not enabled, then VMPL1_PERMS, VMPL2_PERMS, and VMPL3_PERMS must be zero. If not, the firmware returns INVALID_PARAM.

The firmware updates the GCTX.LD and possibly the GCTX.IMD with information describing the contents and location of the pages inserted into the guest. Each update to the digest is of the following form:

DIGEST_NEW := SHA-384(PAGE_INFO)

where PAGE_INFO is the structure defined in Table 58.

Table 58. Layout of the PAGE_INFO Structure

Byte Offset	Bits	Field	Description
0h	383:0	DIGEST_CUR	The value of the current digest (either LD or IMD).
30h	383:0	CONTENTS	The SHA-384 digest of the measured contents of the region, if any. See the following subsections.
60h	15:0	LENGTH	Length of this structure in bytes.
62h	7:0	PAGE_TYPE	The zero-extended PAGE_TYPE field provided by the hypervisor.
63h	7:1	-	0h
	0	IMI_PAGE	Set to the IMI_PAGE flag provided by the hypervisor.
64h	31:24	VMPL3_PERMS	The VMPL3_PERMS field provided by the hypervisor.
	23:16	VMPL2_PERMS	The VMPL2_PERMS field provided by the hypervisor.
	15:8	VMPL1_PERMS	The VMPL1_PERMS field provided by the hypervisor.
	7:0	-	0Fh
68h	63:0	GPA	The 64-bit gPA of the region.

The firmware unconditionally updates GCTX.LD. If IMI_PAGE is 1, the firmware updates the GCTX.IMD.

The following subsections describe how the PAGE_TYPE, GPA, and CONTENTS fields are determined.

The following subsections describes the actions the firmware takes on the guest address space depending on the page type, PAGE_TYPE. If the page size is 2 MB, then the firmware will update the launch digest as if the data were provided in a contiguous sequence of 4 KB pages. The final launch digest is therefore independent of how the hypervisor chooses to size the pages within the nested page tables and in the RMP.

8.13.2.1 PAGE_TYPE_NORMAL

The firmware performs the actions in this subsection when PAGE_TYPE is PAGE_TYPE_NORMAL.

For each 4 KB chunk within the page, the firmware constructs a PAGE_INFO structure with the following data:

- **PAGE_TYPE:** PAGE_TYPE_NORMAL
- **GPA:** The gPA of the 4 KB chunk. The firmware calculates this by adding the offset of the chunk to RMP.GPA of the page.
- **CONTENTS:** The SHA-384 digest of the contents of the 4 KB chunk

The firmware updates GCTX.LD and GCTX.IMD as described above.

The firmware encrypts the page with the VEK in place. The firmware then sets the VMPL permissions for the page and transitions the destination page to Guest-Valid.

8.13.2.2 PAGE_TYPE_VMSA

The firmware performs the actions in this subsection when PAGE_TYPE is PAGE_TYPE_VMSA.

The firmware checks that the destination page is 4 KB. If not, the firmware returns INVALID_PAGE_SIZE.

The firmware constructs a PAGE_INFO structure with the following data:

- **PAGE_TYPE:** PAGE_TYPE_VMSA
- **GPA:** The gPA of the 4 KB page. The firmware uses the RMP.GPA of the page.
- **CONTENTS:** The SHA-384 digest of the contents of the 4 KB page

The firmware updates GCTX.LD and GCTX.IMD as described above.

The firmware encrypts the page with the VEK in place. The firmware sets the RMP.VMSA of the page to 1. The firmware sets the VMPL permissions for the page and transitions the page to Guest-Valid.

8.13.2.3 PAGE_TYPE_ZERO

The firmware performs the actions in this subsection when PAGE_TYPE is PAGE_TYPE_ZERO.

For each 4 KB chunk within the page, the firmware constructs a PAGE_INFO structure with the following data:

- **PAGE_TYPE:** PAGE_TYPE_ZERO
- **GPA:** The gPA of the 4 KB chunk. The firmware calculates this by adding the offset of the chunk to RMP.GPA of the page.
- **CONTENTS:** 0h.

The firmware updates GCTX.LD and GCTX.IMD as described above.

The firmware encrypts a page of zeroes with the VEK. The firmware sets the VMPL permissions for the page and transitions the page to Guest-Valid.

8.13.2.4 PAGE_TYPE_UNMEASURED

The firmware performs the actions in this subsection when PAGE_TYPE is PAGE_TYPE_UNMEASURED.

For each 4 KB chunk within the page, the firmware constructs a PAGE_INFO structure with the following data:

- **PAGE_TYPE:** PAGE_TYPE_UNMEASURED
- **GPA:** The gPA of the 4 KB chunk. The firmware calculates this by adding the offset of the chunk to RMP.GPA of the page.
- **CONTENTS:** 0h.

The firmware updates GCTX.LD and GCTX.IMD as described above.

The firmware encrypts the page with the VEK in place. The firmware sets the VMPL permissions for the page and transitions the page to Guest-Valid.

8.13.2.5 PAGE_TYPE_SECRETS

The firmware performs the actions in this subsection when PAGE_TYPE is PAGE_TYPE_SECRETS.

The firmware checks that the destination page is 4 KB. If not, the firmware returns INVALID_PAGE_SIZE.

The firmware constructs a PAGE_INFO structure with the following data:

- **PAGE_TYPE:** PAGE_TYPE_SECRETS
- **GPA:** The gPA of the 4 KB page. The firmware uses the RMP.GPA of the page.
- **CONTENTS:** 0h.

The firmware updates GCTX.LD and GCTX.IMD as described above.

The firmware constructs the 4 KB data structure described in Table 56. Reserved fields are set to 0h. The firmware then encrypts the data structure with the guest's VEK and writes it into the page. The firmware ensures that the data structure content remains confidential to the guest and the firmware.

Table 59. Secrets Page Format

Byte Offset	Bits	Name	Description
000h	31:0	VERSION	Version of the secrets page format. The version described in this specification is 1h.
004h	31:1	-	Reserved.
	0	IMI_EN	Set to the value of GCTX.IMIEn.
008h–01Fh		-	Reserved.
020h	255:0	VMPCK0	Set to GCTX.VMPCK0.
040h	255:0	VMPCK1	Set to GCTX.VMPCK1.
060h	255:0	VMPCK2	Set to GCTX.VMPCK2.
080h	255:0	VMPCK3	Set to GCTX.VMPCK3.
0A0h–FFFh		-	Reserved.

The firmware sets the VMPL permissions for the page and transitions the page to Guest-Valid.

8.13.2.6 PAGE_TYPE_CPUID

The firmware performs the actions in this subsection when PAGE_TYPE is PAGE_TYPE_CPUID.

The firmware checks that the destination page is 4 KB. If not, the firmware returns INVALID_PAGE_SIZE.

The firmware constructs a PAGE_INFO structure with the following data:

- **PAGE_TYPE:** PAGE_TYPE_CPUID
- **GPA:** The gPA of the 4 KB page. The firmware uses the RMP.GPA of the page.
- **CONTENTS:** 0h.

The firmware updates GCTX.LD and GCTX.IMD as described above.

The hypervisor should fill the page with CPUID functions as described in Table 59.

The firmware will alter the function structure only when the hypervisor promised a feature that is not truly available. However, the firmware will never indicate features exists that the hypervisor did not report—even if those features are present. This allows the hypervisor to advertise a subset of the features of the platform.

If the firmware encounters a CPUID function that is not defined in [PPR], the firmware sets the EAX_IN, ECX_IN, XCRO_IN, and XSS_IN fields of the function to 0xFFFFFFFF. This indicates to the guest that the CPUID function is not known to the firmware and thus the firmware is unable to confirm the function's value.

The page has enough for COUNT_MAX function structures, but only COUNT function structures are valid. COUNT_MAX is 64.

The firmware then encrypts the page with the VEK in place.

Table 60. CPUID Page Format

Byte Offset	Bits	Name	Description
00h	31:0	COUNT	Number of CPUID functions to validate. Must be less than COUNT_MAX.
04h	31:0	-	Reserved. Must be zero.
08h	63:0	-	Reserved. Must be zero.
10h–C0Fh		CPUID_FUNCTION[]	COUNT_MAX number of CPUID_FUNCTION records (See 7.1 for the format of this record). Only the first COUNT records are valid.

The firmware sets the VMPL permissions for the page and transitions the page to Guest-Valid.

8.13.3 Status Codes

Table 61. Status Codes for SNP_LAUNCH_UPDATE

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_ADDRESS	An address is invalid or incorrectly aligned.
INVALID_PARAM	MBZ fields are not zero.
INVALID_GUEST	The guest is invalid.
INVALID_GUEST_STATE	The guest is not in the GSTATE_LAUNCH state.
INACTIVE	The guest has not been activated.
INVALID_PAGE_STATE	A page was not in the correct state.
INVALID_PAGE_OWNER	The destination page was not owned by the guest.
INVALID_PAGE_SIZE	The destination page was not the correct size.
INVALID_PARAM	IMI_PAGE was incorrectly set.

8.14 SNP_LAUNCH_FINISH

This command completes the guest launch flow.

8.14.1 Parameters

Table 62. Layout of the CMDBUF_SNP_LAUNCH_FINISH Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	GCTX_PADDR	Bits 63:12 of the sPA of the guest context page.
	11:0	-	-	Reserved. Must be zero.
08h	63:0	In	ID_BLOCK_PADDR	sPA of the ID block. Ignored if ID_BLOCK_EN is 0.
10h	63:0	In	ID_AUTH_PADDR	sPA of the authentication information of the ID block. Ignored if ID_BLOCK_EN is 0.
18h	63:2	-	-	Reserved. Must be zero.
	1	In	AUTH_KEY_EN	Indicates that the author key is present in the ID authentication information structure. Ignored if ID_BLOCK_EN is 0.
	0	In	ID_BLOCK_EN	Indicates that the ID block is present.
20h	255:0	In	HOST_DATA	Opaque host-supplied data to describe the guest. The firmware does not interpret this value.

Table 63. Structure of the ID Block

Byte Offset	Bits	Name	Description
0h	383:0	LD	The expected launch digest of the guest.
30h	127:0	FAMILY_ID	Family ID of the guest, provided by the guest owner and uninterpreted by the firmware.
40h	127:0	IMAGE_ID	Image ID of the guest, provided by the guest owner and uninterpreted by the firmware.
50h	31:0	VERSION	Version of the ID block format. Must be 1h for this version of the ABI.
54h	31:0	GUEST_SVN	SVN of the guest.
58h	63:0	POLICY	The policy of the guest.

Table 64. Structure of the ID Authentication Information Structure

Byte Offset	Bits	Name	Description
0h	31:0	ID_KEY_ALGO	The algorithm of the ID Key. 1h indicates ECDSA P-384 with SHA-384. All other encodings are reserved.
4h	31:0	AUTH_KEY_ALGO	The algorithm of the Author Key. 1h indicates ECDSA P-384 with SHA-384. All other encodings are reserved. Ignored if AUTH_KEY_EN is 0.
8h–3Fh		-	Reserved. Should be zero.
40h–23Fh		ID_BLOCK_SIG	The signature of the ID block. See Table 65 for the format of the signature.
240h–643h		ID_KEY	The public component of the ID key. See Table 63 for the format of the key.
644h–67Fh		-	Reserved. Should be zero.
680h–87Fh		ID_KEY_SIG	The signature of the ID_KEY. See Table 65 for the format of the signature.
880h–C83h		AUTHOR_KEY	The public component of the Author key. See Table 66 for the format of the key. Ignored if AUTHOR_KEY_EN is 0.
C84h–FFFh		-	Reserved. Should be zero.

Table 65. Format for an ECDSA P-384 Signature

Byte Offset	Bits	Name	Description
000h	575:0	R	R component of this signature.
048h	575:0	S	S component of this signature.
090h–1FFh		-	Reserved. Must be zero.

Table 66. Format for an ECDSA P-384 Public Key

Byte Offset	Bits	Name	Description
000h	31:0	CURVE	Curve ID. 2h indicates P-384. All other encodings are reserved.
004h	575:0	QX	R component of this signature.
04Ch	575:0	QY	S component of this signature.

Byte Offset	Bits	Name	Description
094h–403h		-	Reserved. Must be zero.

8.14.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that GCTX_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS. The firmware then checks that GCTX_PADDR is a Context page. If not, the firmware returns INVALID_GUEST.

The firmware checks that the guest is in the GSTATE_LAUNCH state. The firmware also checks that GCTX.IMIEn is 0. If either check fails, the firmware returns INVALID_GUEST_STATE.

The firmware checks that the guest is activated—that is, it has an assigned ASID. If not, the firmware returns INACTIVE.

The firmware checks that, if ID_BLOCK_EN is 1, then ID_BLOCK_PADDR and ID_AUTH_PADDR are valid sPAs. If not, the firmware returns INVALID_ADDRESS.

If ID_BLOCK_EN is 1, the firmware checks that the LD field of the ID block is equal to GCTX.LD. If not, the firmware returns BAD_MEASUREMENT. The firmware then checks that the POLICY field of the ID block is equal to GCTX.Policy. If not, the firmware returns POLICY_FAILURE. The firmware then validates the signature of the ID block using the ID public key. If AUTH_KEY_EN is also 1, the firmware validates the signature of the ID key using the Author public key. If either signature fails to validate, the firmware returns BAD_SIGNATURE.

The firmware then initializes the guest context fields according to Table 67.

Table 67. Guest Context Fields Initialized During SNP_LAUNCH_FINISH

Field	Value
HostData	HOST_DATA.
IDBlockEn	ID_BLOCK_EN.
IDBlock	If ID_BLOCK_EN is 1, then set to the ID block. 0 otherwise.
IDKeyDigest	If ID_BLOCK_EN is 1, then set to the SHA-384 digest of the ID public key. 0 otherwise.
AuthorKeyEn	AUTHOR_KEY_EN.
AuthorKeyDigest	If AUTHOR_KEY_EN is 1, then set to the SHA-384 digest of the Author public key. 0 otherwise.

The firmware makes the guest runnable on the ASID it is activated on. The firmware then sets the guest state to GSTATE_RUNNING.

8.14.3 Status Codes

Table 68. Status Codes for SNP_LAUNCH_FINISH

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_GUEST_STATE	The guest is not in the GSTATE_LAUNCH state or GCTX.IMIEn is not 0.
INVALID_GUEST	The guest is invalid.
INVALID_ADDRESS	An address is invalid or incorrectly aligned.
INVALID_PARAM	MBZ fields are not zero.
INVALID_PAGE_STATE	A page was not in the correct state.
INACTIVE	The guest has not been activated.
BAD_SIGNATURE	Incorrect signature provided.

8.15 SNP_GUEST_STATUS

This command is used to retrieve information about an SNP guest.

8.15.1 Parameters

Table 69. Layout of the CMDBUF_SNP_GUEST_STATUS Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	GCTX_PADDR	Bits 63:12 of the sPA of the guest context page.
	11:0	-	-	Reserved. Must be zero.
08h	63:0	In	STATUS_PADDR	Bits 63:0 of the sPA of the guest status structure. See Table 70.

8.15.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that the GCTX_PADDR and STATUS_PADDR are valid sPAs. If either check fails, the firmware returns INVALID_ADDRESS.

The firmware checks that the guest context page is a Context page. If not, the firmware returns INVALID_GUEST. The firmware checks that the guest status page is a Firmware or Default page. If not, the firmware returns INVALID_PAGE_STATE.

The firmware writes the following structure to the beginning of the guest status page.

Table 70. Layout of the STRUCT_SNP_GUEST_STATUS Structure

Byte Offset	Bits	Name	Description
00h	63:0	POLICY	Guest policy.
08h	31:0	ASID	Current ASID. If none is assigned, set to 0h.
0Ch	7:0	STATE	Current guest state.
0Dh	7:0	-	Reserved.
0Eh	15:0	-	Reserved.
10h	63:0	-	Reserved.
18h	63:0	-	Reserved.

8.15.3 Status Codes

Table 71. Status Codes for SNP_GUEST_STATUS

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_ADDRESS	The address is invalid for use by the firmware.
INVALID_PARAM	MBZ fields are not zero.
INVALID_GUEST	The guest context page was invalid.
INVALID_PAGE_STATE	The guest status page was not in the correct state.
INVALID_PAGE_SIZE	The guest status page was not the correct size.

8.16 SNP_PAGE_MOVE

This command moves the contents of SNP-protected pages within the system physical address space without violating SNP security.

8.16.1 Parameters

Table 72. Layout of the CMDBUF_SNP_PAGE_MOVE Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	GCTX_PADDR	Bits 63:12 of the sPA of the guest context page.
	11:0	-	-	Reserved. Must be zero.
08h	31:1	-	-	Reserved. Must be zero.
	0	In	PAGE_SIZE	Indicates page size. 0 indicates a 4 KB page. 1 indicates a 2 MB page.
0Ch	31:0	-	-	Reserved. Must be zero.
10h	63:12	In	SRC_PADDR	Bits 63:12 of the sPA of the source page. The page size is determined by PAGE_SIZE.
	11:0	-	-	Reserved. Must be zero.
18h	63:12	In	DST_PADDR	Bits 63:12 of the sPA of the destination page. The page size is determined by PAGE_SIZE.
	11:0	-	-	Reserved. Must be zero.

8.16.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that GCTX_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS. The firmware then checks that the page at GCTX_PADDR is in the Context state. If not, the firmware returns INVALID_GUEST.

The firmware checks that the guest is in the GSTATE_LAUNCH or GSTATE_RUNNING states. If not, the firmware returns INVALID_GUEST_STATE. The firmware then checks that the guest is activated. If not, the firmware returns INACTIVE.

The firmware checks that SRC_PADDR and DST_PADDR are valid sPAs. If not, the firmware returns INVALD_ADDRESS.

The firmware checks that the source and destination page sizes indicated by the RMP match the page size indicated by the PAGE_SIZE parameter. If not, the firmware returns INVALID_PAGE_SIZE.

This command operates either on guest pages or on Metadata pages. The following subsections describe each case.

8.16.2.1 Guest Pages

The firmware performs the actions in this section when the source page is a Pre-Swap or Pre-Guest page.

The firmware checks that the destination page is a Pre-Guest page. If either check fails, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the RMP.ASID of both the source and destination pages are equal to the ASID of the guest. If not, the firmware returns INVALID_PAGE_OWNER.

The firmware uses the guest’s VEK to copy the plaintext of the source page into the plaintext of the destination page.

The firmware sets the RMP.GPA and RMP.VMSA of the destination page to match the RMP.GPA and RMP.VMSA of the source page. If VMPLs are enabled, the firmware also sets the VMPL permissions bits of the destination page to match the VMPL permission bits of the source page.

If the source page is a Pre-Guest page, the firmware transitions the destination page into a Guest-Invalid page. If the source page is a Pre-Swap page, the firmware transitions the destination page into a Guest-Valid page. Finally, the firmware transitions the source page into a Guest-Invalid page.

8.16.2.2 Metadata Pages

The firmware performs the actions in this section when the source page is a Metadata page.

The firmware checks that the destination page is a Firmware page. If either check fails, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the RMP.GPA of the source page is equal to the sPA of the guest context. If not, the firmware returns INVALID_PAGE_OWNER.

The firmware copies the contents of the source page into the destination page. The firmware then sets the RMP.GPA of the destination to match the sPA of the guest context page and transitions the destination page into a Metadata page.

Finally, the firmware transitions the source page into a Firmware page.

8.16.3 Status Codes

Table 73. Status Codes for SNP_PAGE_MOVE

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_PARAM	MBZ fields are not zero.
INVALID_ADDRESS	An address is invalid for use by the firmware or is misaligned.
INVALID_GUEST	The guest is invalid.
INVALID_GUEST_STATE	The guest is not in the correct state.
INACTIVE	The guest is not activated.
INVALID_PAGE_STATE	A page was in the incorrect state.
INVALID_PAGE_OWNER	A page was not owned by the guest.
INVALID_PAGE_SIZE	A page was not the correct size.

8.17 SNP_PAGE_MD_INIT

This command constructs a new Metadata page that can be used to store metadata entries.

8.17.1 Parameters

Table 74. Layout of the CMDBUF_SNP_PAGE_MD_INIT Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	GCTX_PADDR	Bits 63:12 of the sPA of the guest context page.
	11:0	-	-	Reserved. Must be zero.
08h	63:12	In	PAGE_PADDR	Bits 63:12 of the sPA of the page to turn into a metadata page.
	11:0	-	-	Reserved. Must be zero.

8.17.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that GCTX_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS. The firmware then checks that the page at GCTX_PADDR is in the Context state. If not, the firmware returns INVALID_GUEST.

The firmware checks that the guest is in the GSTATE_LAUNCH or GSTATE_RUNNING states. If not, the firmware returns INVALID_GUEST_STATE.

The firmware checks that PAGE_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS. The firmware then checks that the page pointed at by PAGE_PADDR is a Firmware page. If not, the firmware returns INVALID_PAGE_STATE.

The firmware zeroes the page then transitions the page into a Metadata page and setting its RMP.GPA to GCTX_PADDR.

8.17.3 Status Codes

Table 75. Status Codes for SNP_PAGE_MD_INIT

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_PARAM	MBZ fields are not zero.

Status	Condition
INVALID_ADDRESS	An address is invalid for use by the firmware or is misaligned.
INVALID_GUEST	The guest is invalid.
INVALID_GUEST_STATE	The guest is not in the correct state.
INVALID_PAGE_STATE	A page was in the incorrect state.

8.18 SNP_PAGE_SWAP_OUT

This command swaps an SNP-protected page out so that the hypervisor can relieve memory pressure or migrate the guest.

8.18.1 Parameters

Table 76. Layout of the CMDBUF_SNP_PAGE_SWAP_OUT Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	GCTX_PADDR	Bits 63:12 of the sPA of the Guest Context page.
	11:0	In	-	Reserved. Must be zero.
08h	63:12	In	SRC_PADDR	Bits 63:12 of the sPA of the source page. The page size is determined by PAGE_SIZE.
	11:0	-	-	Reserved. Must be zero.
10h	63:12	In	DST_PADDR	Bits 63:12 of the sPA of the destination page. The page size is determined by PAGE_SIZE.
	11:0	-	-	Reserved. Must be zero.
18h	63:0	In	MDATA_PADDR	Bits 63:0 of the sPA of a metadata entry. See 2.1 for the format of a metadata entry. Ignored if ROOT_MDATA_EN is 1.
20h	63:0	In	SOFTWARE_DATA	Software available data supplied by the hypervisor.
28h	63:5	-	-	Reserved. Must be zero.
	4	In	ROOT_MDATA_EN	Indicates that the metadata entry will be stored in the guest context and not in MDATA_PADDR.
	3	-	-	Reserved. Must be zero.
	2:1	In	PAGE_TYPE	Indicates the page type of the source page. 0h indicates a Data page. 1h indicates a Metadata page. 2h indicates a VMSA page. Other encodings are reserved.
	0	In	PAGE_SIZE	Indicates page size. 0 indicates a 4 KB page. 1 indicates a 2 MB page.

8.18.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that GCTX_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS. The firmware then checks that the page at GCTX_PADDR is in the Context state. If not, the firmware returns INVALID_GUEST.

The firmware checks that the guest is in the GSTATE_LAUNCH or GSTATE_RUNNING states. If not, the firmware returns INVALID_GUEST_STATE. The firmware then checks that the guest is activated. If not, the firmware returns INACTIVE.

The firmware checks that SRC_PADDR, DST_PADDR are valid sPAs. If ROOT_MDATA_EN is 0, the firmware also checks that MDATA_PADDR is a valid sPA, is aligned to the size of an MDATA structure (64 B) and does not overlap the source or destination pages. If not, the firmware returns INVALD_ADDRESS.

The firmware checks that the source page size indicated by the RMP matches the page size indicated by the PAGE_SIZE parameter. If the destination page is not a Default page, the firmware checks that the destination page size also matches the PAGE_SIZE parameter. If either check fails, the firmware returns INVALID_PAGE_SIZE.

If ROOT_MDATA_EN is 0, then the firmware checks that the page containing MDATA_PADDR is a Metadata page. If not, the firmware returns INVALID_PAGE_STATE. Then the firmware checks that the RMP.GPA of the page containing MDATA_ENTRY matches GCTX_PADDR. If not, the firmware returns INVALID_PAGE_OWNER.

This command operates on data pages, metadata pages, or VMSA pages. The firmware performs the actions in one of the following subsections depending on the value of PAGE_TYPE.

8.18.2.1 Data Pages

The actions in this section are performed only when PAGE_TYPE is 0h.

The firmware checks that the source page is a Pre-Swap or a Pre-Guest page. The firmware then checks that the destination page is a Firmware or Default page. If either check fails, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the RMP.ASID of the source page matches the ASID of the guest. If not, the firmware returns INVALID_PAGE_OWNER.

The firmware uses the guest’s VEK to decrypt the contents of the source page and uses the guest’s OEK to wrap the contents with Aead_Wrap() (see Chapter 9) without AAD. The firmware then writes the produced ciphertext into the destination page.

The firmware then constructs a MDATA structure as described in Table 77. If ROOT_MDATA_EN is 0, the firmware writes the MDATA entry at MDATA_PADDR. If ROOT_MDATA_EN is 1, the firmware writes the MDATA entry into GCTX.RootMDEntry.

Table 77. Metadata Entry (MDATA) for Data Pages

MDATA Field	Value
SOFTWARE_DATA	SOFTWARE_DATA.
IV	Generated from a CSRNG.
AUTH_TAG	Authentication tag generated by Aead_Wrap().
PAGE_SIZE	RMP.Page_Size of the source page.
VALID	1
METADATA	0
VMSA	0
GPA	gPA of the source page.
PAGE_VALIDATED	RMP.Validated of the source page.
VMPL0	RMP.VMPL0 of the source page if VMPLs are enabled. 0h otherwise.
VMPL1	RMP.VMPL1 of the source page if VMPLs are enabled. 0h otherwise.
VMPL2	RMP.VMPL2 of the source page if VMPLs are enabled. 0h otherwise.
VMPL3	RMP.VMPL3 of the source page if VMPLs are enabled. 0h otherwise.

The firmware then transitions the source page into a Pre-Guest page state.

8.18.2.2 Metadata Page

The actions in this section are performed only when PAGE_TYPE is 1h.

The firmware checks that the source page is a Metadata page. The firmware then checks that the destination page is a Firmware or Default page. If either check fails, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the RMP.GPA of the source page matches GCTX_PADDR. If not, the firmware returns INVALID_PAGE_OWNER.

The firmware uses the guest’s OEK to wrap the contents of the source page with Aead_Wrap() without AAD. The firmware then writes the produced ciphertext into the destination page.

The firmware then constructs a MDATA structure as described in Table 78. If ROOT_MDATA_EN is 0h, the firmware writes the MDATA entry at MDATA_PADDR. If ROOT_MDATA_EN is 1h, the firmware writes the MDATA entry into GCTX.RootMDEntry.

Table 78. Metadata Entry (MDATA) for Metadata Pages

MDATA Field	Value
SOFTWARE_DATA	SOFTWARE_DATA.
IV	Generated by a CSRNG.
AUTH_TAG	Authentication tag generated by Aead_Wrap().
PAGE_SIZE	RMP.Page_Size of the source page.
VALID	1
METADATA	1
VMSA	0
GPA	PADDR_INVALID.
PAGE_VALIDATED	0
VMPL0	0h
VMPL1	0h
VMPL2	0h
VMPL3	0h

The firmware then transitions the source page into a Firmware page state.

8.18.2.3 VMSA Pages

The actions in this section are performed only when PAGE_TYPE is 2h.

The firmware checks that the source page is a Pre-Swap or Pre-Guest page. The firmware then checks that the destination page is a Firmware or Default page. If either check fails, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the RMP.ASID of the source page matches the ASID of the guest. If not, the firmware returns INVALID_PAGE_OWNER.

The firmware uses the guest's OEK to wrap the contents of the source page with Aead_Wrap() without AAD. The firmware then writes the produced ciphertext into the destination page.

The firmware then constructs a MDATA structure as described in Table 79. If ROOT_MDATA_EN is 0, the firmware writes the MDATA entry at MDATA_PADDR. If ROOT_MDATA_EN is 1, the firmware writes the MDATA entry into GCTX.RootMDEntry.

Table 79. Metadata Entry (MDATA) for Data Pages

MDATA Field	Value
SOFTWARE_DATA	SOFTWARE_DATA.
IV	Generated by a CSRNG.

MDATA Field	Value
AUTH_TAG	Authentication tag generated by Aead_Wrap().
PAGE_SIZE	RMP.Page_Size of the source page.
VALID	1
METADATA	0
VMSA	1
GPA	gPA of the source page.
PAGE_VALIDATED	RMP.Validated of the source page.
VMPL0	RMP.VMPL0 of the source page if VMPLs are enabled. 0h otherwise.
VMPL1	RMP.VMPL1 of the source page if VMPLs are enabled. 0h otherwise.
VMPL2	RMP.VMPL2 of the source page if VMPLs are enabled. 0h otherwise.
VMPL3	RMP.VMPL3 of the source page if VMPLs are enabled. 0h otherwise.

The firmware then transitions the source page into a Pre-Guest page state.

8.18.3 Status Codes

Table 80. Status Codes for SNP_PAGE_SWAP_OUT

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_PARAM	MBZ fields are not zero.
INVALID_ADDRESS	An address is invalid for use by the firmware or is misaligned.
INVALID_GUEST	The guest is invalid.
INVALID_GUEST_STATE	The guest is not in the correct state.
INACTIVE	The guest is not activated.
INVALID_MDATA_ENTRY	The metadata entry is not correct.
INVALID_PAGE_STATE	A page was in the incorrect state.
INVALID_PAGE_OWNER	A page was not owned by the guest.
INVALID_PAGE_SIZE	A page was not the correct size.

8.19 SNP_PAGE_SWAP_IN

This command swaps an SNP-protected page back in.

8.19.1 Parameters

Table 81. Layout of the CMDBUF_SNP_PAGE_SWAP_IN Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	GCTX_PADDR	Bits 63:12 of the sPA of the guest context page.
	11:0	-	-	Reserved. Must be zero.
08h	63:12	In	SRC_PADDR	Bits 63:12 of the sPA of the source page. The page size is determined by PAGE_SIZE.
	11:0	-	-	Reserved. Must be zero.
10h	63:12	In	DST_PADDR	Bits 63:12 of the sPA of the destination page. The page size is determined by PAGE_SIZE.
	11:0	-	-	Reserved. Must be zero.
18h	63:0	In	MDATA_PADDR	Bits 63:0 of the sPA of a metadata entry. See 2.1 for the format of a metadata entry. Ignored if ROOT_MDATA_EN is 1.
20h	63:0	-	-	Reserved. Must be zero.
28h	63:5	-	-	Reserved. Must be zero.
	4	In	ROOT_MDATA_EN	Indicates that the metadata entry will be retrieved in the guest context and not in MDATA_PADDR.
	3	In	SWAP_IN_PLACE	If set, then SRC_PADDR and DST_PADDR are equal and the page will be swapped in place.
	2:1	In	PAGE_TYPE	Indicates the page type of the source page. 0h indicates a data page. 1h indicates a metadata page. 2h indicates a VMSA page. Other encodings are reserved.
	0	In	PAGE_SIZE	Indicates page size. 0 indicates a 4 KB page. 1 indicates a 2 MB page.

8.19.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that GCTX_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS. The firmware then checks that the page at GCTX_PADDR is in the Context state. If not, the firmware returns INVALID_GUEST.

The firmware checks that the guest is in the GSTATE_LAUNCH or GSTATE_RUNNING states. If not, the firmware returns INVALID_GUEST_STATE. The firmware then checks that the guest is activated. If not, the firmware returns INACTIVE.

The firmware checks that SRC_PADDR, DST_PADDR are valid sPAs. If ROOT_MDATA_EN is 0, the firmware also checks that MDATA_PADDR is a valid sPA and is aligned to the size of an MDATA structure (64B) and does not overlap the source and destination pages. If not, the firmware returns INVALD_ADDRESS.

If ROOT_MDATA_EN is 0, then the firmware checks that the page containing MDATA_PADDR is a Metadata page. If not, the firmware returns INVALID_PAGE_STATE. Then the firmware checks that the RMP.GPA of the page containing MDATA_ENTRY matches GCTX_PADDR. If not, the firmware returns INVALID_PAGE_OWNER.

The metadata entry used for this command is selected according to ROOT_MDATA_EN. If ROOT_MDATA_EN is set, the firmware uses the metadata entry in GCTX.RootMDEntry. If ROOT_MDATA_EN is clear, the firmware uses the metadata entry at MDATA_PADDR.

The firmware checks that the destination page size indicated by the RMP matches the page size indicated by the PAGE_SIZE parameter. If the source page is not a Default page, the firmware checks that the destination page size also matches the PAGE_SIZE parameter. The firmware then checks that the PAGE_SIZE field of the metadata entry matches the PAGE_SIZE parameter. If either check fails, the firmware returns INVALID_PAGE_SIZE.

The metadata entry determines the page type according to Table 82.

Table 82. Determining the Page Type Based on the Metadata Entry

Page Type	METADATA	VMSA
PAGE_TYPE_DATA	0	0
PAGE_TYPE_MDATA	1	0
PAGE_TYPE_VMSA	0	1

The firmware checks that the page type indicated by the metadata entry matches PAGE_TYPE. The firmware then checks that that the VALID bit in the metadata entry is set. If either check fails, the firmware returns INVALID_MDATA_ENTRY.

This command operates on data pages, metadata pages, or VMSA pages. The firmware performs the actions in one of the following subsections depending on the value of PAGE_TYPE.

8.19.2.1 Data Pages

The actions in this section are performed only when PAGE_TYPE is PAGE_TYPE_DATA.

If SWAP_IN_PLACE is 0, the firmware checks that the destination page is a Pre-Guest page. If not, the firmware returns INVALID_PAGE_STATE.

If SWAP_IN_PLACE is 1, the firmware checks that the SRC_PADDR equals DST_PADDR. If not, the firmware returns INVALID_ADDRESS. The firmware then checks that the page is in the Pre-Guest state. If not, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the RMP.ASID of the destination page matches the ASID of the guest. If not, the firmware returns INVALID_PAGE_OWNER.

The firmware uses the IV field in the metadata entry and the guest's OEK to unwrap the contents of the source page with Aead_Unwrap() with no AAD. The firmware checks that the produced authentication tag is equal to AUTH_TAG in the metadata entry. If not, the firmware returns BAD_MEASUREMENT.

The firmware clears the VALID flag in the metadata entry.

The firmware writes the plaintext produced by Aead_Unwrap() into the destination page and updates the RMP of the destination page as follows:

- Sets the RMP.GPA to GPA in the metadata entry
- Sets the RMP.VMSA to 0
- If VMPLs are enabled, sets the VMPL permission masks in the RMP entry to the VMPL permission masks in the metadata entry

If PAGE_VALIDATED in the metadata entry is 1, the firmware transitions the destination page into a Pre-Swap page.

8.19.2.2 Metadata Pages

The actions in this section are performed only when PAGE_TYPE is PAGE_TYPE_MDATA.

The firmware checks that SWAP_IN_PLACE is 0. If not, the firmware returns INVALID_PARAM.

The firmware checks that the that the destination page is a Firmware page. If not, the firmware returns INVALID_PAGE_STATE.

The firmware uses the IV field in the metadata entry and the guest's OEK to unwrap the contents of the source page with Aead_Unwrap() with no AAD. The firmware checks that the produced authentication tag is equal to AUTH_TAG in the metadata entry. If not, the firmware returns BAD_MEASUREMENT.

The firmware clears the VALID flag in the metadata entry.

The firmware writes the plaintext produced by Aead_Unwrap() into the destination page.

The firmware then transitions the destination page into a Metadata page by setting the RMP.GPA of the destination page to the GCTX_PADDR of the guest.

8.19.2.3 VMSA Pages

The actions in this section are performed only when PAGE_TYPE is PAGE_TYPE_VMSA.

The firmware checks that SWAP_IN_PLACE is 0. If not, the firmware returns INVALID_PARAM.

The firmware checks that PAGE_SIZE indicates a 4 KB page size. If not, the firmware returns INVALID_PAGE_SIZE.

The firmware checks that the destination page is a Pre-Guest page. If not, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the RMP.ASID of the destination page matches the ASID of the guest. If not, the firmware returns INVALID_PAGE_OWNER.

The firmware uses the IV field in the metadata entry and the guest's OEK to unwrap the contents of the source page with Aead_Unwrap() with no AAD. The firmware checks that the produced authentication tag is equal to AUTH_TAG in the metadata entry. If not, the firmware returns BAD_MEASUREMENT.

The firmware clears the VALID flag in the metadata entry.

The firmware writes the plaintext produced by Aead_Unwrap() into the destination page and updates the RMP of the destination page as follows:

- Sets the RMP.GPA to GPA field in the metadata entry
- Sets the RMP.VMSA to 1
- If VMPLs are enabled, sets the VMPL permission masks in the RMP entry to the VMPL permission masks in the metadata entry

If PAGE_VALIDATED in the metadata entry is 1h, the firmware transitions the destination page into a Pre-Swap page.

8.19.3 Status Codes

Table 83. Status Codes for SNP_PAGE_SWAP_IN

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_PARAM	MBZ fields are not zero.
INVALID_ADDRESS	An address is invalid for use by the firmware or is misaligned.
INVALID_GUEST	The guest is invalid.
INVALID_GUEST_STATE	The guest is not in the correct state.
INACTIVE	The guest is not activated.
INVALID_MDATA_ENTRY	The metadata entry is not correct.
BAD_MEASUREMENT	The page does not match the metadata entry’s authentication tag.
INVALID_PAGE_STATE	A page was in the incorrect state.
INVALID_PAGE_OWNER	A page was not owned by the guest.
INVALID_PAGE_SIZE	A page was not the correct size.

8.20 SNP_PAGE_RECLAIM

This command reclaims Metadata, Firmware, Pre-Guest, and Pre-Swap pages.

8.20.1 Parameters

Table 84. Layout of the CMDBUF_SNP_PAGE_PAGE_RECLAIM Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	PAGE_PADDR	Bits 63:12 of the sPAs of the page. The page size is determined by PAGE_SIZE.
	11:1	-	-	Reserved. Must be zero.
	0	In	PAGE_SIZE	Indicates page size. 0 indicates a 4 KB page. 1 indicates a 2 MB page.

8.20.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that PAGE_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS.

The firmware checks that RMP.Immutable equals 1. If not, the firmware returns SUCCESS without taking any further actions. The firmware then checks that the page is either a Metadata, Firmware, Pre-Guest, or Pre-Swap page. If not, the firmware returns INVALID_PAGE_STATE.

The firmware checks that PAGE_SIZE equals the RMP.PageSize of the page. If not, the firmware returns INVALID_PAGE_SIZE. The firmware then checks that if the page size is 2 MB, then the PAGE_PADDR is 2 MB aligned. If not, the firmware returns INVALID_ADDRESS.

The firmware transitions the provided page according to Table 85.

Table 85. State Transitions Triggered by the SNP_PAGE_RECLAIM Command

Original State	New State
Metadata	Reclaim.
Firmware	Reclaim.
Pre-Guest	Guest-Invalid.
Pre-Swap	Guest-Valid.

8.20.3 Status Codes

Table 86. Status Codes for SNP_PAGE_RO_RESTORE

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	The address is invalid for use by the firmware or is misaligned.
INVALID_PARAM	MBZ fields are not zero.
INVALID_PAGE_STATE	The page is not in the correct state.
INVALID_PAGE_SIZE	The page is not the correct size.

8.21 SNP_PAGE_UNSMASH

This command combines 512 pages of 4 KB in size into a single 2 MB page in the RMP.

8.21.1 Parameters

Table 87. Layout of the CMDBUF_SNP_PAGE_UNSMASH Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	PAGE_PADDR	Bits 63:12 of the sPAs of the page.
	11:0	-	-	Reserved. Must be zero.

8.21.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that PAGE_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS.

The firmware checks that each 4 KB page in the 2 MB region starting at PAGE_PADDR meet the following requirements.

1. Each page has RMP.PageSize that indicates a 4 KB page.
2. Each page has RMP.Immutable equal to 1.
3. Each page has RMP.VMSA equal to 0.
4. All pages are in the same state.
5. If VMPLs are enabled, then all pages have identical VMPL permissions.
6. All pages have RMP.ASID set identically and must not be zero.

If any of the above checks fail, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the range of guest physical pages are 2 MB total in size, 2 MB aligned, and consecutive. The firmware also checks that PAGE_PADDR is 2 MB aligned. If either check fails, the firmware returns INVALID_PAGE_STATE.

The firmware then turns the 4 KB pages into one 2 MB page. The resulting page is in the same state as its constituent pages.

8.21.3 Status Codes

Table 88. Status Codes for SNP_PAGE_UNSMASH

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_PARAM	MBZ fields are not zero.
INVALID_ADDRESS	An address is invalid for use by the firmware or is misaligned.
INVALID_PAGE_STATE	A page was in the incorrect state.

8.22 SNP_GUEST_REQUEST

This command sends a guest message to the firmware and returns the firmware response. See Chapter 7 for details.

8.22.1 Parameters

Table 89. Layout of the CMDBUF_SNP_GUEST_REQUEST Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	GCTX_PADDR	Bits 63:12 of the sPA of the guest context page.
	11:0	-	-	Reserved. Must be zero.
08h	63:0	In	REQUEST_PADDR	Bits 63:0 of the sPA of the request message. See Chapter 7 for details.
10h	63:0	In	RESPONSE_PADDR	Bits 63:0 of the sPA of the response message See Chapter 7 for details.

Table 90. Message Header Format

Byte Offset	Bits	Name	Description
00h	255:0	AUTHTAG	Message authentication tag. If the authentication tag for the designated algorithm is shorter than 32 B, the first bytes of AUTHTAG are used and the remaining bytes must be zero. The authentication tag authenticates the bytes from 20h to the end of the encrypted payload.
20h	127:0	IV	Message initialization vector. If the IV for the designated algorithm is shorter than 16 B, the first bytes of MSG_IV are used and the remaining bytes must be zero.
30h	7:0	ALGO	The AEAD used to encrypt this message. See Table 91.
31h	7:0	HDR_VERSION	The version of the message header. Set to 1h for this specification.
32h	15:0	HDR_SIZE	The size of the message header in bytes.
34h	7:0	MSG_TYPE	The type of the payload. See Table 92.
35h	7:0	MSG_VERSION	The version of the payload.
36h	15:0	MSG_SIZE	The size of the payload in bytes.

Byte Offset	Bits	Name	Description
38h	31:0	MSG_SEQNO	The sequence number for this message.
3Ch	7:0	MSG_VMPCK	The ID of the VMPCK used to protect this message.
3Dh	7:0	-	Reserved. Must be zero.
3Eh	15:0	-	Reserved. Must be zero.
40h-5Fh		-	Reserved. Must be zero.
60h		PAYLOAD	Encrypted payload.

Table 91. AEAD Algorithm Encodings

Value	Algorithm
0	Invalid
1	AES-256-GCM
All other encodings reserved.	

Table 92. Message Type Encodings

Value	Message Type	Message Version
0	Invalid	-
1	MSG_CPUID_REQ	1
2	MSG_CPUID_RSP	1
3	MSG_KEY_REQ	1
4	MSG_KEY_RSP	1
5	MSG_REPORT_REQ	1
6	MSG_REPORT_RSP	1
7	MSG_EXPORT_REQ	1
8	MSG_EXPORT_RSP	1
9	MSG_IMPORT_REQ	1
10	MSG_IMPORT_RSP	1
11	MSG_ABSORB_REQ	1
12	MSG_ABSORB_RSP	1
13	MSG_VMRK_REQ	1
14	MSG_VMRK_RSP	1
All other encodings reserved.		-

8.22.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that GCTX_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS. The firmware then checks that the page at GCTX_PADDR is in the Context state. If not, the firmware returns INVALID_GUEST.

The firmware checks that the guest is in the GSTATE_RUNNING states. If not, the firmware returns INVALID_GUEST_STATE.

The firmware checks that REQUEST_PADDR and RESPONSE_PADDR are valid sPAs. If not, the firmware returns INVALD_ADDRESS.

The firmware checks that the request and response page sizes indicated by the RMP are 4KB. If not, the firmware returns INVALID_PAGE_SIZE.

The firmware checks that the response page is a Firmware page. If not, the firmware returns INVALID_PAGE_STATE.

The firmware unwraps the message by setting the parameters of Aead_Unwrap() to the following:

- C: PAYLOAD
- A: Bytes 30h to 5Fh of the request message
- IV: IV
- K: The guest’s VMPCK identified by MSG_VMPCK
- T: AUTHTAG

The firmware checks that the Aead_Unwrap() did not indicate inauthenticity. If the Aead_Unwrap() function did report inauthenticity, the firmware returns BAD_MEASUREMENT.

The firmware checks that the guest’s message count of the VMPCK used to unwrap this message will not overflow by processing this message. If this check fails, the firmware returns AEAD_OFLOW.

The firmware checks that MSG_SEQNO is one greater than the guest’s message count for the VMPCK used to unwrap this message. If not, the firmware returns AEAD_OFLOW.

The firmware checks that HDR_VERSION is supported by this ABI version and that the HDR_SIZE matches the expected size for the given header version. When HDR_VERSION is 1h, then HDR_SIZE must be 60h. The firmware also checks that MSG_VERSION is supported by this ABI. If any of these checks fail, the firmware returns INVALID_PARAM.

The firmware checks that MSG_TYPE is a valid message type. The firmware then checks that MSG_SIZE is large enough to hold the indicated message type at the indicated message version. If not, the firmware returns INVALID_PARAM.

The firmware creates a message in response to the guest's message. The firmware sets MSG_SEQNO of the response message to one greater than the MSG_SEQNO of the request message. The firmware then generates a new IV and wraps the message by setting the parameters of Aead_Wrap() to the following:

- P: PAYLOAD plaintext
- A: Bytes 30h to 5Fh of the request message
- IV: Bits 95:0 of the IV
- K: The guest's VMPCK identified by VMPCK_ID

The firmware writes the resulting authentication tag into AUTHTAG and writes the ciphertext into PAYLOAD.

The firmware then increments the guest's message count for the VMPCK count by two to account for both the request message and the firmware's response message.

8.22.3 Status Codes

Table 93. Status Codes for SNP_GUEST_REQUEST

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_PARAM	MBZ fields are not zero.
INVALID_ADDRESS	An address is invalid for use by the firmware or is misaligned.
INVALID_GUEST	The guest is invalid.
INVALID_GUEST_STATE	The guest is not in the correct state.
INACTIVE	The guest is not activated.
INVALID_PAGE_STATE	A page was in the incorrect state.
INVALID_PAGE_SIZE	A page was not the correct size.
AEAD_OFLOW	The message sequence number was incorrect or the guest's message count would overflow.
BAD_MEASUREMENT	The message failed to authenticate.

8.23 SNP_DBG_DECRYPT

This command enables developers to read encrypted memory in debug enabled VMs.

8.23.1 Parameters

Table 94. Layout of the CMDBUF_SNP_DBG_DECRYPT Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	GCTX_PADDR	Bits 63:12 of the sPA of the guest context page.
	11:0	-	-	Reserved. Must be zero.
08h	63:12	In	SRC_PADDR	Bits 63:12 of the sPA of the source 4 KB region to decrypt.
	11:0	-	-	Reserved. Must be zero.
10h	63:12	In	DST_PADDR	Bits 63:12 of the sPA of the destination page to store the decrypted data.
	11:0	-	-	Reserved. Must be zero.

8.23.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that GCTX_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS. The firmware checks that GCTX_PADDR is a Context page. If not, the firmware returns INVALID_GUEST.

The firmware checks that the guest is in the GSTATE_LAUNCH or GSTATE_RUNNING guest state. If not, the firmware returns INVALID_GUEST_STATE. The firmware then checks that the guest is activated. If not, the firmware returns INACTIVE.

The firmware checks that the guest's policy allows debugging. If not, the firmware returns POLICY_FAILURE.

The firmware checks that SRC_PADDR and DST_PADDR are valid sPAs. If not, the firmware returns INVALID_ADDRESS.

The firmware checks that the page containing the 4 KB region to decrypt is a Pre-Guest, Pre-Swap, Guest-Invalid, or Guest-Valid page. The firmware also checks that the destination page is a Firmware page. If either check fails, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the source page containing the 4 KB region is owned by the indicated guest. If not, the firmware returns INVALID_PAGE_OWNER.

Note that this command always operates on 4 KB regions despite the page size indicated by the RMP entries. If the underlying page is a 2 MB page, the firmware uses the RMP entry for the 2 MB page for the RMP checks.

The firmware decrypts the contents of the 4 KB region at SRC_PADDR with the guest's VEK and writes the plaintext to DST_PADDR.

8.23.3 Status Codes

Table 95. Status Codes for SNP_DBG_DECRYPT

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_GUEST	The guest is not valid.
INACTIVE	The guest is not active.
INVALID_GUEST_STATE	The guest is not in the RUNNING or LAUNCH states.
POLICY_FAILURE	The guest policy disallows debugging.
INVALID_ADDRESS	An address is invalid or misaligned.
INVALID_PARAM	MBZ fields are not zero.
INVALID_PAGE_STATE	A page is not in the correct state.
INVALID_PAGE_OWNER	A page is not owned by the guest.

8.24 SNP_DBG_ENCRYPT

This command enables developers to write to encrypted memory in debug enabled VMs

8.24.1 Parameters

Table 96. Layout of the CMDBUF_SNP_DBG_ENCRYPT Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:12	In	GCTX_PADDR	Bits 63:12 of the sPA of the guest context page.
	11:0	-	-	Reserved. Must be zero.
08h	63:12	In	SRC_PADDR	Bits 63:12 of the sPA of the 4 KB region to be encrypted.
	11:0	-	-	Reserved. Must be zero.
10h	63:12	In	DST_PADDR	Bits 63:12 of the sPA of the 4 KB region page to store the encrypted data.
	11:0	-	-	Reserved. Must be zero.

8.24.2 Actions

The firmware checks that the platform is in the INIT state. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that GCTX_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS. The firmware checks that GCTX_PADDR is a Context page. If not, the firmware returns INVALID_GUEST. The firmware then checks that the guest is activated. If not, the firmware returns INACTIVE.

The firmware checks that the guest is in the GSTATE_LAUNCH or GSTATE_RUNNING guest state. If not, the firmware returns INVALID_GUEST_STATE.

The firmware checks that the guest's policy allows debugging. If not, the firmware returns POLICY_FAILURE.

The firmware checks that SRC_PADDR and DST_PADDR are valid sPAs. If not, the firmware returns INVALID_ADDRESS.

The firmware checks that the destination 4 KB region is a Pre-Swap or a Pre-Guest page. If not, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the destination page containing the 4 KB region is owned by the indicated guest. If not, the firmware returns INVALID_PAGE_OWNER.

Note that this command always operates on 4 KB regions despite the page size indicated by the RMP entries. If the underlying page is a 2 MB page, the firmware uses the RMP entry for the 2 MB page for the RMP checks.

The firmware encrypts the contents of the source 4 KB region at SRC_PADDR with the guest's VEK and writes the ciphertext to DST_PADDR.

8.24.3 Status Codes

Table 97. Status Codes for SNP_DBG_ENCRYPT

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_GUEST	The guest is invalid.
INACTIVATE	The guest is not activated.
INVALID_GUEST_STATE	The guest is not in the RUNNING or LAUNCH states.
POLICY_FAILURE	The guest policy disallows debugging.
INVALID_ADDRESS	An address is invalid or misaligned.
INVALID_PARAM	MBZ fields are not zero.
INVALID_PAGE_STATE	A page is not in the correct state.
INVALID_PAGE_OWNER	A page is not owned by the guest.

Chapter 9 APPENDIX: Common Algorithms

9.1 Aead_Wrap()

Inputs:

- P: Zero or more bytes to be encrypted and authenticated
- A: Zero or more bytes to be authenticated
- IV: Initialization vector (at most 96 bits)
- K: Key used to encrypt and authenticate the plaintext and AAD (256 bits)

Outputs:

- C: The encrypted plaintext
- T: Authentication tag (128 bits)

Algorithm:

1. If len(IV) < 96, then let IV’ = 096-len(IV) || IV. Otherwise, IV’ = IV
2. Let (C,T) = GCM-AEK(IV’, P, A)
3. Return (C, T)

9.2 Aead_Unwrap()

Inputs:

- C: Zero or more bytes to be decrypted and authenticated
- A: Zero or more bytes to be authenticated
- IV: Initialization vector (at most 96 bits)
- K: Key used to encrypt and authenticate the plaintext and AAD (256 bits)
- T: Authentication tag (128 bits)

Outputs:

- P: The decrypted plaintext or indication of inauthenticity

Algorithm:

1. If len(IV) < 96, then let IV’ = 096-len(IV) || IV. Otherwise, IV’ = IV
2. Let P = GCM-ADK(IV’, C, A, T)
3. Return P