SEV Secure Nested Paging Firmware ABI Specification

Publication #	**56860** Revision: **1.59**
Issue Date:	**August 2026**

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.

Contents

List of Figures

List of Tables

Revision History

Date	Revision	Description
August 2026	1.59	**Updates and Additions:** • Section 1.4: Added SEV-TIO ABI reference • Section 2.2: Updated VCEK description, added CSVSEK • Section 2.3: Added TCB version for "Venice" • Added section 2.4: ETCB_VERSION structure • Added section 2.6: TPM_INFO structure • Section 3.2: Updated Platform state machine commands • Added section 3.8: dTPM Binding to Guest Attestation • Added section 3.9: Extended TCB • Section 4.1: Updated Guest Context fields • Section 4.11: Removed mentions of Guest-Assisted Migration • Section 4.12: Guest-Assisted Migration deprecated • Section 4.13: Added many new Feature bits • Section 5.2: Description updated, a picture added • Section 6.1: New command identifiers added • Section 7.2: ETCB Version fields added • Section 7.3: ETCB and CSVCEK fields added to structures • Section 7.4: GCTX Field Structure updated, IMI/IMD deprecated, CSVCEK added • Section 7.5: VM Import deprecated • Section 7.6: VM Absorb deprecated • Section 7.7: IMD checks removed, added new IN_GCTX fields • Section 7.8: IMI checks removed • Added section 7.10: Get CSR guest message • Added section 7.11: Get TPM Info guest message • Section 8.5: Added VIOMMU • Added section 8.6: SNP_PLATFORM_STATUS_EX • Section 8.7: Added ETCB_VERSION • Section 8.9: Updated MAX_SNP_ASID and ciphertext hiding logic, added VIOMMU. • Section 8.12: Updated MAX_SNP_ASID logic • Section 8.13: Minor updates to Actions • Section 8.16: Minor update to Actions • Section 8.17: Added CVSEK, deprecated IMI/IMD • Section 8.18: Deprecated IMI/IMD, updated VMPL permission logic • Section 8.20, 8.33, 8.38: Added CSVCEK • Section 8.21, 8.24: Added feature bit check • Section 8.23: Deprecated PAGE_TYPE, added RMP.VMSA checks • Section 8.27: Added new guest REQ and RSP message encodings, deprecated VM Import and VM Absorb messages, updated Actions

Date	Revision	Description
		• Removed sections SNP_RMP_CREATE, SNP_RMP_INSTALL, SNP_RST_CREATE, SNP_RMPSEG_CREATE, SNP_RMPSEG_INSTALL and SNP_RST_INSTALL • Added section 8.34: SNP_INIT_START • Added section 8.35: SNP_INIT_CONTINUE • Added section 8.36: SNP_INIT_FINISH • Added section 8.39: SNP_CSVCEK_CERT • Added section 8.40: SNP_GET_CORIM • Added section 8.41: SNP_GET_CSR • Added section 8.42: SNP_GET_TPM_INFO • Added Appendix C: CSVCEK Certificate Chain • Added Appendix D: Certificate Signing Requests
May 2025	1.58	**Updates and Additions:** • Added Section 3.5 on Mitigation Status • Added Section 8.37 SNP_VERIFY_MITIGATION command • Incremented Chapter 7 SNP_GUEST_REQUEST message versions and message structures • Incremented ATTESTATION_REPORT version
January 2025	1.57	**Updates and Additions:** • Section 2.2: VCEK o The Versioned Chip Endorsement Key (VCEK) is an attestation signing key derived from chip-unique secrets and a TCB_VERSION. The TCB_VERSION contains TCB components and associated versions. The VCEK can be computed for any TCB_VERSION less than or equal to the CurrentTcb (see Section 3.3 for details), allowing for migrations of secrets from previous versions to the current version. The VCEK is endorsed by a certificate chain whose root corresponds to an AMD manufacturing CA. o There are two versions of the VCEK supported in EPYC – the "legacy" VCEK and chip-secret VCEK (CSVCEK). The certificate chain for the legacy VCEK can be retrieved from the Key Distribution Service (KDS). The certificate chain for the CSVCEK is retrieved through the SNP ABI. Moreover, the CSVCEK certificate chain is derived using a Device Identifier Composition Engine (DICE). For more information about DICE see *https://trustedcomputinggroup.org/work-groups/dice-architectures/*. CSVCEK functionality was introduced in version 1.59 of the ABI. The VCEK must be determined to be secure to allow for secure generation of attestation reports. • 2.3: TCB_VERSION Added "Turin" structure. • Sections 4.13, 7.2, and 8.5: Added Alias check logic for CVE-2024-21944. • Section 8.2 DOWNLOAD_FIRMWARE_EX Updated logic flow. • Sections 4.13, 5.2, 8.33, 8.34, 8.35, 8.36, 0, 0, and 8.9: Modified

Date	Revision	Description
		support for RMP segmented and non-segmented functionality, modified SNP_RST_CREATE and SNP_RMPSEG_INSTALL parameters.
October 2024	1.56	**Updates and Additions:** • Section 3.2: Added UNINIT state valid for FEATURE_INFO • Sections 4.13, 7.2, and 8.33: Added SNP_HV_REPORT_REQ command for the Host to request a Guest attestation report, clarified SNP_MSG_REPORT_REQ for Guest provided data • Section 4.13 and 8.31: Added support for the SNP_TSC_INFO command to support Secure TSC. • Section 4.13: Added support for IOMMU Buffer write safe checks. • Sections 4.13, 5.2, 8.33, 8.34, 8.35, 8.36, 0, 0, and 8.9: Added support for RMP segmented and non-segmented functionality • Section 4.13: Clarified ciphertext Hiding support for DRAM memory • Section 7.2: Modified ATTESTATION_REPORT version • Section 7.2: Modified VMPL field to support Host initiated and Guest initiated guest attestation report. • Section 7.2: Added CPUID for CPU family / model / stepping • Table 45. Added IS_TIO_EN support. • Section 8.9: Extend SNP_INIT_EX for TIO_EN • Section 8.9: Added TIO_EN flag to the CMDBUF_SNP_INIT_EX Structure. • Section 8.9: Added requirements for HWCR MSR to be set identically across all cores. • Section 8.30: Added check for 2MB RANGES element alignment requirement in the SNP_PAGE_SET_STATE command.
September 2023	1.55	**Updates and Additions:** • Added feature capability reporting. • Added ciphertext hiding feature. • Added AES-256 XTS guest policy. • Added CXL guest policy. • Added RAPL disable guest policy. • Added ECC guest policy. • Added SNP disable on SNP shutdown feature.
November 2022	1.54	**Updates and Additions:** • Added Section 3.7 • Added VLEK capability to SNP_MSG_KEY_REQ in Section 7.2 • Added VLEK capability to SNP_MSG_REPORT_REQ in Section 7.3 • Updated SNP_COMMIT to manage the VLEK in Section 8.3 • Updated SNP_PLATFORM_STATUS to output whether the VLEK is loaded in Section 8.5 • Updated SNP_CONFIG to manage the VLEK in Section 8.6 • Added the SNP_VLEK_LOAD command in Section 8.31
August 2022	1.53	**Updates and Additions:**

Date	Revision	Description
		• Sections 3.6, 7.2, 7.3, 8.5, 8.6, and 8.9: Defined MaskChipKey and its effects on attestation and key derivation. Updated SNP_CONFIG, SNP_PLATFORM_STATUS, SNP_INIT_EX commands. • Sections 4.1, 7.5, 7.6, 7.7, 7.8, and 0: Bind MA via report ID instead of MA context address. • Section 8.9 Actions: Added check for HWCR[SmmLock] • Section 8.16 Actions: Transitioned IOMMU pages to Reclaim state instead of Hypervisor state.
August 2022	1.52	**Updates and Additions:** • Section 5.2 Added HV-fixed page state definition. • Section 6.1 Added SNP_PAGE_SET_STATE command ID. • Section 8.8 Deprecated SNP_INIT • Section 8.9 SNP_INIT_EX addition of HV-fixed pages. • Section 8.30 Added SNP_PAGE_SET_STATE
January 2022	1.51	**Updates and Additions:** • Updated Section 2.2 TCB_VERSION • Updated Section 1.1 VCEK • Added Section 3.3 Firmware Updates • Added Section 3.4 Reported TCB • Updated Section 4.1 Guest Context • Added Section 4.1 Live Update • Updated Section 4.3 Guest Policy • Updated Section 4.4 Guest Activation • Updated Section 5.3.9 SEV Legacy Commands • Updated Section 6.1 Command Identifier • Updated Section 6.2 Status Codes • Updated Section 7.2 Key Derivation • Updated Section 7.3 Attestation • Updated Section 7.4 VM Export • Updated Section 7.5 VM Import • Updated Section 7.6 VM Absorb • Updated Section 7.7 VM Absorb – No Migration Agent • Added Section 7.9 TSC Info • Added Section 8.2 DOWNLOAD_FIRMWARE_EX • Added Section 8.3 SNP_COMMIT • Updated Section 8.3 Actions • Updated Section 8.6 SNP_CONFIG • Updated Section 8.9 Actions for SNP_INIT_EX • Updated Section 8.10 Actions for SNP_GCTX_CREATE • Updated Section 8.11 Actions and Status Codes for SNP_ACTIVATE • Updated Section 8.12 SNP_ACTIVATE_EX • Updated Section 8.13 SNP_DECOMMISSION • Updated Section 8.15 SNP_SHUTDOWN

Date	Revision	Description
		• Added Section 8.16 SNP_SHUTDOWN_EX • Updated Section 8.17 SNP_LAUNCH_START • Updated Section 8.18 SNP_LAUNCH_UPDATE • Updated Section 8.19 SNP_LAUNCH_FINISH • Updated Section 8.20 SNP_LAUNCH_STATUS • Updated Section 8.21 SNP_PAGE_MOVE • Updated Section 8.22 SNP_PAGE_MD_INIT • Updated Section 8.23 SNP_PAGE_SWAP_OUT • Updated Section 8.24 SNP_PAGE_SWAP_IN • Updated Section 8.27 SNP_GUEST_REQUEST • Updated Section 8.28 SNP_DBG_DECRYPT • Updated Section 8.29 SNP_DBG_ENCRYPT • Added Appendix B: Digital Signatures
April 2021	0.90	**Updates and Additions:** • Updated Section 5.3.9 SEV Legacy Commands • Updated Section 6.1 Command Identifier • Updated Section 7.1 CPUID Reporting • Updated Section 7.3 Attestation • Updated Section 7.4 VM Export • Updated Section 7.5 VM Import • Updated Section 7.6 VM Absorb • Updated Section 7.7 VM Absorb – No Migration Agent • Updated Section 8.4 GET_ID • Added Section 8.8 SNP_INIT • Updated Section 8.9 SNP_INIT_EX • Updated Section 8.11 Actions for SNP_ACTIVATE • Updated Section 8.12 Status Codes for SNP_ACTIVATE_EX • Updated Section 8.15 SNP_SHUTDOWN • Updated Section 8.17 SNP_LAUNCH_START • Updated Section 8.18 SNP_LAUNCH_UPDATE • Updated Section 8.19 SNP_LAUNCH_FINISH • Updated Section 8.23 SNP_PAGE_SWAP_OUT • Updated Section 8.27 SNP_GUEST_REQUEST
August 2020	0.80	**Updates and Additions:** • Updated 3.2 Platform State Machine • Updated Table 16. Command Identifiers • Updated Section 7.3 Attestation • Updated Table 27. ATTESTATION_REPORT Structure • Updated Section 8.5 Actions for SNP_PLATFORM_STATUS • Updated Table 46. Layout of the STRUCT_PLATFORM_STATUS Structure • Added Section 8.6 SNP_CONFIG

Date	Revision	Description
		• Updated Section 8.9 Actions for SNP_INIT • Updated Table 54. Status Codes for SNP_INIT • Updated Section 8.14 Actions for SNP_DF_FLUSH • Updated Table 64. Status Codes for SNP_DF_FLUSH • Updated Section 8.15 Actions for SNP_SHUTDOWN
April 2020	0.70	Initial public release.

Chapter 1 Introduction

1.1 Purpose

The purpose of this document is to provide details of 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 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 *AMD 64 Architecture Programmer's Manual* (publication #40332) for the x86 ISA mechanisms related to SEV-SNP and to the whitepaper *AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More* 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, publication #40332*
PPR	*Processor Programming Reference*
SNP-WP	*AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and More*
SEV	*Secure Encrypted Virtualization API Specification, publication #55766*
KDS-VCEK	*Versioned Chip Endorsement Key (VCEK) Certificate and KDS Interface Specification, publication #57230*
SEV-TIO	*SEV-TIO Firmware Interface Specification, publication #58271*

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	MDATA_INFO	Internally used MDATA fields.
30h	63:0	–	Reserved.
38h	63:0	–	Reserved.

2.2 VCEK

The Versioned Chip Endorsement Key (VCEK) is an attestation signing key derived from chip-unique secrets and a TCB_VERSION. The TCB_VERSION contains TCB components and associated versions. The VCEK can be computed for any TCB_VERSION less than or equal to the CurrentTcb (see *Section 3.3* for details), allowing for migrations of secrets from previous versions to the current version. The VCEK is endorsed by a certificate chain whose root corresponds to an AMD manufacturing CA.

There are two versions of the VCEK supported in EPYC – the "legacy" VCEK and chip-secret VCEK (CSVCEK). The certificate chain for the legacy VCEK can be retrieved from the Key Distribution Service (KDS). The certificate chain for the CSVCEK is retrieved through the SNP ABI. Moreover, the CSVCEK certificate chain is derived using a Device Identifier Composition Engine (DICE). For more information about DICE see *https://trustedcomputinggroup.org/work-groups/dice-architectures/*. CSVCEK functionality was introduced in version 1.59 of the ABI.

The VCEK must be determined to be secure to allow for secure generation of attestation reports.

2.3 TCB_VERSION

The TCB_VERSION is a structure containing the security version numbers of each component in the trusted computing base (TCB) of the SNP firmware. A TCB_VERSION is associated with each image of firmware.

AMD SoCs have associated different TCB_VERSION structure definitions. The tables below provide information for each program.

The new TCB_VERSION structure is described in *Table 4* for processors codenamed "Venice". It contains only those components that could impact the security integrity of the ASP specifically with respect to the security of the VCEK. Note also that the ASP boot loader was removed in "Venice."

Table 3. Structure of the TCB_VERSION Field for "Venice" based programs

Bits	Field	Description
63:24	–	• Reserved.
23:16	SNP	• Version of the SNP firmware. • Security Version Number (SVN) of SNP firmware.
15:8	TEE	• Current ASP OS version. • SVN of ASP operating system.
7:0	FMC	• Current ASP FMC firmware version. • SVN of ASP FMC firmware.

The TCB_VERSION structure is described in *Table 4* for processors formerly codenamed "Turin" with Family 1Ah and Models 90h-AFh and Models C0h-CFh.

Table 4. Structure of the TCB_VERSION Field for "Turin" based programs

Bits	Field	Description
63:56	MICROCODE	• Lowest current patch level of all cores
55:32	–	• Reserved
31:24	SNP	• Version of the SNP firmware • Security Version Number (SVN) of SNP firmware
23:16	TEE	• Current ASP OS version • SVN of ASP operating system
15:8	BOOT_LOADER	• Current bootloader version • SVN of ASP bootloader
7:0	FMC	• Current FMC firmware version • SVN of FMC firmware

The TCB_VERSION structure is described in *Table 5* for processors formerly codenamed "Genoa" with Family 19h Models 00h-1Fh and for "Milan" with family 19h Models 00h-0Fh.

Table 5. Structure of the TCB_VERSION Field for "Genoa" and "Milan" based programs

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

2.4 ETCB_VERSION

The Extended TCB (ETCB) is 256-bit in size and contains the CPU ucode as well as additional components that could affect the integrity of the Guest.

Table 6. Structure of the ETCB_VERSION Field for "Venice" based programs

Bits	Field	Description
255:112	–	• Reserved
111:104	AMD Root Guest (ARG)	• SVN of ARG
103:96	DPE_DRIVER (ASP)	• DPE Driver version • SVN of DPE Driver firmware
95:88	FHP_DRIVER (ASP)	• FHP Driver version • SVN of FHP Driver firmware
87:80	ASP_OS_DRIVER (ASP)	• ASP OS Driver version • SVN of ASP OS Driver firmware
79:72	SOC_DRIVER (ASP)	• SoC Driver firmware version • SVN of SoC Driver firmware
71:64	MICROCODE	• Lowest current patch level of all cores • SPL (Security Patch Level) of all cores
63:56	TMPM	• Current TMPM firmware version • SVN of TMPM firmware
55:48	PRE_ESID (ASP)	• Current PreEsid driver firmware version • SVN of PreEsid driver firmware
47:40	BOOT_DRIVER (ASP)	• Current driver boot firmware version • SVN of driver boot firmware
39:32	HAD_DRIVER (ASP)	• Current High Availability and Debuggability (HAD) firmware version • SVN of HAD firmware
31:24	ART_RT	• Current AMD Root of Trust (ART) version • SVN of ART runtime
23:16	ART_FMC	• Current ART FMC version • SVN of ART FMC
15:8	MP1	• Power Management firmware version • SVN of MP1 firmware
7:0	IP_KEY_MANAGER (ASP)	• Current IP Key Manager firmware version • SVN of IP Key Manager firmware

2.5 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**: 0h is a valid sPA and gPA.*

2.6 TPM_INFO

The TPM_INFO is the structure representing the TPM binding object. It is formatted as follows.

Table 7. Layout of the TPM_INFO structure

Byte Offset	Bits	Name	Description
0h	63:0	–	Must be zero.
08h	31:0	VERSION	Version of this structure.
0Ch	31:5	–	Reserved. Must be zero.
	4:2	SIGNING_KEY	Encodes the key used to sign this report. 0: VCEK. 1: VLEK. 2: Chip-secret VCEK. 3-6: Reserved. 7: None.
	1	GUEST	Indicates if this was retrieved by the host or the guest. 0: Retrieved by the host. 1: Retrieved by a guest.
	0	VALID	Indicates that a TPM is present and we measured its identity. The following fields are valid only if this field is set. If cleared, the remainder of this structure is zeroed and is not signed.
10h	255:0	REPORT_ID	If VALID is set, the REPORT_ID of the guest this is bound to. Zero otherwise.
30h	383:0	EK_DIGEST	If VALID is set, SHA-384 digest of the public component of the TPM EK collected during platform boot. Zero otherwise.
60h	127:0	NONCE	If VALID is set, the nonce supplied by the caller. Zero otherwise.
70h	31:0	NV_INDEX	If VALID is set, the NV index used to retrieve the TPM EK. Zero otherwise.
74h	255:0	Reserved	Must be zero.

Byte Offset	Bits	Name	Description
94h-294h	512 bytes	SIGNATURE	Signature of bytes 0h to 93h inclusive of this report. The format of the signature is described in *Chapter 10* of this ABI.

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 8. 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.	DOWNLOAD_FIRMWARE GET_ID SNP_INIT SNP_INIT_EX SNP_PLATFORM_STATUS SNP_DF_FLUSH SNP_SHUTDOWN SNP_SHUTDOWN_EX INIT_START SNP_CONFIG DOWNLOAD_FIRMWARE_EX SNP_COMMIT SNP_FEATURE_INFO
INIT	1h	The platform is initialized.	All SNP commands except DOWNLOAD_FIRMWARE SNP_INIT SNP_INIT_START SNP_VLEK_LOAD

3.3 Firmware Updates

Each SNP firmware is associated with a firmware version that comprises a major version, minor version, and build number. When loaded, the firmware tracks its firmware version with CurrentVersion. SNP firmware images are also associated with a security version number (SVN). Together with the SVNs of the other TCB components, loaded firmware tracks its current TCB_VERSION in CurrentTcb.

The hypervisor may request to replace the current firmware image with a different firmware image using the DOWNLOAD_FIRMWARE_EX command. This is usable when the SNP firmware is in either the UNINIT or INIT states, but SEV-legacy firmware must be in the UNINIT state. When the new firmware image is installed, the CurrentVersion and CurrentTcb are updated with the new firmware image's version and SVN.

The firmware supports provisional updates such that the hypervisor can roll back to previously loaded firmware if it chooses. To accomplish this, the firmware tracks the committed firmware version and TCB_VERSION in the CommittedVersion and CommittedTcb fields. When CommittedVersion is equal to CurrentVersion, the currently loaded firmware is committed. When CommittedVersion is less than CurrentVersion, the currently loaded firmware is provisional.

Provisional firmware execution is identical to committed firmware execution except that the TCB_VERSION used to derive the VCEK for key derivation and attestation reports never exceeds the CommittedTcb.

When executing provisionally installed firmware images, the hypervisor may choose to commit or roll back. To commit, the hypervisor calls SNP_COMMIT, which updates CommittedVersion and CommittedTcb to CurrentVersion and CurrentTcb, respectively. To roll back, the hypervisor invokes DOWNLOAD_FIRMWARE_EX with the image of the previously committed firmware version.

As an example, consider a platform that boots with version 1.51.1 stored in flash. The hypervisor may provisionally install an image of version 1.51.21 with DOWNLOAD_FIRMWARE_EX. At this point, CurrentVersion is 1.51.21 and CommittedVersion is 1.51.1. The hypervisor may either invoke SNP_COMMIT to set CommittedVersion to 1.51.21, or the hypervisor may invoke DOWNLOAD_FIRMWARE_EX with the firmware image of 1.51.1 to roll back. The firmware will reject any firmware update other than to 1.51.1. After committing to 1.51.21, the hypervisor may provisionally install newer versions of firmware.

Each firmware image is also associated with a minimum version from which it can live upgrade called MinUpgradeFrom. DOWNLOAD_FIRMWARE_EX uses this number to determine if the current firmware version is too far from the provided firmware image to be capable of changing with SNP guests running. In this scenario, the hypervisor must invoke SNP_SHUTDOWN before executing DOWNLOAD_FIRMWARE_EX to return the firmware to UNINIT. MinUpgradeFrom is set per image based on the specific nature of the upgrade and technical limitations of upgrading from distant past versions.

Guest context pages are versioned with the last firmware version that touched them. If CurrentVersion is different from the latest firmware version that touched the guest context page, the firmware will upgrade or downgrade the context page. A command that triggers an upgrade of the guest context page to a provisional version of the firmware may fail by returning UPDATE_FAILED. A failed update does not alter the guest context page.

If a guest context page is updated to a provisional firmware version, then updating the context page back to the committed version after a rollback will always succeed.

3.4 Reported TCB

The firmware maintains a TCB_VERSION called ReportedTcb. ReportedTcb is used to derive the VCEK that signs the attestation report.

ReportedTcb is initially set to the CurrentTcb. When SNP_CONFIG is invoked with a non-zero REPORTED_TCB parameter, ReportedTcb is set to the provided value. ReportedTcb is reset to CommittedTcb either on SNP_COMMIT or if SNP_CONFIG is provided a zero REPORTED_TCB.

ReportedTcb can be used by hypervisors to decouple the installation of a new firmware image from the use of its new VCEK. A hypervisor can install a new firmware image and then set ReportedTcb via SNP_CONFIG so that all attestation reports are still signed with VCEK. This allows a hypervisor the opportunity to ensure that guest owners have retrieved the VCEK certificates before using the new VCEK.

3.5 Mitigation Status

Mitigation status for security vulnerabilities which potentially impact the SEV-SNP TCB are available on the platform to allow the Host OS/HV (Host) and the Guest VMs (Guest) to determine status for specific vulnerability mitigations.

Mitigation status information is contained within mitigation vector data objects. There are two mitigation vector types available, the supported mitigation vector and the currently verified/enabled mitigation vector. These vector values are tracked like the TCB data and apply for CURRENT and LAUNCH states.

3.5.1 Guest Mitigation Status

Mitigation information content is accessible to the Guest VMs (Guest) and is provided via the Guest Attestation Report and the Guest Secrets Page. The Guest information contains only the mitigation vector values that are currently verified/enabled on the platform, while the Host OS/HV (Host) receives both the verified mitigation vector as well as the supported mitigation vector values. Note that the Guest has the option to add in a specific mitigation vector value to include into the Derived Guest Key. This mitigation vector for the derived key must only contain mitigation bits that have been set in the launch mitigation vector value set at Guest launch.

3.5.2 Host Mitigation Status

The SNP_VERIFY_MITIGATION command allows the Host to initiate a request to perform mitigation operations and to determine the verified mitigations and supported mitigation status information.

The steps required to mitigate a specific vulnerability may vary depending on the nature of the vulnerability. The AMD Security Bulletin associated with each vulnerability will contain mitigation details for that specific issue. In some cases, the SEV FW will automatically perform the necessary mitigation steps as part of the SNP_INIT function. In other cases, the host may be able to call the SNP_VERIFY_MITIGATION function to resolve the vulnerability without performing SNP_INIT.

The Host may issue the SNP_VERIFY_MITIGATION command to access the status of one or more mitigations and obtain several elements of information regarding the mitigation status:

- Provide status regarding if the current SEV firmware has the mitigation verification support available to verify the mitigation (CurrentSupMitVector)
- Provide information concerning the status of this specific mitigation (CurrentMitVector)

***Note**: If the SNP_VERIFY_MITIGATION command is not supported in the currently loaded SEV firmware, then INVALID_COMMAND will be returned.*

The SNP_VERIFY_MITIGATION command can be determined to be supported by using the FEATURE_INFO command Fn8000_0024_ECX_x00 bit 13.

The Host may issue the request the current vector of mitigations which are supported and verified (CurrentSupMitVector/CurrentMitVector). The Host can use these vector sets to determine if there are sufficient mitigations that are presently supported, and what has been reported.

Some mitigation solutions require the SEV firmware to be in a specific state. This information will be provided for each mitigation solution. The firmware will check the SNP firmware state based on the mitigation requirements, if the SNP firmware state is incorrect, then the firmware will return INVALID_PLATFORM_STATE.

3.6 Chip Key Masking

In version 1.53, the firmware maintains a flag, MaskChipKey, that controls use of the VCEK in guest attestation and guest key derivation. MaskChipKey initializes to 0 at SNP_INIT_EX and can be set by the hypervisor using the SNP_CONFIG command.

When MaskChipKey is 1, the attestation report will not be signed and will contain zeroes instead of a signature. Also, MaskChipKey prevents the guest from using the VCEK in guest key derivations.

3.7 Versioned Loaded Endorsement Key

A Versioned Loaded Endorsement Key (VLEK) is a versioned Elliptic Curve Digital Signature Algorithm (ECDSA) P-384 signing key certified by AMD and used by SNP firmware to sign attestation reports as an alternative to the VCEK. While the VCEK is derived from a chip-unique seed, the VLEK is derived from a seed maintained by the AMD Key Derivation Service (KDS). Each Cloud Service Provider (CSP) that enrolls with AMD has dedicated VLEK seeds.

The CSP requests from the KDS the VLEK hashstick for a given TCB and machine identified by CHIP_ID. The KDS calculates the hashstick based on the CSP's VLEK seed and the request's TCB and then wraps the VLEK hashstick with a transport key derived from a chip-unique secret. The CSP then provisions the platform with the wrapped VLEK hashstick using the SNP_VLEK_LOAD command.

With the VLEK hashsticks loaded, the firmware can use the VLEK as a drop-in replacement for the VCEK in all use cases. The hypervisor can restrict guests to use only the VLEK with the VCEK_DIS flag in SNP_LAUNCH_FINISH. Otherwise, guests can select whether to use the VLEK or the VCEK in key derivation and attestation report signing.

The AMD KDS will provide an interface to retrieve the VLEK certificate for a CSP at a specified TCB version. The VLEK certificates authenticate the VLEK as owned by AMD.

VLEK functionality was introduced in version 1.54.

3.8 dTPM Binding to Guest Attestation

When a guest runs on a system that depends on a discrete TPM (dTPM), binding the identity of the dTPM to a guest attestation report allows TPM quotes of host software and other measurements to be consumable by a guest relying party.

3.8.1 dTPM EK Fingerprint

The dTPM Endorsement Key (EK) is installed into the dTPM during data center intake of an EPYC server containing a TPM. The EK is assumed to uniquely identify the dTPM and is used by relying parties to validate the authenticity of TPM quotes.

The fingerprint of the EK is the SHA-384 digest of its public key component. During platform boot, before the x86 cores are released from reset, the ASP retrieves the EK public key from the dTPM from a predetermined NV index, calculates its fingerprint, and stores the fingerprint and its NV index into ASP private memory. This stored fingerprint represents the dTPM presented at boot.

3.8.2 Trusted Boot

When the x86 cores are released, the existing Trusted Platform Boot feature is used to authenticate and measure the x86 boot chain. After the trusted boot process is completed, the PCRs of the TPM have been extended with measurements of all components and configurations of the TCB of the platform.

At any point after trusted boot is completed, software can request a quote from the TPM which produces attestation evidence including the current PCR values of the platform. The EK is used in the process of authenticating the TPM quote.

3.8.3 Guest Attestation

A guest can request an attestation report with the existing SEV-SNP features. The attestation report contains evidence of the boot configuration of the guest as well as various platform configurations. The guest attestation report is signed by the SEV attestation key, either the VCEK or the VLEK as chosen by the host software.

Inside the guest attestation report is the REPORT_ID field. This value is 256 bits and generated by a hardware RNG at every guest launch and is constant for the lifetime of the guest. This value is used by the SNP_GET_TPM_INFO_DIGEST host command and the SNP_MSG_GET_TPM_INFO_REQ guest message.

3.8.4 Binding Guest Attestation to TPM EK

The SEV firmware constructs a cryptographic binding between the guest attestation report and the TPM EK by constructing a TPM binding object with the following data,

- TPM EK fingerprint
- Guest REPORT_ID
- TPM NV index
- Nonce

and signing the object with the SEV attestation key—either the VCEK or the VLEK as chosen by the host software.

The relying party receives the following objects:

- SEV attestation key certificate chain
- Guest attestation report
- TPM EK certificate chain
- TPM attestation key (AK) blob
- TPM quote
- TPM binding object (TPM_INFO structure. See *Table 7*).

To perform attestation verification, the relying party proceeds as follows:

1. Validates the signature on the guest attestation report with the SEV attestation key certificate chain.
2. Verifies the evidence in guest attestation report against internal policy.
3. Authenticates the TPM quote with a process that uses the EK certificate chain and AK blob.
4. Verifies the evidence in the TPM quote against internal policy.
5. Validates signature on TPM binding object with the SEV attestation key certificate chain.
6. Verifies that the REPORT_ID in the guest attestation report is equal to the REPORT_ID in the TPM binding object.
7. Verifies that the TPM EK fingerprint in the TPM binding object matches the TPM EK certificate chain.

Step (6) connects the TPM binding object to the guest and its attestation report. Step (7) connects the TPM binding object to the TPM EK and the TPM quote.

3.9 Extended TCB

SEV/SNP customers require verifiable security on the platform including verifiable security of the Guest. AMD provides various attestation processes and attestation reports to provide proof that the current platform configuration contains acceptable security functionality and is in a secure configuration state. The Trusted Computing Base (TCB) is an important aspect of security policy enablement, with trust in the TCB required to make progress in ascertaining the security of the platform.

Attestation evidence (conveyed in reports) is an important aspect of the security verification of the TCB under use. Guest security verification utilizes AMD security processor (ASP) SEV firmware-originated attestation reports. These reports include the existing TCB structural elements within the attested data, which provides a secure verifiable method for the Host OS/HV and Guest VMs to verify the platform configuration. The TCB structural contents include specific version information for the TCB elements that can be helpful for determining if the TCB components within the platform conform to acceptable security revisions.

To provide a secure platform verification, there are two areas of security under consideration that must be determined by the Host OS and Guest VMs to be trusted and secured.

- TCB of the Guest
- TCB of the ASP

The TCB of the Guest consists of all ASP TCB components, as well as other platform components uniquely relevant to the security of the Guest.

3.9.1 Solution

The added separation of the Guest TCB attestation generation from the ASP TCB attestation generation is managed with the addition of an Extended TCB (ETCB). The ETCB will be in addition to the current TCB, with updated components.

TCB and ETCB structure elements are separated so that only those components which impact the security of the ASP related to the VCEK are contained within the TCB, while all other components related to guest security are in the ETCB. The Guest security could be impacted from components in both the TCB as well as the ETCB. If a security issue is associated with a component in the TCB, then the impact to the ASP and provided key derivations is impacted.

The new TCB_VERSION structure contains only those components that could impact the security integrity of the ASP specifically with respect to the security of the VCEK.

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 and then activating the guest on a specific ASID and adding an initial set of plaintext pages into the guest address space. After the guest has begun execution, it may request attestation reports and derived keys, and it may 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 9 below*.

Table 9. 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.
MaReportId	No	The attestation report ID of the migration agent of the guest, if the guest is associated with a migration agent.
MaReportIdValid	No	Indicates if the MaReportId is valid.
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.
OekIvCount	Yes	The IV counter used for encryption with the OEK.
VEK	No	The VM encryption key used to encrypt the guest's memory.
VMPCK0, VMPCK1,	Yes	The VM communication keys.

Field	Migrated?	Description
VMPCK2, VMPCK3		
VMRK	Yes	The VM root key provided by the MA at guest launch or guest import.
HostData	No	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.
CsVcekEn	No	Indicated whether legacy or chip-secret VCEK used by guest
CsVcekPref	No	Indicated whether legacy or chip-secret VCEK preferred for guest report signing
VcekDis	Yes	Indicates that the VCEK is disabled for this guest.
ReportID	No	Attestation report ID
RootMDEntry	Yes	The root metadata entry.
IMD	Yes	Unsupported. Measurement of the Incoming Migration Image (IMI). Set to 0h.
IMIEn	No	Unsupported. Set to zero.
GOSVW	Yes	Guest OS visible workarounds. Provided in SNP_LAUNCH_START by the hypervisor.
DesiredTscFreq	Yes	Desired TSC frequency of the guest in kHz.
PspTscOffset	Yes	Offset applied to guest TSC reads.
LaunchTcb	Yes	The CurrentTcb of the firmware at the time the guest was created, imported, or absorbed.
LaunchMitVector	Yes	The CurrentMitVector value at the time the guest was created, imported, or absorbed.
LastAccessVersion	No	The CurrentVersion of the firmware that last updated or created this guest context page.
LaunchEtcb	Yes	The CurrentEtcb at the time the guest was launched.

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 the 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.

Live Update

The DOWNLOAD_FIRMWARE_EX command allows the hypervisor to replace the existing firmware without affecting SNP guests. This command will update (that is, downgrade or upgrade) the internal state of the SNP firmware immediately. In contrast, guest context pages will be updated during the next command or guest message that takes the guest context page.

See *Section 3.3* for further information on live updates.

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 10. 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
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 11. 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 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 12 below*.

Table 12. Guest Policy Structure

Bit(s)	Name	Description
63:26	–	Reserved. MBZ.
25	PAGE_SWAP_DISABLE	Guest policy to disable Guest access to SNP_PAGE_MOVE, SNP_SWAP_OUT and SNP_SWAP_IN commands. If this policy option is selected to disable these Page Move commands, then these commands will return POLICY_FAILURE. 0: Do not disable Guest support for the commands. 1: Disable Guest support for the commands.
24	CIPHERTEXT_HIDING_DRAM	0: Ciphertext hiding for the DRAM may be enabled or disabled. 1: Ciphertext hiding for the DRAM must be enabled.
23	RAPL_DIS	0: Allow Running Average Power Limit (RAPL). 1: RAPL must be disabled.
22	MEM_AES_256_XTS	0: Allow either AES 128 XEX or AES 256 XTS for memory encryption. 1: Require AES 256 XTS for memory encryption.
21	CXL_ALLOW	0: CXL cannot be populated with devices or memory. 1: CXL can be populated with devices or memory.
20	SINGLE_SOCKET	0: Guest can be activated on multiple sockets.

Bit(s)	Name	Description
		1: Guest can be activated only on one socket.
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. A guest's ASID selects the key used by the memory controller to encrypt that guest's memory. The hypervisor must inform the firmware with which ASID it will execute the guest 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 is not accessible to a new guest, SNP_ACTIVATE will require that the caches are invalidated, and 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 plain text data owned by another guest exists 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. If POLICY.SINGLE_SOCKET is set for a guest executing on a system with more than one populated socket, SNP_ACTIVATE will always fail since it activates the guest on all sockets. Instead, the hypervisor can use SNP_ACTIVATE_EX to activate the guest on the core complexes of a single socket.

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 *Section 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 the guest used as part of the attestation report. For further information about the identity block, see *Section 4.6*. For attestation, see *Section 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 and 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 an 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 third parties to establish trust. The third 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 an 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 on the identity of the guest or for other purposes.

4.11 Migration

Migration is supported by 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 with only 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 third party verifying the attestation report of a guest should also verify the report of the guest's MA.

When the hypervisor wishes to migrate a 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 Offline Encryption Key (OEK). 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 guest's context page 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 an 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.

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

4.12 Guest-Assisted Migration

The guest assisting the hypervisor during the migration process using an Initial Migration Image (IMI) is not supported. Likewise, measuring the IMI into the Initial Migration Digest (IMD) is deprecated.

4.13 Feature Discovery

The command provides the host and guests with a programmatic means to learn about the supported features of the currently loaded firmware. Host software invokes FEATURE_INFO with an index, and the firmware returns four 4-byte values containing feature information associated with that index. The format of each index is described later in this section.

FEATURE_INFO leverages the same mechanism as the CPUID instruction. Instead of using the CPUID instruction to retrieve Fn8000_0024, software can use FEATURE_INFO. The input to feature info includes ECX_IN which selects the CPUID subfunction. For instance, Fn8000_0024_x02 is retrieved by invoking FEATURE_INFO with ECX_IN set to 2.

Note: *The CPUID instruction will return all zeroes for Fn8000_0024.*

The hypervisor can provide Fn8000_0024 values to the guest via the CPUID page in SNP_LAUNCH_UPDATE. As with all CPUID output recorded in that page, the hypervisor can filter Fn8000_0024. The firmware will examine Fn8000_0024 and apply its CPUID policy.

The FEATURE_INFO command is valid in the UNINIT state and subsequent states.

The FEATURE_INFO command itself is present when bit 3 of offset 8h of the output buffer of SNP_PLATFORM_STATUS is 1.

The contents of Fn8000_0024 are described in *Table 13 below*.

Table 13. Contents of Each Subfunction of Fn8000_0024

Function	Bits	Field	Description
Fn8000_0024_EAX_x00	31:0	MaxIndex	The largest subfunction that FEATURE_INFO supports.
Fn8000_0024_EBX_x00	31:4	–	Reserved
	3	TioDevKeyUpdate	Support for SEV-TIO IDE key update host command
	2	VIOMMU	Support for SEV-TIO svIOMMU feature
	1	SevTio	SEV TIO commands [SEV-TIO]
	0	SevLegacy	SEV legacy commands [SEV]
Fn8000_0024_ECX_x00	31:23	–	Reserved
	22	SnpPageSwapDisablePolicy	Support for PAGE_SWAP_DISABLE guest policy
	21	–	Reserved (for the future versions)
	20	SnpGetTpmInfo	Support for SNP_GET_TPM_INFO host command
	19	ETCBSupported	Support for Extended TCB
	18	PKITransfer	PKI transfer
	17	SevEsSnpConcurrent	Supports SEV-ES and SNP at the same time
	16	PageSwapSupported	If set to 1 then guest page swap commands can be enabled on the

Function	Bits	Field	Description
			platform if allowed in guest policy. The platform has an integrated TMPM (Trusted Memory Page Management) module enabled for this feature to be present. Platforms before Turin will not support this feature.
	15	CSVCEK	Support for CSVCEK derivation
	14	–	Reserved (for the future versions)
	13	VerifyMit	Support for the SNP_VERIFY_MITIGATION command, and support for the mitigation vector inclusion for the Guest Key Derivation support.
	12	SecureTSC	Support for the SNP_TSC_INFO command.
	11	HostGuestAttest	Support for Host request to obtain the Guest attestation report SNP_HV_REPORT_REQ command.
	10	–	Reserved
	9	AliasCheck	The SEV firmware has support for validating the Alias check mitigation. Contains mitigation for CVE-2024-21944
	8	PreInitSegRMP	Support for Segmented RMP with SNP_INIT_CREATE, SNP_INIT_CONTINUE and SNP_INIT_FINISH commands.
	7	PreInitNonSegRMP	Support for Non-segmented RMP with SNP_INIT_CREATE, SNP_INIT_CONTINUE and SNP_INIT_FINISH commands
	6	EccMemReporting	Error correcting memory attestation
	5	CxlAllowPolicy	CXL guest policy requirement
	4	Aes256XtsPolicy	AES 256 XTS guest policy requirement
	3	CiphertextHidingDRAM	Cipher text hiding support for DRAM
	2	RaplDis	RAPL disable support. See SNP_INIT_EX
	1	X86SnpShutdown	X86_SNP_SHUTDOWN allowed. See SNP_SHUTDOWN_EX.
	0	Vlek	VLEK support
Fn8000_0024_EDX_x00	31:0	–	Reserved

Function	Bits	Field	Description
Fn8000_0024_EAX_x01	31:19	–	Reserved
	18	SnpTpmInfoReq	SNP_MSG_TPM_INFO_REQ guest message
	17:16	–	Reserved.
	15	SnpMsgGetCsrReq	SNP_MSG_GET_CSR_REQ guest message
	14	TioMsgSetupMappingReq	TIO_MSG_SETUP_MAPPING_REQ guest message
	13	SnpMsgConfigViommuReq	SNP_MSG_CONFIG_VIOMMU_REQ guest message
	12	TioMsgSdteWriteReq	TIO_MSG_SDTE_WRITE_REQ guest message
	11	TioTioMgsMmioConfigReq	TIO_MSG_MMIO_CONFIG_REQ guest message
	10	TioMsgMmioValidateReq	TIO_MSG_MMIO_VALIDATE_REQ guest message
	9	TioMsgTdiInfoReq	TIO_MSG_TDI_INFO_REQ guest message
	8	SnpMsgTscInfoReq	SNP_MSG_TSC_INFO_REQ guest message
	7	SnpMsgAbsorbNomaReq	SNP_MSG_ABSORB_NOMA_REQ guest message
	6	SnpMsgVmrkReq	SNP_MSG_VMRK_REQ guest message
	5	–	Reserved (~~MSG_ABSORB_REQ~~)
	4	–	Reserved (~~MSG_IMPORT_REQ~~)
	3	SnpMsgExportReq	SNP_MSG_EXPORT_REQ guest message
	2	SnpMsgReportReq	SNP_MSG_REPORT_REQ guest message
	1	SnpMsgKeyReq	SNP_MSG_KEY_REQ guest message
	0	SnpMsgCpuidReq	SNP_MSG_CPUID_REQ guest message
Fn8000_0024_EBX_x01	31:0	–	Reserved
Fn8000_0024_ECX_x01	31:0	–	Reserved
Fn8000_0024_EDX_x01	31:0	–	Reserved

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 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 RMP Initialization

This section describes two mechanisms to initialize the RMP.

The duration of the SNP_INIT_EX command increases corresponding to the increase in the size of RMP-protected memory. The duration required is due to the requirement that the entire RMP must be written by SEV firmware.

The SNP_INIT_EX command requires that the VM_HSAVE_PA MSR is set to 0h. The VM_HSAVE_PA MSR stores the physical address where host processor state information can be stored. This requirement forces guests to be suspended for the entire time the RMP initialization is in progress.

These sections describe additional commands, and additional software flows which reduces the amount of time that software must ensure guest execution is suspended (i.e., guest not scheduled).

Each of these mechanisms reduce the duration of time in which the hypervisor must suspend running (non-SNP) guests. One mechanism is designed to initialize a non-segmented RMP, and the other mechanism is designed to initialize a segmented RMP. In both cases, the RMP is initialized before SNP_INIT_EX is invoked, and software sets INIT_RMP to 0 when it invokes SNP_INIT_EX. In that regard, three new commands are defined that will allow software to pre-initialize the RMP whether it is segmented or not: SNP_INIT_START, SNP_INIT_CONTINUE and SNP_INIT_FINISH. The appropriate state transitions are depicted below:

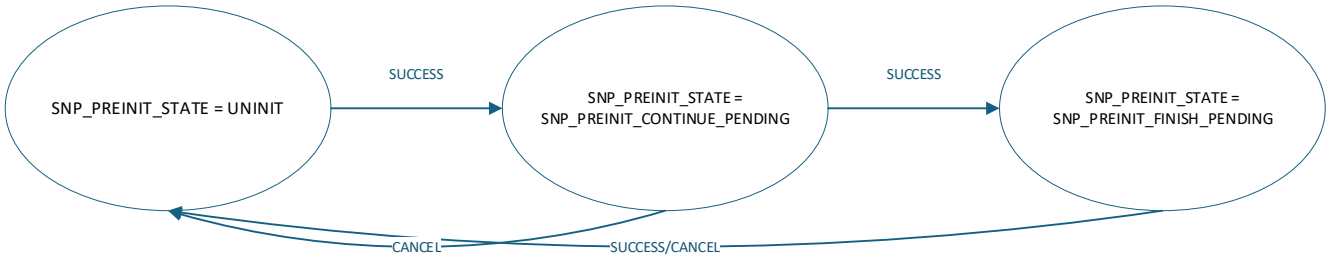


Figure 1 RMP Pre-Initialization State Transitions

5.2.1 Non-Segmented RMP

Software will invoke the SNP_INIT_START command to initialize the contents of the RMP, at which point guests cannot run. However, guests can resume as necessary after SNP_INIT_START completes. SNP FW will maintain the RMP initialization status using the SNP_PREINIT_STATE register, which is accessible by SW. Based on the pre-initialization state, SW then must invoke the SNP_INIT_CONTINUE command wherein FW will initialize RMP entries, invalidate TLB entries and restore the VM_HSAVE page addresses accordingly. After SNP_PREINIT_STATE is updated to reflect that SNP pre-initialization is ready to complete, then SW will invoke the SNP_INIT_FINISH command. Guests once again are suspended until this command is complete (as reflected by the state register).

SNP_INIT_START can only be executed in the UNINIT state and must be the first command invoked when RMP pre-initialization is desired. An SEV firmware global flag, SNP_PRE_INIT_STATE, indicates whether SNP_INIT_START can be invoked. This flag must also be in the UNINIT state. The Hypervisor must then enable SNP (set SYS_CFG[SNPEn] MSR) and invoke SNP_INIT_START. All guests are suspended after this command is invoked.

SNP_INIT_START performs the following actions:

1. Checks that the SNP platform is in the UNINIT state.
2. Checks that SNP_PREINIT_STATE is in the SNP_PREINIT_STATE_UNINIT state.
3. Checks that RMP table has not been initialized.
4. Checks that the SYS_CFG[SNPEn] flag is set for all CPU cores.
5. Disables RMPUPDATE and VM_HSAVE_PA.
6. Checks VM_HSAVE_PA is 0h for all CPU cores (i.e., no guest is running and host state save feature is disabled).
7. Enables memory protection using hardware-enforced memory fencing.
8. Enables VM_HSAVE_PA but keeps RMPUPDATE disabled.
9. Sets SNP_PREINIT_STATE to SNP_PREINIT_CONTINUE_PENDING

Guests can resume at this point. However, the RMP is not yet initialized and SW must invoke SNP_INIT_CONTINUE based on the SNP_PREINIT_STATE = SNP_PREINIT_CONTINUE_PENDING flag. Guests will once again be disabled (i.e., VM_HSAVE_PA not writeable by host).

SNP_INIT_CONTINUE results in the following actions:

1. Checks that SNP_PREINIT_STATE is SNP_PREINIT_CONTINUE_PENDING
2. If (CANCEL) then abort pre-init.
   a. Marks internally that the RMP must be re-initialized.
   b. Clears SYS_CFG[SNPEn] on all cores.
   c. Re-enable VM_HSAVE_PA and RMPUPDATE
   d. Clears SNP_PREINIT_CONTINUE_PENDING flag
   e. Returns SUCCESS to caller. Guests can resume.
3. Disables VM_HSAVE_PA
4. Initializes the RMP
5. Enables VM_HSAVE_PA
6. Set SNP_PREINIT_STATE to SNP_PREINIT_FINISH_PENDING.

The SNP_INIT_FINISH command must now be invoked to complete RMP pre-initialization. SNP_INIT_FINISH performs the following actions:

1. Checks that snp_preinit_state is SNP_PREINIT_FINISH_PENDING.
2. If CANCEL is set,
   a. Marks internally that the RMP must be re-initialized.
   b. Clears SYS_CFG[SNPEn] on all cores.
   c. Re-enable VM_HSAVE_PA and RMPUPDATE
   d. Clears SNP_PREINIT_CONTINUE_PENDING flag
   e. Returns SUCCESS to caller. Guests can resume.
3. Disable VM_HSAVE_PA
4. Check VM_HSAVE_PA is 0h across all cores. (no guest can run).
5. Marks internally that the RMP does not need to be re-initialized.
6. Enable VM_HSAVE_PA and RMPUPDATE
7. Set SNP_PREINIT_STATE to SNP_PREINIT_FINISH_COMPLETE.
8. SNP_INIT_EX can now be invoked with RMP_INIT = 0 (RMP has been initialized).

5.2.2 Segmented RMP

As with non-segmented RMP, pre-initialization of segmented RMP required invocation of SNP_INIT_START, SNP_INIT_CONTINUE and SNP_INIT_FINISH. However, based on the platform support of segmented RMP (RST, or reverse map segmented table), the sequence of actions taken by FW is different.

SNP_INIT_START can only be executed in the UNINIT state and must be the first command invoked when RMP pre-initialization is desired. An SEV firmware global flag, SNP_PRE_INIT_STATE, indicates whether SNP_INIT_START can be invoked. This flag must also be in the UNINIT state. The Hypervisor must then enable SNP (set SYS_CFG[SNPEn]) and invoke SNP_INIT_START. All guests are suspended after this command is invoked.

SNP_INIT_START performs the following actions:

1. Checks that the SNP platform is in the UNINIT state.
2. Checks that SNP_PREINIT_STATE is in the SNP_PREINIT_STATE_UNINIT state.
3. Checks that the SYS_CFG[SNPEn] flag is set for all CPU cores.
4. Disable RMPUPDATE and VM_HSAVE_PA.
5. Check VM_HSAVE_PA is 0h for all CPU cores (i.e., no guest is running and host state save feature is disabled).
6. Validate RST or return ERROR.
7. Enable VM_HSAVE_PA but keep RMPUPDATE disabled.
8. Set SNP_PREINIT_STATE to SNP_PREINIT_CONTINUE_PENDING

Guests can resume at this point. However, RST is not yet initialized and SW must invoke SNP_INIT_CONTINUE based on the SNP_PREINIT_STATE = SNP_PREINIT_CONTINUE_PENDING. Guests will once again be disabled (i.e. VM_HSAVE_PA not writeable by host).

SNP_INIT_CONTINUE performs the following actions:

1. Checks that SNP_PREINIT_STATE is SNP_PREINIT_CONTINUE_PENDING
2. If (CANCEL) then abort pre-init.
   a. Marks internally that the RST must be re-initialized.
   b. Clears SYS_CFG[SNPEn] on all cores.
   c. Re-enable VM_HSAVE_PA and RMPUPDATE
   d. Clears SNP_PREINIT_CONTINUE_PENDING flag
   e. Returns SUCCESS to caller. Guests can resume.
3. Disable VM_HSAVE_PA
4. Initialize RST.
5. Check that the pages pointed to by VM_HSAVE_PA (non-zero values) are in the HYPERVISOR or default state.
6. Removes any memory protection that prevents the CPU from reading and writing the RST.
7. Enable VM_HSAVE_PA
8. Set SNP_PREINIT_STATE to SNP_PREINIT_FINISH_PENDING.

At this point, guests may be resumed as VM_HSAVE_PA has been re-enabled. However, pre-initialization is not complete. The SNP_INIT_FINISH command must now be invoked in order to complete RST pre-initialization.

SNP_INIT_FINISH performs the following actions:

1. Checks that SNP_PREINIT_STATE is SNP_PREINIT_FINISH_PENDING.
2. If CANCEL is set,
   a. Marks internally that the RST must be re-initialized.
   b. Clears SYS_CFG[SNPEn] on all cores.
   c. Re-enables VM_HSAVE_PA and RMPUPDATE.
   d. Clears SNP_PREINIT_CONTINUE_PENDING flag.

e. Returns SUCCESS to caller. Guests can resume.
3. Disables VM_HSAVE_PA.
4. Checks VM_HSAVE_PA is 0h across all cores. (no guest can run).
5. Marks internally that the RST does not need to be re-initialized.
6. Enables VM_HSAVE_PA and RMPUPDATE.
7. Set SNP_PREINIT_STATE to SNP_PREINIT_FINISH_COMPLETE.
8. SNP_INIT_EX can now be invoked with RMP_INIT = 0 (RMP has been initialized).

5.3 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 14 below* enumerates and defines each of the page states.

Note: *(–) in a cell indicates that the page state is not dependent on that field.*

Table 14. Page State Definitions

Page State	Assigned	Validated	ASID	Immutable	GPA	VMSA
Hypervisor	0	0	0	0	–	–
HV-fixed	0	0	0	1	–	–
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.

A HV-fixed page is used by the hypervisor for its normal execution. It also may be used with other SoC services, such as the non-SNP commands to the ASP. The RMPUPDATE instruction cannot create HV-fixed pages. Instead, software should use SNP_PAGE_SET_STATE or SNP_INIT_EX with a list of the ranges that should be transitioned to HV-fixed. Once in the HV-fixed state, a page remains there until the RMP is reinitialized.

Pages in the firmware state are owned by the firmware. Because the RMP.Immu 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 nonimmutable 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 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 page into a Metadata page by making the GPA field non-zero.

A Guest-Invalid page has been donated to the guest but not yet 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.

5.3.1 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.

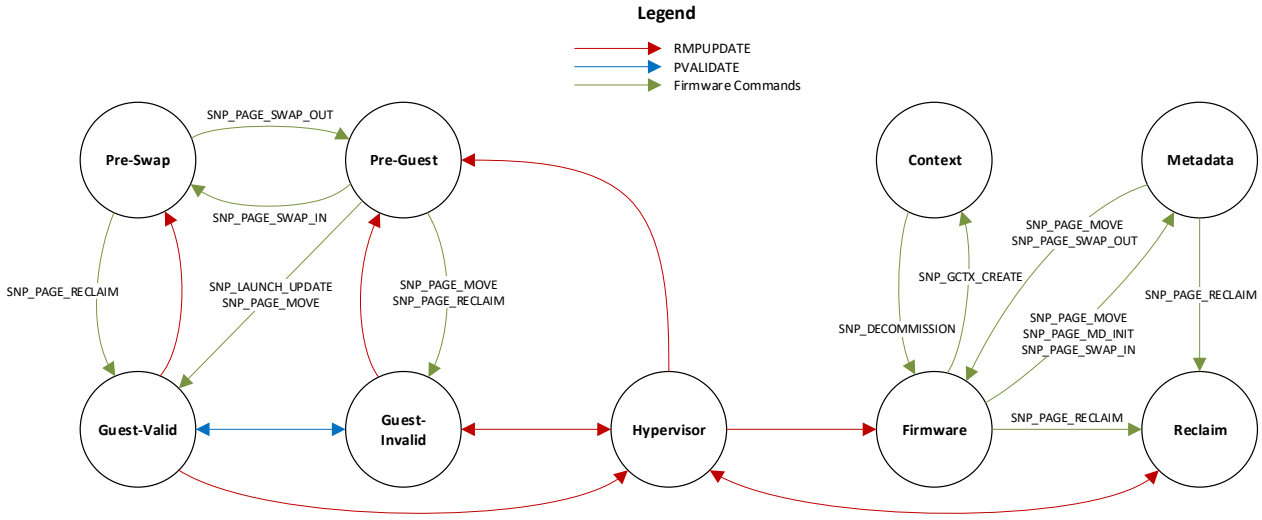


Figure 2. SNP Page State Machine

Red edges in *Figure 2* represent hypervisor actions. Blue edges represent guest actions. Green edges represent firmware commands specified in this document.

Note: *Some transitive RMPUPDATE edges are omitted for clarity.*

Actions that trigger a page state transition are depicted in *Figure 2*. 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.

5.3.2 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.

5.3.3 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.

5.3.4 Additional Page Management x86 Instructions

Additional x86 instructions are available for the Guest and the Host OS to perform Page Management. These instructions are described in [APM] volume 3.

5.3.5 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 reassignments of pages so that the appropriate TLB and cache operations happen.

5.3.6 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.

5.3.7 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.

5.3.8 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 page.

5.3.9 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.

When the RMP has been initialized, as reported by SNP_PLATFORM_STATUS, any invocation of the SEV-legacy commands INIT and INIT_EX require that TMR_PADDR must be 2 MB aligned instead of 1 MB, and TMR_LENGTH must be 2 MB instead of 1 MB.

When SNP is in the INIT state, the SEV-legacy command INIT will check that the buffer addressed by the TMR_PADDR parameter resides entirely inside Firmware pages.

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 15 below 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 15. 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 set to 0, the firmware does not interpret any other fields in the entry.

Chapter 6 Mailbox Protocol

Software on the x86 CPUs communicates 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 16 below* summarizes the additional commands and their identifiers. See the command definitions for further details.

Table 16. 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_INIT_EX	85h	Initialize platform for SNP with extended parameters.
SNP_SHUTDOWN_EX	86h	Shut down the platform with extended capabilities to shut down SNP-enforcing controls such as IOMMU SNP enforcement.
SNP_INIT_START	87h	Initializes the RMP while SYS_CFG[SNPE] is not set
SNP_INIT_CONTINUE	88h	Continues the initialization of the RMP
SNP_INIT_FINISH	89h	The last command invoked in creation of an RMP
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_HV_REPORT_REQ	96h	Host initiated request for the Guest attestation report.
SNP_TSC_INFO	97h	Host request for TSC information for a given guest.
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.

Command	ID	Description
SNP_DBG_DECRYPT	B0h	Decrypt guest memory for debugging.
SNP_DBG_ENCRYPT	B1h	Encrypt guest memory for debugging.
SNP_VERIFY_MITIGATION	B2h	Check or perform vulnerability mitigations.
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_SET_STATE	C6h	Set the page state of a set of pages to HV-fixed.
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_CONFIG	C9h	Set systemwide configuration values.
DOWNLOAD_FIRMWARE_EX	CAh	Perform a live update of SNP firmware.
SNP_COMMIT	CBh	Commit the current firmware.
SNP_VLEK_LOAD	CDh	Load the VLEK into the firmware.
FEATURE_INFO	CEh	Feature support information.
SNP_PLATFORM_STATUS_EX	CFh	Query platform information that includes ETCB
Reserved (SEV-TIO)	D0h-EFh	Reserved.
SNP_CSVCEK_CERT	F0h	Get the CSVCEK certificate chain
SNP_GET_CORIM	F1h	Get the CoRIM ID list
SNP_GET_CSR	F2h	Get CSR for the LDevID certificate
SNP_GET_TPM_INFO	F3h	Get TPM_INFO structure for a specific guest

6.2 Status Codes

This ABI introduces several new status codes to the mailbox protocol. *Table 17 below* summarizes the additional status codes added by this ABI.

Table 17. 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.

Status	Code	Description
AEAD_OFLOW	1Dh	The AEAD algorithm would have overflowed.
EXIT_RING_BUFFER	1Fh	Ring Buffer error.
RMP_INIT_REQUIRED	20h	The RMP must be reinitialized.
BAD_SVN	21h	SVN of provided image is lower than the committed SVN.
BAD_VERSION	22h	Firmware version anti-rollback.
SHUTDOWN_REQUIRED	23h	An invocation of SNP_SHUTDOWN is required to complete this action.
UPDATE_FAILED	24h	Update of the firmware internal state or a guest context page has failed.
RESTORE_REQUIRED	25h	Installation of the committed firmware image required.
RMP_INIT_FAILED	26h	The RMP initialization failed.
INVALID_KEY	27h	The key requested is invalid, not present, or not allowed.

The existing SEV status codes are provided below for reference:

Table 18. SEV Status Codes

Status	Code	Description
SUCCESS	0000h	Successful completion
INVALID_PLATFORM_STATE	0001h	The platform state is invalid for this command
INVALID_GUEST_STATE	0002h	The guest state is invalid for this command
INVALID_CONFIG	0003h	The platform configuration is invalid
INVALID_LENGTH	0004h	A memory buffer is too small.
ALREADY_OWNED	0005h	The platform is already owned
INVALID_CERTIFICATE	0006h	The certificate is invalid
POLICY_FAILURE	0007h	Request is not allowed by guest policy
INACTIVE	0008h	The guest is inactive
INVALID_ADDRESS	0009h	The address provided is invalid
BAD_SIGNATURE	000Ah	The provided signature is invalid
BAD_MEASUREMENT	000Bh	The provided measurement is invalid
ASID_OWNED	000Ch	The ASID is already owned
INVALID_ASID	000Dh	The ASID is invalid
WBINVD_REQUIRED	000Eh	WBINVD instruction required
DF_FLUSH_REQUIRED	000Fh	DF_FLUSH invocation required
INVALID_GUEST	0010h	The guest handle is invalid

Status	Code	Description
INVALID_COMMAND	0011h	The command issued is invalid.
ACTIVE	0012h	The guest is active.
HWERROR_PLATFORM	0013h	A hardware condition has occurred affecting the platform. It is safe to re-allocate parameter buffers.
HWERROR_UNSAFE	0014h	A hardware condition has occurred affecting the platform. Re-allocating parameter buffers is not safe.
UNSUPPORTED	0015h	Feature is unsupported.
INVALID_PARAM	0016h	A parameter is invalid.
RESOURCE_LIMIT	0017h	The SEV FW has run out of a resource necessary to complete the command.
SECURE_DATA_INVALID	0018h	The part-specific SEV data failed integrity checks.
RB_MODE_EXITED	001Fh	A Mailbox mode command was sent while the SEV firmware was in Ring Buffer mode. Ring Buffer mode has been exited; the Mailbox mode command has been ignored. Retry is recommended.

Chapter 7 Guest Messages

Guest messages provide the guest with 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 plain text 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 8.18* for 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 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 8.27*.

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 8.18.)*

The firmware provides a service to guests 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 SNP_MSG_CPUID_REQ message.

The guest constructs an SNP_MSG_CPUID_REQ message, which contains an array of CPUID function structures, as defined in *Table 19.* 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 19. SNP_MSG_CPUID_REQ Structure

Byte Offset	Bits	Name	Description
00h	31:0	COUNT	Number of CPUID functions to validate. Must be less or equal to 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 20. 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 SNP_MSG_CPUID_RSP message as defined in *Table 21. SNP_MSG_CPUID_RSP Structure*. The message contains the same CPUID function structures that may be altered by the firmware. The firmware will alter the function structure when the hypervisor has provided an insecure value.

If firmware encounters a CPUID function that is not in the standard range (Fn0000_0000 through Fn0000_FFFF) or the extended range (Fn8000_0000 through Fn8000_FFFF), the firmware does not perform any checks on the function output.

If firmware encounters a CPUID function that is in the standard or extended ranges, then the firmware performs a check to ensure that the provided output would not lead to an insecure guest state. If insecure function output is identified, the firmware sets the field in the response message to an acceptable value.

Note: *Some functions have multiple acceptable values, and the firmware may choose any one of them.*

The firmware then returns INVALID_PARAM in the STATUS field of the response message.

The policy used by the firmware to assess CPUID function output can be found in the [PPR].

Table 21. SNP_MSG_CPUID_RSP Structure

Byte Offset	Bits	Name	Description
00h	31:0	STATUS	The status of the CPUID reporting 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 are described in *Table 22 below*. 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 22. Data Mixed into the Derived Guest Key

Data	Description	Mix Type	Provided in Message
VCEK/VLEK/ VMRK	VCEK, VLEK, 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
Image ID	The image ID provided at launch.	Optional	No

Data	Description	Mix Type	Provided in Message
Guest Policy	The guest policy provided at launch.	Optional	No
Mitigation Launch Vector	The mitigation vector. The guest is not allowed to set any bits in this key derivation vector (LAUNCH_MIT_VECTOR) that were not originally set in SNP_VERIFY_MITIGATION's MIT_VERIFIED_VECTOR at the time the Guest was launched.	Optional	Yes
ETCB Version	The ETCB version selected by the guest.	Optional	Yes

Table 23 below describes the SNP_MSG_KEY_REQ message structure that the guest sends to the firmware to request a derived key.

Table 23. SNP_MSG_KEY_REQ Message Structure

Byte Offset	Bits	Name	Description
0h	31:4	–	Reserved. Must be zero.
	3	ETCB_SEL	0: ETCB_VERSION not included 1: ETCB_VERSION included
	2:1	KEY_SEL	0: Derive with VLEK if VLEK is installed. Otherwise, derive with the key indicated by CsVcekPref. 1: Derive with legacy VCEK. 2: Derive with VLEK. 3: Derive with chip secret VCEK.
	0	ROOT_KEY_SELECT	Selects the root key from which to derive the key. 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 24* 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 in the ID block.
18h	63:0	TCB_VERSION	The TCB version to mix into the derived key. Must not exceed LaunchTcb.
20h	63:0	LAUNCH MIT_VECTOR	The mitigation vector value to mix into the derived key. Specific bit settings corresponding to mitigations required for Guest operation. ***Note**: The guest is not allowed to set any bits in this key derivation vector (LAUNCH_MIT_VECTOR) that were not*

Byte Offset	Bits	Name	Description
			originally set in the launch mitigation vector.
28h	63:0	–	Reserved. Must be zero
30h	128:0	–	Reserved. Must be zero
40h	255:0	ETCB_VERSION	The ETCB version to mix into the derived key. Must not exceed LaunchETcb.

If ROOT_KEY_SELECT is 0 and GCTX.MaReportIdValid of the requesting guest is 1 then the firmware returns the INVALID_KEY status code to the guest.

If ROOT_KEY_SELECT is 0 and MaskChipKey is 1, the firmware returns the INVALID_KEY status code to the guest.

If ROOT_KEY_SELECT is 0, the key used is further selected by the KEY_SEL parameter. The firmware returns INVALID_KEY to the guest if any of the following conditions occur:

- KEY_SEL is 0, VcekDis is 1, and the VLEK is not installed.
- KEY_SEL is 1, and VcekDis is 1.
- KEY_SEL is 2, and the VLEK is not installed.
- KEY_SEL is 3, and VcekDis is 1 or CsVcekEn is 0.

This guest message requesting to provide a derived key will contain a flag indicating whether the guest wants to make use of the chip secret VCEK or legacy-derived VCEK. The key selection based on CSVCEK options and KEY_SEL is summarized in *Section 8.33*.

The KEY_SEL parameter was introduced in version 1.54.

The SNP_MSG_KEY_REQ detailed in *Table 23* describes the SNP_MSG_KEY_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 24 below*.

Table 24. Structure of the GUEST_FIELD_SELECT Field

Bits	Field	Description
63:7	–	Reserved. Must be zero.
6	LAUNCH_MIT_VECTOR	Indicates that the guest-provided LAUNCH_MIT_VECTOR will be mixed into the key.
5	TCB_VERSION	Indicates that the guest-provided TCB_VERSION 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

Bits	Field	Description
		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 SNP_MSG_KEY_RSP message defined in *Table 25 below* to the guest.

Table 25. SNP_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. 27h: Invalid key selection.
04h–1Fh		–	Reserved.
20h	255:0	DERIVED_KEY	The requested derived key if STATUS is 0h.

7.3 Attestation

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

The Host OS can directly request that the firmware construct a similar Guest attestation report. See the SNP_HV_REPORT_REQ command.

A guest requests an attestation report by constructing an SNP_MSG_REPORT_REQ as specified in *Table 26.* The message contains data provided by the guest in REPORT_DATA to be included in the report; the firmware does not interpret this data.

The guest sets the VMPL field to a value from 0 through 3 which indicates a request from the guest.

Table 26. SNP_MSG_REPORT_REQ Message Structure

Byte Offset	Bits	Name	Description
00h	511:0	REPORT_DATA	Guest-provided data to be included in the attestation report.
40h	31:0	VMPL	The VMPL to put in the attestation report. Must be greater than or equal to the current VMPL and, at most, three.
44h	31:2	–	Reserved.
	1:0	KEY_SEL	Selects which key to use for generating the signature.

Byte Offset	Bits	Name	Description
			0: Sign with VLEK if VLEK is installed. Otherwise, sign with the key indicated by CsVcekPref. 1: Sign with legacy VCEK. 2: Sign with VLEK. 3: Sign with chip secret VCEK.
48h–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 27*.

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 an associated migration agent, the REPORT_ID_MA is filled in with the report ID of the migration agent.

If MaskChipKey is 0, the firmware signs the attestation report with either the VCEK or VLEK based on the key selection made in KEY_SEL. The firmware will return INVALID_KEY to the guest if any of the following conditions occur:

- KEY_SEL is 0, the VLEK is not loaded, and VcekDis is 1.
- KEY_SEL is 1 and VcekDis is 1.
- KEY_SEL is 2 and the VLEK is not loaded.
- KEY_SEL is 3, and VcekDis is 1 or CsVcekEn is 0.

This guest message requesting an attestation report will contain a flag indicating whether the guest wants to make use of the chip secret VCEK or legacy-derived VCEK. The key selection based on CSVCEK options and KEY_SEL is summarized in *Section 8.33*.

The firmware uses the systemwide ReportedTcb value as the TCB version to derive the VCEK, VLEK, or CSVCEK. This value is set by the hypervisor. The firmware guarantees that the ReportedTcb value is never greater than the installed TCB version.

If MaskChipKey is 1, the firmware writes zeroes into the SIGNATURE field instead of signing the report.

***Note**: All reserved fields in the attestation report should be checked to be zero by verifiers.*

Table 27. ATTESTATION_REPORT Structure

Byte Offset	Bits	Name	Description
00h	31:0	VERSION	Version number of this attestation report. Set to 6h for this specification.
04h	31:0	GUEST_SVN	The guest SVN.
08h	63:0	POLICY	The guest policy. See *Table 12* 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 firmware sets this value depending on whether a guest (SNP_MSG_REPORT_REQ) or host (SNP_HV_REPORT_REQ) requested the guest attestation report. For a Guest requested attestation report this field will contain the value (0-3). A Host requested attestation report will have a value of 0xffffffff.
34h	31:0	SIGNATURE_ALGO	The signature algorithm used to sign this report. See *Appendix B* for encodings.
38h	63:0	CURRENT_TCB	CurrentTcb.
40h	63:0	PLATFORM_INFO	Information about the platform. See *Table 28*.
48h	31:6	–	Reserved. Must be zero.
	5	–	Reserved
	4:2	SIGNING_KEY	Encodes the key used to sign this report. 0: Legacy VCEK. 1: VLEK. 2: Chip-secret VCEK. 3-6: Reserved. 7: None.
	1	MASK_CHIP_KEY	The value of MaskChipKey.
	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	If REQUEST_SOURCE is guest provided, then contains Guest-provided data, else host request and zero (0) filled by firmware.
90h	383:0	MEASUREMENT	The measurement calculated at launch.

Byte Offset	Bits	Name	Description
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_LAUNCH_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_FINISH. Zeroes if AUTHOR_KEY_EN is 1.
140h	255:0	REPORT_ID	Report ID of this guest.
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	7:0	CPUID_FAM_ID	Family ID (Combined Extended Family ID and Family ID)
189h	7:0	CPUID_MOD_ID	Model (combined Extended Model and Model fields)
18Ah	7:0	CPUID_STEP	Stepping.
18Bh–19Fh	–	–	Reserved.
1A0h–1DFh	511:0	CHIP_ID	If MaskChipId is set to 0, Identifier unique to the chip as output by GET_ID. Otherwise, set to 0h.
1E0h	63:0	COMMITTED_TCB	CommittedTcb.
1E8h	7:0	CURRENT_BUILD	The build number of CurrentVersion.
1E9h	7:0	CURRENT_MINOR	The minor number of CurrentVersion.
1Eah	7:0	CURRENT_MAJOR	The major number of CurrentVersion.
1Ebh	7:0	–	Reserved.
1Ech	7:0	COMMITTED_BUILD	The build number of CommittedVersion.
1Edh	7:0	COMMITTED_MINOR	The minor version of CommittedVersion.
1Eeh	7:0	COMMITTED_MAJOR	The major version of CommittedVersion.
1Efh	7:0	-	Reserved.
1F0h	63:0	LAUNCH_TCB	The CommittedTcb at the time the guest was launched or imported.
1F8h	63:0	LAUNCH_MIT_VECTOR	The verified mitigation vector value at the time the guest was launched (LaunchMitVector).
200h	63:0	CURRENT_MIT_VECTOR	Value is set to the current verified mitigation vector value (CurrentMitVector).
208h	192:0	–	Reserved. MBZ.
220h	255:0	CURRENT_ETCB	CurrentEtcb.

Byte Offset	Bits	Name	Description
			See *Section 3.9*.
240h	255:0	LAUNCH_ETCB	LaunchEtcb at the time the guest was launched or imported. See *Section 3.9*.
260h	255:0	COMMITTED_ETCB	CommittedEtcb. See *Section 3.9*
280h–29Fh	–	–	Reserved. MBZ.
2A0h–49Fh	–	SIGNATURE	Signature of bytes 0h to 29Fh inclusive of this report. The format of the signature is described in *Appendix B*.

Table 28. Structure of the PLATFORM_INFO Field

Byte Offset	Bits	Name	Description
0h	63:8	–	Reserved.
	7	TIO_EN	Indicates that SEV-TIO is enabled.
	6	IOMMU_WRITE_SAFE	Indicates that platform has a hardware fix for CVE-2023-20585. Platforms without hardware fix use VerifyMitigation to indicate compliance.
	5	ALIAS_CHECK_COMPLETE	Indicates that alias detection has completed since the last system reset and there are no aliasing addresses. Resets to 0. Contains mitigation for CVE-2024-21944.
	4	CIPHERTEXT_HIDING_DRAM_EN	Indicates ciphertext hiding is enabled for the DRAM.
	3	RAPL_DIS	Indicates that the RAPL feature is disabled.
	2	ECC_EN	Indicates that the platform is using error correcting codes for memory. Present when EccMemReporting feature bit is set.
	1	TSME_EN	Indicates that TSME is enabled in the system.
	0	SMT_EN	Indicates that SMT is enabled in the system.

The firmware constructs an SNP_MSG_REPORT_RSP message containing the generated attestation report as defined in *Table 29*.

Table 29. SNP_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. 27h: Invalid key selection.
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 or its migration agent. The guest (or its migration agent) then sends the PSP a request message to export the guest's data. The format of this request is defined in *Table 30*.

Table 30. SNP_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 target guest to be exported.
	11:0	–	Reserved. Must be zero.
08h	31:1	–	Reserved. Must be zero.
	0	IMI_EN	Unsupported. Must be zero.
0Ch	31:0	–	Reserved. Must be zero.

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

The firmware checks that GCTX_PADDR is a Context page. The firmware checks that either:

- The guest sending the message is the migration agent of the exported guest—that is, the GCTX.MaReportId of the provided guest context page matches the ReportId of the requesting guest's context page, or
- The guest sending the message has no migration agent and is exporting itself—that is, the GCTX_PADDR matches the sPA of the requesting guest's context page, and GCTX.MaReportIdValid of the requesting guest is 0.

In summary, the guest can export itself if it has no migration agent. Otherwise, only its migration agent can export it. If either check fails, firmware returns an INVALID_GUEST status.

If the guest is exporting itself, the firmware checks that the guest message was encrypted with VMPCK0. That is, only VMPL0 can self-export. If not, firmware returns an INVALID_GUEST status.

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 an SNP_MSG_EXPORT_RSP message containing the guest context defined in *Table 31 below*. The size of the payload is such that HDR_SIZE + MSG_SIZE is 4096. That is, the message fills a 4 KB page.

Table 31. SNP_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 4h for this ABI version.
0Ch–1Fh		–	Reserved.
20h–39Fh		GCTX	Guest context. See *Table 32 below* for the format of this field.

If the exported guest supports the Secure TSC feature, the caller of this guest request should update the guest context before migration as follows:

PspTscOffset = PspTscOffset + (TSC/ GUEST_TSC_FREQ) * DesiredTscFreq

where the RDTSC instruction invocation and the GUEST_TSC_FREQ MSR read occurs within the guest that sent this message.

Table 32. GCTX Field Structure

Byte Offset	Bits	Name	Description
000h	383:0	LD	See *Section 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	

Byte Offset	Bits	Name	Description
110h	383:0	IDKeyDigest	
140h	383:0	AuthorKeyDigest	
170h	255:0	ReportID	
190h	383:0	IMD	Unsupported. Must be zero.
1C0h	63:0	MsgCount0	
1C8h	63:0	MsgCount1	
1D0h	63:0	MsgCount2	
1D8h	63:0	MsgCount3	
1E0h		RootMDEntry	See *Section 5.1* for description of this field. If IMI_EN is set, then this field is set to 0h.
220h	63:6	–	Reserved. Must be zero.
	5	CsvcekEn	See *Section 4.1* for description of this field.
	4	CsVcekPref	See *Section 4.1* for description of this field.
	3	–	Reserved
	2	VCEK_DIS	See *Section 4.1* for description of this field.
	1	IDBlockEn	See *Section 4.1* for description of this field.
	0	AuthorKeyEn	See *Section 4.1* for description of this field.
228h	63:0	Policy	See *Section 4.1* for description of this field.
230h	7:0	State	See *Section 4.1* for description of this field.
238h	63:0	OekIvCount	See *Section 4.1* for description of this field.
240h-29Fh		IDBlock	See *Section 8.1* for the description of this field and *Table 57* for the format of the field.
2A0h	127:0	GOSVW	See *Section 4.1* for description of this field.
2B0h	31:0	DesiredTscFreq	See *Section 4.1* for description of this field.
2B4h	31:0	–	Reserved.
2B8h	63:0	PspTscOffset	See *Section 4.1* for description of this field.
2C0h	63:0	LaunchTcb	The CurrentTcb at the time the guest was launched.
2C8h	63:0	LAUNCH_MIT_VECTOR	The mitigation vector value at the time the guest was launched (LaunchMitVector).
2D0h	15:0	–	Reserved, future use.
2D2h	111:0	–	Reserved, MBZ
2E0h	255:0	–	Reserved, future use.
300h	511:0	–	Reserved, future use.
340h	255:0	LaunchEtcb	The CurrentEtcb at the time the guest was launched.

Byte Offset	Bits	Name	Description
360h – 37Fh		–	Reserved. MBZ.

Note: *If the hypervisor relies on the VcekDis to migrate accurately, the hypervisor must trust the migration agent to not alter the VcekDis flag in the guest context.*

7.5 VM Import

Receiving a migrated guest from another system by the hypervisor with the help of the migration agent is deprecated, along with the corresponding SNP_MSG_IMPORT_REQ message.

The firmware returns INVALID_PARAM if this message is sent.

7.6 VM Absorb

An accelerated guest migration with an IMI is deprecated, along with the corresponding SNP_MSG_ABSORB_REQ message.

The firmware returns INVALID_PARAM if this message is sent.

7.7 VM Absorb – No Migration Agent

This message allows a guest to import its own guest context. This can be used with the SNP_MSG_EXPORT_REQ message to allow a guest to manage its migration without a migration agent.

If the imported guest supports the Secure TSC feature, the guest calling this guest message should update the guest context before importing as follows:

PspTscOffset = PspTscOffset – (TSC / GUEST_TSC_FREQ) * DesiredTscFreq

Where TSC is the timestamp counter read by the guest using RDTSC, GUEST_TSC_FREQ is the MSR (C001_0134) to retrieve the guest-effective TSC frequency, and DesiredTscFreq is the value stored in the guest's context page.

Table 33. SNP_MSG_ABSORB_NOMA_REQ Message Structure

Byte Offset	Bits	Name	Description
00h	63:0	–	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 4h for this ABI version.
10h	63:0	LAUNCH_MIT_VECTOR	The mitigation vector value. Bitmask may not set any

Byte Offset	Bits	Name	Description
			bits which are not set in the current value of SNP_VERIFY_MITIGATION's MIT_VERIFIED_VECTOR
18h–1Fh		–	Reserved.
20h–39Fh		IN_GCTX	Incoming guest context. See *Table 32* for the format of this field.

The firmware checks that GCTX.MaReportIdValid is 0—that is, the guest sending this message has no migration agent. If this check fails, the firmware returns the status INVALID_GUEST.

The firmware checks that the guest message was encrypted with VMPCK0. If not, the firmware returns the status of INVALID_GUEST.

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, firmware returns INVALID_MDATA_ENTRY.

The firmware overwrites the guest context at GCTX_PADDR with the guest context in the IN_GCTX field excluding the following fields which remain unaltered or conditionally altered (where noted).

- HostData
- IDKeyDigest
- AuthorKeyDigest
- ReportId
- AuthorKeyEn
- IDBlockEn
- VcekDis
- CsVcekEn
- CsVcekPref
- State
- IDBlock
- LaunchTcb
  - ***Note**: Cannot be greater than the actual LaunchTcb, but can be less.*
- LaunchMitVector
  - ***Note**: Can change CVEs from 1 to 0 but cannot change CVEs from 0 to 1.*
- VMRK
- LD

- Policy
- ASID

The firmware responds with a message containing the status of the import. The response message is defined in *Table 34* and it is encrypted using the same key that the guest used to encrypt the request.

Table 34. SNP_MSG_ABSORB_NOMA_RSP Message Structure

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

7.8 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 35*.

Table 35. Structure of the SNP_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.
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.MaReportIdValid of the guest is 1 and GCTX.MaReportId of the guest matches the ReportId in GCTX_PADDR of the migration agent; that is, the guest sending the SNP_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 36*.

Table 36. SNP_MSG_VMRK_RSP Message Structure

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

7.9 TSC Info

When a guest creates its own VMSA, it must query the PSP for information with the TSC_INFO message to determine the correct values to write into GUEST_TSC_SCALE and GUEST_TSC_OFFSET. The guest SNP_MSG_TSC_INFO_REQ request is described in *Table 37 below*.

Table 37. SNP_MSG_TSC_INFO_REQ Message Structure

Byte Offset	Bits	Name	Description
0h-7Fh		–	Reserved. Must be zero.

The firmware responds with the SNP_MSG_TSC_INFO_RSP response as described in *Table 38 below*.

Table 38. SNP_MSG_TSC_INFO_RSP Message Structure

Byte Offset	Bits	Name	Description
0h	31:0	STATUS	Status of the TSC_INFO message
4h	31:0	–	Reserved.
8h	63:0	GUEST_TSC_SCALE	Calculated as GCTX.DesiredTscFreq / (mean native frequency)
10h	63:0	GUEST_TSC_OFFSET	GCTX.PspTscOffset
18h	31:0	TSC_FACTOR	Encoding of the percentage decrease from nominal TSC frequency to mean TSC frequency due to clocking parameters. Mean TSC frequency can be calculated by the guest as: GUEST_TSC_FREQ * (1 – (TSC_FACTOR * 0.00001)) For instance, a TSC_FACTOR value of 200 indicates a reduction of 0.2% from nominal TSC frequency.
1Ch-7Fh		–	Reserved.

The guest should set the GUEST_TSC_SCALE and GUEST_TSC_OFFSET VMSA fields to the values provided by the PSP.

7.10 Get CSR

This message allows the guest to retrieve a self-signed CSR to replace the FMC_Alias certificate. The guest takes on responsibility for maintaining the resultant certificate. There are no inputs required for this message.

If the guest has been launched with the CSVCEK_EN flag set, then the firmware will construct the CSR and return the SNP_MSG_GET_CSR_RSP message. Otherwise, the firmware will return UNSUPPORTED.

Table 39. SNP_MSG_GET_CSR_RSP Message Structure

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

7.11 Get TPM Info

The SNP_MSG_GET_TPM_INFO_REQ guest request returns the TPM_INFO structure for the calling guest.

Table 40: Layout of the SNP_MSG_GET_TPM_INFO_REQ request message.

Byte Offset	Bits	Name	Description
0h	127:0	NONCE	The nonce to include in the TPM_INFO structure.

Table 41: Layout of the SNP_MSG_GET_TPM_INFO_RSP response message.

Byte Offset	Bits	Name	Description
0h	31:0	STATUS	The status of the guest request.
04h	660 bytes	TPM_INFO	The TPM_INFO structure.
298h	63:0	Reserved	Must be zero.

If the platform does not have a TPM to report, the command returns a TPM_INFO structure with the VALID bit cleared and the following fields zeroed.

If the platform does have a TPM, the firmware constructs a TPM_INFO object with the following

- The NONCE field is set to the NONCE value here.
- The GUEST field is set.
- The VALID bit is set.
- The REPORT_ID set to the REPORT_ID in of the calling guest.
- The EK_DIGEST is set to the collected EK digest.
- The NV_INDEX is set to the NV index used to retrieve the EK.
- The SIGNATURE is the signature over the TPM_INFO object.
- The SIGNING_KEY is set to indicate the key that signed this object.

The firmware returns an SNP_MSG_GET_TPM_INFO_RSP response message with STATUS set to SUCCES and the TPM_INFO field set to the constructed TPM_INFO object.

Chapter 8 Command Reference

8.1 DOWNLOAD_FIRMWARE

This command allows the hypervisor to install 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 DOWNLOAD_FIRMWARE_EX

This command replaces the current SEV-SNP firmware application with a new SEV-SNP application. This command extends the functionality of DOWNLOAD_FIRMWARE with support for provisional updates and for updates while SNP firmware is in the INIT state.

See *Section 3.3* for further information on live updates.

Note: *When SNP is in the UNINIT state and COMMIT is set to 1, this command behaves as if DOWNLOAD_FIRMWARE was called instead.*

8.2.1 Parameters

Table 42. Layout of the CMDBUF_SNP_DOWNLOAD_FIRMWARE_EX Structure

Byte Offset	Bits	In/Out	Name	Description
00h	31:0	In	LENGTH	Length of this command buffer in bytes.
04h	31:0	–	–	Reserved. Must be zero.
08h	63:0	In	FW_PADDR	System physical address of the region that contains an SEV-SNP firmware image. This region must be 32 B aligned.
10h	31:0	In	FW_LEN	Length of the SEV-SNP firmware in bytes.
14h	31:1	–	–	Reserved. Must be zero.
	0	In	COMMIT	Indicates that this command will automatically commit the newly installed image.

8.2.2 Actions

The SNP firmware may be in any state. If SEV is not in the UNINIT state, then the firmware returns INVALID_PLATFORM_STATE.

The firmware checks that the image is well formed and compatible with currently installed firmware within the PSP. This check is implementation specific and includes internal consistency checks and signature validation. If the provided image is not well formed, then the firmware returns INVALID_PARAM. If the image is compressed or encrypted, then the firmware returns UNSUPPORTED.

If the package length does not validate, then the firmware returns INVALID_LENGTH. If the signature of the image does not validate, then the firmware returns BAD_SIGNATURE.

If a version or SVN rollback is detected, then INVALID_CONFIG is returned. If FW_PADDR is not 32-byte aligned, then INVALID_ADDRESS is returned.

If the package provided is not an SEV binary, then INVALID_PARAM is returned. If the binary is encrypted or compressed and the platform does not support that feature for DownloadFirmware binaries, then INVALID_CONFIG is returned. The COMMIT parameter is used to have the firmware commit the provided image and set the CommittedTCB version to the provided image's FirmwareVersion. Otherwise, the SNP_COMMIT command may be called at a later time. This feature exists so the Hypervisor can try out the new firmware and revert back from the uncommitted version in case any issues arise.

If the FirmwareVersion of the provided firmware is greater than or equal to the FirmwareVersion of the current image, then this command is processing an upgrade. In this case, the provided image restricts the minimum version from which it will upgrade with its MinUpgradeFrom attribute. The firmware checks that MinUpgradeFrom of the provided image is less than or equal to the FirmwareVersion of the current firmware. If not, the firmware returns SHUTDOWN_REQUIRED.

When the firmware returns SHUTDOWN_REQUIRED status, the Host should send the SHUTDOWN or SHUTDOWN_EX commands prior to sending any other SEV commands. The firmware will continue to process commands, although commands may not execute properly. Following the subsequent SHUTDOWN/SHUTDOWN_EX, the Host could perform DOWNLOAD_FIRMWARE/DOWNLOAD_FIRMWARE_EX to load an alternate firmware image.

If the FirmwareVersion of the provided firmware is less than the FirmwareVersion of the current image, then this command is processing a downgrade. The current firmware will check that the FirmwareVersion of the provided image is equal to the CommittedVersion of the current firmware. If not, the firmware returns BAD_VERSION.

The firmware then installs the provided image, replacing the current firmware. If the SNP firmware is in the INIT state, all SNP firmware state is retained.

If a provided image is installed but the new firmware detects it cannot proceed safely, the firmware will automatically try to revert back to the CommittedVersion. If firmware reversion is successful, the firmware will return UPDATE_FAILED.

If firmware reversion is unsuccessful, then firmware will return HWERROR_UNSAFE. Following a return of HWERROR_UNSAFE, operation of the SEV firmware is indeterminate, and the recommendation is to reboot the platform.

If the firmware does not have the ability to automatically roll back to the CommittedVersion (per older Milan firmware versions), the firmware may return RESTORE_REQUIRED. After returning RESTORE_REQUIRE status, the firmware will only successfully execute DOWNLOAD_FIRMWARE_EX. Hypervisors should resolve this condition by rolling back to the CommittedVersion of the firmware by invoking DOWNLOAD_FIRMWARE_EX with the firmware image of the CommittedVersion. The firmware will continue to process commands, although command execution will be indeterminate and may not execute properly.

If COMMIT is 1 and the command successfully completes, the firmware implicitly commits the SVN and FirmwareVersion of the provided image as if SNP_COMMIT was called.

Until the new image has been committed, the Hypervisor will have the ability to revert back to the CommittedVersion by invoking DOWNLOAD_FIRMWARE_EX again with the committed image. Note that in order to perform a firmware upgrade, the current firmware must be successfully committed.

8.2.3 Status Codes

Table 43. Status Codes for DOWNLOAD_FW_EX

Status	Condition
SUCCESS	Successful completion.
RESTORE_REQUIRED	One of the following has occurred: • New firmware image is installed but unusable. • Update of the firmware internal state has failed, and the firmware has not successfully reverted to the COMMITTED_VERSION. For older firmware/platforms (Milan), the firmware may return RESTORE_REQUIRED instead of HWERROR_UNSAFE.
INVALID_PARAM	Provided image is not well formed.
SHUTDOWN_REQUIRED	Provided image cannot be live updated.
BAD_VERSION	Provided image is less than CommittedVersion.
INVALID_PLATFORM_STATE	The SEV state is not UNINIT.
INVALID_ADDRESS	The address is invalid for use by the firmware (not aligned).
UNSUPPORTED	A required feature is not supported.

Status	Condition
INVALID_CONFIG	The system is not in a valid configuration that can support SNP.
BAD_SIGNATURE	Incorrect signature provided.
UPDATE_FAILED	Update of the firmware internal state has failed, and the firmware has successfully reverted to the COMMITTED_VERSION.
HWERROR_UNSAFE	Update of the firmware internal state has failed, and the firmware has not successfully reverted to the COMMITTED_VERSION. For older firmware/platforms (Milan), the firmware may return RESTORE_REQUIRED instead of HWERROR_UNSAFE.

8.3 SNP_COMMIT

This command commits the currently installed firmware. Once committed, the firmware cannot be replaced with a previous firmware version or SVN.

See *Section 3.3* for further information on live updates.

8.3.1 Actions

The firmware sets the CommittedTcb to the CurrentTcb of the current firmware.

The firmware sets the CommittedVersion to the FirmwareVersion of the current firmware.

The firmware sets the ReportedTcb to the CurrentTcb.

The firmware deletes the VLEK hashstick if ReportedTcb changed.

8.3.2 Status Codes

Table 44. Status Codes for SNP_COMMIT

Status	Condition
SUCCESS	Successful completion.

8.4 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 [KDS-VCEK], which expects the output of this command to be provided in the HWID URL parameter. 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 Firmware or Default pages. Otherwise, the firmware returns INVALID_PAGE_STATE.

8.5 SNP_PLATFORM_STATUS

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

If CSVCEK_EN is supported, then the number of memory pages required for the certificate chain can be obtained using the PLATFORM_STATUS. In addition, the number of pages required to store the CoRIM ID list (see *Section 8.40*) is also provided.

8.5.1 Parameters

Table 45. Layout of the CMDBUF_SNP_PLATFORM_STATUS Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:0	In	STATUS_PADDR	sPA to write the platform status structure. See *Table 46 below*.

8.5.2 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 46. 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 *Section 3.2* for encodings.
03h	7:4	–	Reserved.
	3	IS_TIO_INIT	Indicates TIO_INIT has succeeded and TIO has

Byte Offset	Bits	Name	Description
			been initialized in the firmware. Cleared during SNP_SHUTDOWN_EX (IOMMU=1). Present if SevTio feature bit is set.
	2	IOMMU_W_SAFE	Indicates that the IOMMU Write Buffer Vulnerability is resolved. Resets to 0. Contains mitigation for CVE-2023-20585.
	1	ALIAS_CHECK_COMPLETE	Indicates that alias detection has completed since the last system reset and there are no aliasing addresses. Resets to 0. Contains mitigation for CVE-2024-21944.
	0	IS_RMP_INIT	Set to the value of IsRmpInitialized.
04h	31:0	BUILD	Firmware build ID for this API version.
08h	31:21	–	Reserved.
	20:17	NUM_CORIM_ID_PAGES	Number of 4 kB memory pages required for CoRIM ID list
	16:9	NUM_CERT_PAGES_P384	Number of 4 kB memory pages required for chip secret certificate chain. ***Note**: the number of pages returned is based on the maximum size of a guest message response, which is limited to one 4-kB page and always includes a 4-byte STATUS.*
	8	IS_VIOMMU_EN	Indicates VIOMMU is enabled as set by SNP_INIT_EX[VIOMMU_EN]. Cleared during SNP_SHUTDOWN_EX (x86=1). Present if VIOMMU feature bit is set.
	7	IS_TIO_EN	Indicates TIO is enabled as set by SNP_INIT_EX[TIO_EN]. Cleared during SNP_SHUTDOWN_EX (x86=1). Present if SevTio feature bit is set.
	6	CIPHERTEXT_HIDING_DRAM_EN	Indicates ciphertext hiding is enabled for the DRAM. Present if CipherTextHidingDRAM feature bit is set.
	5	CIPHERTEXT_HIDING_DRAM_CAP	Indicates platform capable of ciphertext hiding for the DRAM. Present if CipherTextHidingDRAM feature bit is set.
	4	RAPL_DIS	Indicates that the RAPL is disabled. Present if RaplDis feature bit is set.
	3	FEATURE_INFO	Indicates that the SNP_FEATURE_INFO command

Byte Offset	Bits	Name	Description
			is available.
	2	VLEK_EN	Indicates whether a VLEK hashstick is loaded.
	1	MASK_CHIP_KEY	Set to the value of MaskChipKey.
	0	MASK_CHIP_ID	Set to the value of MaskChipId.
0Ch	31:0	GUEST_COUNT	The number of guests currently managed by the firmware.
10h	63:0	CURRENT_TCB	The CurrentTcb of the firmware.
18h	63:0	REPORTED_TCB	The reported TCB version in guest attestation reports.

8.5.3 Status Codes

Table 47. 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.

8.6 SNP_PLATFORM_STATUS_EX

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

If CSVCEK_EN is supported, then the number of memory pages required for the certificate chain can be obtained using the PLATFORM_STATUS. In addition, the number of pages required to store the CoRIM ID list (see *Section 8.40*) is also provided.

8.6.1 Parameters

Table 48. Layout of the CMDBUF_SNP_PLATFORM_STATUS_EX Structure

Byte Offset	Bits	In/Out	Name	Description
00h	31:0	In	VERSION	Command version
04h	31:0	–	–	Reserved. Must be zero.
08h	63:0	In	STATUS_PADDR	sPA to write the platform status structure. See *Table 46*.

8.6.2 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 49. Layout of the STRUCT_PLATFORM_STATUS_EX 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 *Section 3.2* for encodings.
03h	7:4	–	Reserved.
	3	IS_TIO_INIT	Indicates TIO has been initialized in the firmware. Present if SevTio feature bit is set.
	2	–	Reserved.
	1	ALIAS_CHECK_COMPLETE	Indicates that alias detection has been completed since the last system reset and there are no aliasing addresses. Resets to 0. Contains mitigation for CVE-2024-21944.
	0	IS_RMP_INIT	Set to the value of IsRmpInitializied.
04h	31:0	BUILD	Firmware build ID for this API version.

Byte Offset	Bits	Name	Description
08h	31:21	–	Reserved.
	20:17	NUM_CORIM_ID_PAGES	Number of 4 kB memory pages required for CoRIM ID list
	16:9	NUM_CERT_PAGES_P384	Number of 4 kB memory pages required for chip secret certificate chain. ***Note**: the number of pages returned is based on the maximum size of a guest message response, which is limited to one 4-kB page and always includes a 4-byte STATUS.*
	8	IS_VIOMMU_EN	Indicates VIOMMU is enabled. Present if VIOMMU feature bit is set.
	7	IS_TIO_EN	Indicates TIO is enabled. Present if SevTio feature bit is set.
	6	CIPHERTEXT_HIDING_DRAM_EN	Indicates ciphertext hiding is enabled for the DRAM. Present if CipherTextHidingDRAM feature bit is set.
	5	CIPHERTEXT_HIDING_DRAM_CAP	Indicates platform capable of ciphertext hiding for the DRAM. Present if CipherTextHidingDRAM feature bit is set.
	4	RAPL_DIS	Indicates that the RAPL is disabled. Present if RaplDis feature bit is set.
	3	FEATURE_INFO	Indicates that the SNP_FEATURE_INFO command is available.
	2	VLEK_EN	Indicates whether a VLEK hashstick is loaded.
	1	MASK_CHIP_KEY	Set to the value of MaskChipKey.
	0	MASK_CHIP_ID	Set to the value of MaskChipId.
0Ch	31:0	GUEST_COUNT	The number of guests currently managed by the firmware.
10h	63:0	CURRENT_TCB	The CurrentTcb of the firmware.
18h	63:0	REPORTED_TCB	The reported TCB version in guest attestation reports.
20h	255:0	CURRENT_ETCB	The CurrentEtcb of firmware.
40h	255:0	REPORTED_ETCB	The reported ETCB version in guest attestation reports.
60h	511:0	–	Reserved. Mbz.

8.6.3 Status Codes

Table 50. Status Codes for SNP_PLATFORM_STATUS_EX

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.

8.7 SNP_CONFIG

This command sets the systemwide configuration values for SNP.

8.7.1 Parameters

Table 51. Layout of the CMDBUF_SNP_CONFIG_STATUS Structure

Byte Offset	Bits	In/Out	Name	Description
00h	63:0	In	REPORTED_TCB	The TCB_VERSION to report in guest attestation reports.
08h	31:2	–	–	Reserved. Must be zero.
	1	In	MASK_CHIP_KEY	Indicates that the VCEK is not used in attestation and guest key derivation.
	0	In	MASK_CHIP_ID	Indicates that the CHIP_ID field in the attestation report will always be zero.
0Ch	31:0		–	Reserved. Must be zero.
10h	128:0		–	Reserved. Must be zero.
20h-3Fh	255:0	In	REPORTED_ETCB	The ETCB_VERSION to report in guest attestation reports.

8.7.2 Actions

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

The firmware checks that the REPORTED_ETCB parameter is less than or equal to CommittedEtcb. If not, the firmware returns INVALID_PARAM.

If REPORTED_TCB is 0, the firmware sets ReportedTcb to CommittedTcb. Otherwise, the firmware sets ReportedTcb value to REPORTED_TCB. If the ReportedTcb changes, the firmware deletes the VLEK hashstick.

If REPORTED_ETCB is 0, the firmware sets ReportedEtcb to CommittedEtcb. Otherwise, the firmware sets ReportedEtcb value to REPORTED_ETCB.

The firmware sets the systemwide MaskChipId to MASK_CHIP_ID.

The firmware sets MaskChipKey to MASK_CHIP_KEY. This bit was introduced in version 1.53.

8.7.3 Status Codes

Table 52. Status Codes for SNP_CONFIG_STATUS

Status	Condition
SUCCESS	Successful completion.
INVALID_PARAM	The desired reported TCB_VERSION or ETCB_VERSION are invalid, or MBZ fields are not zero.

8.8 SNP_INIT

This command validates the platform configuration of the SNP and initializes the firmware. This command is a specialization of the SNP_INIT_EX command.

Deprecated in version 1.52: In version 1.52 and later, SNP_INIT_EX should be used instead of SNP_INIT. It is important to ensure that UEFI reserved pages are marked HV-fixed to ensure the stability of the system.

8.8.1 Parameters

None.

8.8.2 Actions

This command behaves as if SNP_INIT_EX was called with INIT_RMP set to 1 and all other parameters set to zero.

8.8.3 Status Codes

See SNP_INIT_EX below.

8.9 SNP_INIT_EX

This command validates the platform configuration of the SNP and initializes the firmware. Hypervisors should not schedule any guests during the execution of this command.

During the execution of this command, the firmware configures and enables SNP security policy enforcement in many system components. Some system components write to regions of memory

reserved by early x86 firmware (e.g., UEFI). Other system components write to regions provided by the operation system, hypervisor, or x86 firmware. Such system components can write only to HV-fixed or Default pages. An error will result when they attempt to write to other page states after SNP_INIT enables their SNP enforcement.

Starting in version 1.52, this command takes a list of sPA ranges to convert into HV-fixed page states during the RMP initialization. If INIT_RMP is 1, a hypervisor should provide all sPA ranges that it will never assign to a guest until the next RMP reinitialization. For instance, the memory that UEFI reserves should be included in the range list. This allows system components that occasionally write to memory (e.g., logging to UEFI-reserved regions) to not fail due to RMP initialization and SNP enablement.

Some processors support the Running Average Power Limit (RAPL) feature which provides information about power utilization of software. RAPL can be disabled using the RAPL_DIS flag in SNP_INIT_EX to disable RAPL while SNP firmware is in the INIT state. Guests may require that RAPL is disabled by using the POLICY.RAPL_DIS guest policy flag. RAPL disable is supported when RaplDis (bit 2) in Fn8000_0024_ECX_x00 is 1.

Ciphertext hiding for DRAM prevents host accesses from reading the ciphertext of SNP guest private memory. Instead of reading ciphertext, the host will see constant default values. Ciphertext hiding separates the ASID space into SNP guest ASIDs and host ASIDs. All SNP active guests must have an ASID less than or equal to MAX_SNP_ASID provided to the SNP_INIT_EX command. All SEV-legacy guests must be greater than MAX_SNP_ASID. If MAX_SNP_ASID is 0 then firmware treats it as meaning that all ASIDs between 1 and MIN_SEV_ASID -1 are allocated for SEV-SNP. This feature is present when CipherTextHidingDRAM (bit 3) in Fn8000_0024_ECX_x00 is 1.

A platform is capable of ciphertext hiding if DDR-BF (bounded fault) mode is configured. If not, the SNP_INIT_EX command will fail if ciphertext hiding is requested.

SNP_INIT_EX will fail if RmpInstallPending or RstInstallPending are set.

Hypervisors must not assign the VM_HSAVE pages as HV-fixed. Otherwise, VMRUN will fail.

8.9.1 Parameters

Table 53. Layout of the CMDBUF_SNP_INIT_EX Structure

Byte Offset	Bits	In/Out	Name	Description
00h	31:6	–	–	Reserved. Must be zero.
	5	In	VIOMMU_EN	0: Do not enable vIOMMU support. 1: Enable vIOMMU support.
	4	In	TIO_EN	0: Do not enable TIO support. 1: Enable TIO support.
	3	In	CIPHERTEXT_HIDING_DRAM_EN	0: Ciphertext hiding for the DRAM is disabled. 1: Ciphertext hiding for the DRAM is enabled. Present when CipertextHiding feature bit is set.
	2	In	RAPL_DIS	0: RAPL enabled. 1: RAPL disabled. Present when RaplDis feature bit is set.
	1	In	LIST_PADDR_EN	0: LIST_PADDR is not valid. 1: LIST_PADDR is valid.
	0	In	INIT_RMP	Indicates that the RMP should be initialized.
04h	31:0	–	–	Reserved. Must be zero.
08h	63:0	In	LIST_PADDR	sPA of the RANGE_LIST structure. See *Table 113* for SNP_PAGE_SET_STATE.
10h	15:0	In	MAX_SNP_ASID	The maximum ASID usable for a SNP guest. Valid only when CIPHERTEXT_HIDING_DRAM_EN is 1 or SevEsSnpConcurrent is 1.
12h–3Fh	-	-	-	Reserved. Must be zero.

8.9.2 Actions

Before invoking SNP_INIT_EX with INIT_RMP set to 1, 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 checks for CXL devices in the system. If there are Type 1 or Type 2 CXL devices present in the system, the firmware returns INVALID_CONFIG.

If INIT_RMP is 0, then the firmware determines if SNP can be initialized securely without initializing the RMP table. The firmware requires initialization if the RMP is not yet initialized. The firmware may also require initialization for other reasons, such as if the RMP was incompatibly initialized by a previous version of the firmware. If the firmware determines the RMP requires initialization, the firmware returns RMP_INIT_REQUIRED.

If INIT_RMP is 1, then the firmware ensures the following system requirements are met:

- SYS_CFG[MemoryEncryptionModEn] must be set to 1 across all cores. (SEV must be enabled.)
- SYS_CFG[SecureNestedPagingEn] must be set to 1 across all cores.
- SYS_CFG[VMPLEn] must be set to 1 across all cores.
- SYS_CFG[MFDM] must be set to 1 across all cores.
- VM_HSAVE_PA (MSR C001_0117) must be set to 0h across all cores.
- HWCR[SmmLock] (MSR C001_0015) must be set to 1 across all cores.

The following MSRs must be set identically across all cores:

- All MTRRs
- IORR_BASE
- IORR_MASK
- TOM
- TOM2

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

If INIT_RMP is 1, then 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 initializes the IOMMU to perform RMP enforcement. The firmware also transitions the event log, PPR log, and completion wait buffers of the IOMMU to an RMP page state that is read only to the hypervisor and cannot be assigned to guests.

If LIST_PADDR_EN is 1, then the firmware performs the following checks:

- If INIT_RMP is 0, the firmware returns INVALID_PARAM.
- If LIST_PADDR is an invalid sPA, the firmware returns INVALID_ADDRESS.

- If a range in RANGES contains an invalid address, the firmware returns INVALID_ADDRESS. For this check, a range overlapping the RMP is not considered an invalid address.

If RAPL_DIS is 1 but the power management controller does not support RAPL disable, the firmware returns INVALID_CONFIG. Otherwise, if RAPL_DIS is 1, the firmware disables the RAPL feature for the entire system. RAPL (Running Average Power Limit) is an x86 feature where runtime measurements of power usage (memory or CPUs) can be found in designated MSRs.

If CIPHERTEXT_HIDING_DRAM_EN is 1, the firmware checks that the platform is capable of ciphertext hiding. If not, the firmware returns INVALID_CONFIG.

If CIPHERTEXT_HIDING_DRAM_EN is 1, the firmware checks for CXL memory in the system. If there is CXL memory present in the system, the firmware returns INVALID_CONFIG.

Otherwise, the firmware enables Ciphertext hiding and sets the maximum SNP ASID to be MAX_SNP_ASID.

The firmware checks that the platform can run SEV-ES and SEV-SNP guests at the same time (bit 17 of Fn8000_0024_ECX_x00 is present). If it is, the firmware sets the maximum SNP ASID to be MAX_SNP_ASID. If MAX_SNP_ASID is 0 then firmware treats it as meaning that all ASIDs between 1 and MIN_SEV_ASID -1 are allocated for SEV-SNP.

If INIT_RMP is 1, then the firmware alters the RMP such that pages of the RMP are in the Firmware state and all other pages covered by the RMP are in the Hypervisor state. If LIST_PADDR_EN is 1, then the firmware initializes the provided ranges in LIST_PADDR as HV-fixed pages. 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 sets MaskChipKey to 0.

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

The firmware sets the platform state to INIT.

8.9.3 Status Codes

Table 54. Status Codes for SNP_INIT_EX

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.

RMP_INIT_REQUIRED	Initialization of the RMP is required.
EXPAND_BUFFER_LENGTH_REQUEST	Indicates that a firmware write buffer is not large enough. Indicates that the firmware is asking to increase the data buffer size. The new buffer size is specified in the BUFFER_CAPACITY field of BUFFER structure.

8.10 SNP_GCTX_CREATE

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

8.10.1 Parameters

Table 55. 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.

8.10.2 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 56 below*. All other fields within the guest context remain indeterminate until they are initialized through the launch process or through the import process.

Table 56. 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.

Field	Value
OekIvCount	0h.
LaunchTcb	Set to CommittedTcbS
LaunchMitVector	Set to CurrentMitVector
LastAccessVersion	Set to CurrentVersion.

8.10.3 Status Codes

Table 57. 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.

8.11 SNP_ACTIVATE

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

8.11.1 Parameters

Table 58. 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.

8.11.2 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, then the firmware returns INVALID_ASID.

The firmware checks the ASID to be within the range 1h to (MIN_SEV_ASID-1), inclusive. The MIN_SEV_ASID value is discovered by CPUID Fn8000_001F[EDX]. Further, if ciphertext hiding is enabled or concurrent SEV-ES and SEV-SNP guests are allowed (bit 17 of Fn8000_0024_ECX_x00 is present), the firmware checks that the ASID is within the range 1h to MAX_SNP_ASID, inclusive. If not, the firmware returns INVALID_ASID. If MAX_SNP_ASID is 0 then firmware treats it as meaning that all ASIDs between 1 and MIN_SEV_ASID -1 are allocated for SEV-SNP. 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 DF_FLUSH_REQUIRED.

Note: *All ASIDs are marked to require a DF_FLUSH at reset.*

The firmware checks that no pages are assigned to the ASID in the RMP. If pages are assigned, the firmware returns INVALID_CONFIG.

If POLICY.SINGLE_SOCKET is 1 and the system has more than one socket populated, the firmware returns POLICY_FAILURE. The firmware installs the guest's VEK into the memory controllers in the key slot associated with the given ASID.

8.11.3 Status Codes

Table 59. 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.

Status	Condition
ASID_OWNED	The ASID is already owned by another guest.
POLICY_FAILURE	The guest policy prevents activation on multiple sockets.
UPDATE_FAILED	Update of the firmware internal state or a guest context page has failed.
ACTIVE	The guest is already activated.
DF_FLUSH_REQUIRED	DF_FLUSH was not invoked before this command.

8.12 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 reused, WBINVD needs be done only on core complexes associated with the guest.

8.12.1 Parameters

Table 60. 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.

8.12.2 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 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. Further, if ciphertext hiding is supported or concurrent SEV-ES and SEV-SNP guests are allowed (bit 17 of Fn8000_0024_ECX_x00 is present), the firmware checks that the ASID is within the range 1h to MAX_SNP_ASID, inclusive. If not, the firmware returns INVALID_ASID. If MAX_SNP_ASID is 0 then firmware treats it as meaning that all ASIDs between 1 and MIN_SEV_ASID -1 are allocated for SEV-SNP. 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 one is needed, the firmware returns DFLUSH_REQUIRED.

Note: *All ASIDs are marked to require a DF_FLUSH at reset.*

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

If POLICY.SINGLE_SOCKET is 1, the firmware performs the following checks:

- If the guest is bound to a migration agent, the migration agent must already be activated. Also, completing SNP_ACTIVATE_EX must not result in activating the guest on a different socket than its migration agent.
- Completing SNP_ACTIVATE_EX will not result in activating the guest on multiple sockets.

If any of the checks fail, the firmware returns POLICY_FAILURE. Otherwise, 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 to expand the set of CCXs on which the guest may execute.

8.12.3 Status Codes

Table 61. Status Codes for SNP_ACTIVATE_EX

Status	Condition
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.
POLICY_FAILURE	The guest policy prevents activation on multiple sockets.
UPDATE_FAILED	Update of the firmware internal state or a guest context page has failed.
ACTIVE	The guest is already activated.
DF_FLUSH_REQUIRED	DF_FLUSH was not invoked before this command.
SUCCESS	Successful completion.

8.13 SNP_DECOMMISSION

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

8.13.1 Parameters

Table 62. 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.

8.13.2 Actions

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

The firmware checks that 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 checks that all metadata pages have been reclaimed. If not, the firmware returns INVALID_GUEST_STATE.

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.

8.13.3 Status Codes

Table 63. 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.
UPDATE_FAILED	Update of the firmware internal state or a guest context page has failed.

8.14 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 guests are no longer in any CPU caches.

8.14.1 Parameters

None.

8.14.2 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 SNP_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.

8.14.3 Status Codes

Table 64. Status Codes for SNP_DF_FLUSH

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

8.15 SNP_SHUTDOWN

This command returns the firmware to an uninitialized state.

8.15.1 Parameters

None.

8.15.2 Actions

This command is equivalent to executing SNP_SHUTDOWN_EX with a command buffer containing zeroes.

8.15.3 Status Codes

Table 65. Status Codes for SNP_SHUTDOWN

Status	Condition
DF_FLUSH_REQUIRED	DF_FLUSH was not invoked before this command.
SUCCESS	Successful completion.

8.16 SNP_SHUTDOWN_EX

This command returns the firmware to an uninitialized state and optionally disables the SNP enforcement in the IOMMU and sets the associated pages to the Hypervisor state.

8.16.1 Parameters

Table 66. Layout of the CMDBUF_SNP_SHUTDOWN_EX Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length of this command buffer in bytes.
4h	31:2	–	–	Reserved. Must be zero.
	1	In	X86_SNP_SHUTDOWN	Disables SNP on all cores by clearing the SYS_CFG[SNPEn] bit. Present when X86SnpShutdown feature bit is 1.
	0	In	IOMMU_SNP_SHUTDOWN	Disable enforcement of SNP in the IOMMU.

8.16.2 Actions

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

If IOMMU_SNP_SHUTDOWN is set to 1, the firmware performs the following actions:

- Disables SNP enforcement by the IOMMU.
- Transitions all pages associated with the IOMMU to the Reclaim state. Firmware before version 1.53 transitions to the Hypervisor state. Starting with version 1.53, the hypervisor must execute RMPUPDATE to recover the page by transitioning it from Reclaim.
- Records that a full RMP reinitialization is required by the next SNP_INIT invocation.

If IOMMU_SNP_SHUTDOWN is 0, the firmware leaves the IOMMU and its pages unaltered.

If X86_SNP_SHUTDOWN is set to 1, the firmware clears the SYS_CFG[SNPEn] bit in each core. Software must set IOMMU_SNP_SHUTDOWN to 1 if X86_SNP_SHUTDOWN is 1.

The firmware reenables the RAPL feature if it was disabled in SNP_INIT_EX.

If the SNP firmware is in the INIT state, the firmware checks every encryption-capable ASID to verify that it is not in use by a guest, and a DF_FLUSH is not required. If there is an active SNP guest, the firmware returns ACTIVE. If a DF_FLUSH is required, the firmware returns DF_FLUSH_REQUIRED.

The firmware clears the encryption keys from the memory controller and transitions the platform to the UNINIT state and returns SUCCESS.

Note: *Aside from the IOMMU pages referenced above, 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 afterward to reset the RMP.*

8.16.3 Status Codes

Table 67. Status Codes for SNP_SHUTDOWN_EX

Status	Condition
INVALID_PLATFORM_STATE	SEV is not in the UNINIT state.
DF_FLUSH_REQUIRED	DF_FLUSH was not invoked before this command.
ACTIVE	An SNP guest is active.
SUCCESS	Successful completion.

8.17 SNP_LAUNCH_START

This command initializes the flow to launch a guest. The command will be modified with a new flag to select the DICE derived VCEK for the target guest.

8.17.1 Parameters

Table 68. 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 11* 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	31:4	–	–	Reserved. Must be zero.
	3	In	CSVCEK_EN	0: Legacy VCEK used for guest 1: Chip secret VCEK enabled for guest
	2	In	CSVCEK_PREF	0: Legacy VCEK preferred for guest report signing 1: Chip secret VCEK preferred for guest report signing CSVCEK_EN must be set to 1 for this option to be selected.
	1	–	~~MI_EN~~	Must be zero.
	0	In	MA_EN	1 if this guest is associated with a migration agent. Otherwise, 0.
1Ch	31:0	In	DESIRED_TSC_FREQ	Hypervisor-desired means TSC frequency in kHz of the guest. This field has no effect if guests do not enable Secure TSC in the VMSA. The hypervisor should set this field to 0h if it does not support Secure TSC for this guest.
20h	127:0	In	GOSVW	Hypervisor-provided value to indicate guest OS-visible workarounds. The format is hypervisor defined.

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 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 page. 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.

If the caller attempts to launch the command with a non-zero unsupported IMI_EN parameter, the firmware returns INVALID_PARAM.

If the platform does not support migration agents and MA_EN is set, the firmware returns INVALID_PARAM.

FEATURE_INFO must indicate chip secret VCEK is supported on the platform (see *Section 8.38*) if CSVCEK_EN is set to '1', else firmware will return INVALID_PARAM. If CSVCEK_EN is set to '0' and CSVCEK_PREF is set to '1', then firmware will return INVALID_PARAM.

SNP firmware will select the appropriate VCEK derivation based on the value of CSVCEK_EN and CSVCEK_PREF on platforms that enable chip secret VCEK derivation as follows:

- MSG_KEY_REQ will make use of the chip secret VCEK if CSVCEK_EN=1. Otherwise, MSG_KEY_REQ will use the legacy VCEK.

The guest context will also reflect this selection with the entries CsVcekEn and CsVcekPref.

The firmware verifies 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.SMT is 0, then SMT must be disabled.
- POLICY.ABI_MAJOR must equal the major version of this ABI.
- POLICY.ABI_MINOR must be less than or equal to the minor version of this ABI.
- If POLICY.SINGLE_SOCKET is 1 and MA_EN is 1, then the migration agent's POLICY.SINGLE_SOCKET must be 1.
- If POLICY.CXL_ALLOW is 0, then CXL channels must not have memory or devices populated. This policy check is performed when CxlAllowPolicy feature bit is set.

- If POLICY.MEM_AES_256_XTS is 1, then the memory controller must be configured to use AES-256 XTS. This policy check is performed when Aes256XtsPolicy feature bit is set.
- If POLICY.RAPL_DIS is 1, then the Running Average Power Limit (RAPL) feature must be disabled. This policy check is performed when RaplDis feature bit is set.
- If POLICY.CIPHERTEXT_HIDING_DRAM is 1, then ciphertext hiding must be enabled. This policy check is performed when CiphertextHidingDRAM feature bit is set.
- If POLICY. PAGE_SWAP_DISABLE is 1, then page swap disable feature must be enabled. This policy check is performed when SnpPageSwapDisablePolicy feature bit is set.

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 69*.

Table 69. Guest Context Field Initialization for the Launch Flow

Field	Value
MsgCount0 MsgCount1 MsgCount2 MsgCount3	0h
Policy	Set to POLICY.
MaReportId	Set to ReportId of MA if MA_EN is 1. Set to PADDR_INVALID otherwise.
MaReportIdValid	Set to the value of MA_EN.
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 *Section 7.8*.
LD	0h
IMD	Unsupported. Set to 0h.
IDBlockEn	0
IDBlock	0h
IDKeyDigest	0h
AuthorKeyEn	0
AuthorKeyDigest	0h
ReportID	Generated using a CSRNG.
IMIEn	Unsupported. Must be zero.
GOSVW	GOSVW field.

Field	Value
DesiredTscFreq	Set to DESIRED_TSC_FREQ.
PspTscOffset	0h

The firmware sets the guest state to GSTATE_LAUNCH.

8.17.3 Status Codes

Table 70. 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.
UPDATE_FAILED	Update of the firmware internal state or a guest context page has failed.

8.18 SNP_LAUNCH_UPDATE

This command inserts pages into the guest physical address space.

8.18.1 Parameters

Table 71. 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	–	~~IMI_PAGE~~	Must be zero.
	3:1	In	PAGE_TYPE	Encoded page type. See *Table 72*.
	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.

Byte Offset	Bits	In/Out	Name	Description
				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 73* for the definition of the mask.
	23:16	In	VMPL2_PERMS	VMPL permission mask for VMPL2. See *Table 73* for the definition of the mask.
	15:8	In	VMPL1_PERMS	VMPL permission mask for VMPL1. See *Table 73* for the definition of the mask.
	7:0	–	–	Reserved. Must be zero.

Table 72. 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.
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 73. VMPL Permission Mask

Bit	Field	Description
7:5	–	Reserved. Must be zero.
4	Supervisor-Shadow-Stack	Guest supervisor shadow stack page access is allowed.
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.

8.18.2 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 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 IMI_PAGE is 0. If not, the firmware returns INVALID_PARAM. The firmware unconditionally sets GCTX.IMIEn to zero.

The following VMPL permissions settings combinations are prohibited and will result in firmware returning INVALID_PARAM:

- Bit 0 is set to 0, but any other mask permission bit is set to 1. Non-readable pages are not supported.
- Bit 4 is set to 1, and either Bit 2 or Bit 3 are set to 1. Executable supervisor shadow stack pages are not supported.
- Bit 3 set to 1, and Bit 2 set to 0. Supervisor execute but not user-execute pages are not supported.

The firmware updates the GCTX.LD 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 74*.

Table 74. 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	Unsupported. Must be zero.
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	–	0h
68h	63:0	GPA	The 64-bit gPA of the region.

The firmware unconditionally updates GCTX.LD.

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

***Note**: The guest physical address space is limited according to CPUID Fn80000008_EAX and thus the GPAs used by the firmware in measurement calculation are equally limited. Hypervisors should not attempt to map pages outside of this limit.*

The following subsections describe 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.

8.18.2.1 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 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.

8.18.2.2 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 ignores the values of GUEST_TSC_SCALE and GUEST_TSC_OFFSET and measures the VMSA as if those fields contained zero.

The firmware updates GCTX.LD as described above.

If VmsaRegProt in the SEV_FEATURES field of VMSA is 1 and the current microcode level supports VmsaRegProt, then the firmware generates an 8 B random tweak value and writes it to offset 300h of the VMSA. The firmware then XORs the tweaked quadwords of the VMSA with the tweak value. The quadwords of the VMSA that are tweaked are determined by the family, model, stepping, and microcode patch of the processor. This information is shared with the guest via the PAGE_TYPE_SECRETS page. If the current microcode level does not support VmsaRegProt, the firmware returns UNSUPPORTED.

Note: *The firmware measures the VMSA provided by the hypervisor prior to any tweak operations.*

If SecureTsc in the SEV_FEATURES field of VMSA is 1, the firmware sets the GUEST_TSC_SCALE and GUEST_TSC_OFFSET fields in the VMSA as follows:

```
GUEST_TSC_SCALE := GCTX.DesiredTscFreq / (mean native frequency)
GUEST_TSC_OFFSET := 0
```

Note: *These VMSA fields are changed after the measurement is calculated.*

If SecureTsc in the SEV_FEATURES field of VMSA is 0, then the firmware does not alter GUEST_TSC_SCALE or GUEST_TSC_OFFSET.

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

8.18.2.3 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 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.

8.18.2.4 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 as described above.

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

8.18.2.5 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 as described above.

The firmware constructs the 4 KB data structure described in *Table 75*. 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 75. 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 5h.
004h	31:1	–	Reserved.
	0	IMI_EN	Unsupported. Must be zero.
008h	31:0	FMS	Family, model, and stepping information as reported in CPUID Fn0000_0001_EAX.
0Ch	31:0	–	Reserved.
10h	127:0	GOSVW	GOSVW guest context field as provided by the hypervisor in SNP_LAUNCH_START.
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–0FFh		–	Reserved for guest OS usage.
100h–13Fh		VMSA_TWEAK_BITMAP	Set to the bitmap of the VMSA tweak. The *k*th bit of the bitmap indicates that the *k*th quadword of the VMSA is tweaked.
140h–15Fh		–	Reserved for guest OS usage.
160h	31:0	TSC_FACTOR	Encoding of the percentage decrease in mean TSC frequency due to clocking parameters. Real TSC frequency can be calculated by the guest as: GUEST_TSC_FREQ * (1 – (TSC_FACTOR * 0.00001)) For instance, a TSC_FACTOR value of 200 indicates a

Byte Offset	Bits	Name	Description
			reduction of 0.2% of TSC frequency.
164h	31:0	–	Reserved.
168h	63:0	LAUNCH_MIT_VECTOR	Set to the launch mitigation vector value (LaunchMitVector)
170h	63:0	LAUNCH_TCB_VERSION	Seto to launch TCB Version (LaunchTcb)
178h	63:0	–	Reserved.
180h	255:0	LAUNCH_ETCB_VERSION	Set to launch Extended TCB Version (LaunchEtcb)
1A0h–FFFh		–	Reserved.

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

8.18.2.6 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 hypervisor should fill the page with CPUID function structures as described in *Table 76 below*. These structures inform the guest of the machine configuration exposed to the guest by the hypervisor. However, a malicious hypervisor could provide a value that puts the guest in an insecure state. Therefore, the firmware checks each CPUID function structure to determine if the provided value is secure.

If firmware encounters a CPUID function that is not in the standard range (Fn0000_0000 through Fn0000_FFFF) or the extended range (Fn8000_0000 through Fn8000_FFFF), the firmware does not perform any checks on the function output.

If firmware encounters a CPUID function that is in the standard or extended ranges, then the firmware performs a check to ensure that the provided output would not lead to an insecure guest state. If insecure function output is identified, the firmware updates the field with an acceptable value.

Note: *Some functions have multiple acceptable values, and the firmware may choose any one of them. The firmware then returns INVALID_PARAM. In this failure case, the page is not encrypted with the VEK, the page measurement is not updated, and the page state remains unaltered.*

The policy used by the firmware to assess CPUID function output can be found in [PPR].

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 as described above.

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 76. CPUID Page Format

Byte Offset	Bits	Name	Description
00h	31:0	COUNT	Number of CPUID functions to validate. Must be less than or equal to 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 *Section 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.

8.18.3 Status Codes

Table 77. 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.
UPDATE_FAILED	Update of the firmware internal state or a guest context page has failed.

Status	Condition
UNSUPPORTED	A required feature is not supported.

8.19 SNP_LAUNCH_FINISH

This command completes the guest launch flow.

8.19.1 Parameters

Table 78. 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	63:3	–	–	Reserved. Must be zero.
	2	In	VCEK_DIS	Indicates that the VCEK is disabled for this guest.
	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 and the ID authentication information structure are present.
20h	255:0	In	HOST_DATA	Opaque host-supplied data to describe the guest. The firmware does not interpret this value.

Table 79. 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.

Byte Offset	Bits	Name	Description
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 80. Layout of the ID Authentication Information Structure

Byte Offset	Bits	Name	Description
0h	31:0	ID_KEY_ALGO	The algorithm of the ID Key. See *Appendix B* for details.
4h	31:0	AUTH_KEY_ALGO	The algorithm of the Author Key. See *Appendix B* for details.
8h–3Fh		–	Reserved. Should be zero.
40h–23Fh		ID_BLOCK_SIG	The signature of all bytes of the ID block. See *Appendix B* for the format of the signature.
240h–643h		ID_KEY	The public component of the ID key. See *Appendix B* for the format of the public key.
644h–67Fh		–	Reserved. Should be zero.
680h–87Fh		ID_KEY_SIG	The signature of the ID_KEY. See *Appendix B* for the format of the signature.
880h–C83h		AUTHOR_KEY	The public component of the Author key. See *Appendix B* for the format of the public key. Ignored if AUTHOR_KEY_EN is 0.
C84h–FFFh		–	Reserved. Should be zero.

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 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 81*.

Table 81. 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.
VcekDis	VCEK_DIS.

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

8.19.3 Status Codes

Table 82. 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.

Status	Condition
BAD_SIGNATURE	Incorrect signature provided.
UPDATE_FAILED	Update of the firmware internal state or a guest context page has failed

8.20 SNP_GUEST_STATUS

This command is used to retrieve information about an SNP guest. The returned GUEST_STATUS will be modified to include whether the chip secret VCEK has been selected for the target guest.

8.20.1 Parameters

Table 83. 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 84 below*.
	11:0	–	–	Reserved. Must be zero.

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 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 84. 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.

Byte Offset	Bits	Name	Description
0Dh	7:0	–	Reserved.
0Eh	15:0	–	Reserved.
10h	31:3	–	Reserved.
	2	CSVEK_EN	0: Legacy VCEK used by guest. 1: Chip secret VCEK used by guest.
	1	CSVCEK_PREF	0: Legacy VCEK preferred for guest report signing 1: Chip secret VCEK preferred for guest report signing
	0	VCEK_DIS	Value of VcekDis for this guest. Indicates that the guest cannot use the VCEK for attestation or key derivation.
14h	31:0	–	Reserved.
18h	63:0	–	Reserved.

8.20.3 Status Codes

Table 85. 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.
UPDATE_FAILED	Update of the firmware internal state or a guest context page has failed.

8.21 SNP_PAGE_MOVE

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

8.21.1 Parameters

Table 86. 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.

Byte Offset	Bits	In/Out	Name	Description
	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.

8.21.2 Actions

If SnpPageSwapDisablePolicy feature bit is set, the firmware checks the Guest policy bit PAGE_SWAP_DISABLE (bit 25). If this bit is set (enabled) then the firmware returns POLICY_FAILURE.

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 INVALID_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.

8.21.2.1 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 the 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.

8.21.2.2 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 the 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.

8.21.3 Status Codes

Table 87. Status Codes for SNP_PAGE_MOVE

Status	Condition
SUCCESS	Successful completion.

Status	Condition
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.
UPDATE_FAILED	Update of the firmware internal state or a guest context page has failed.
POLICY_FAILURE	The guest policy prevents this command from operation.

8.22 SNP_PAGE_MD_INIT

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

8.22.1 Parameters

Table 88. 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.

8.22.2 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_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 and then transitions it into a Metadata page, setting its RMP.GPA to GCTX_PADDR.

8.22.3 Status Codes

Table 89. 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.
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.
UPDATE_FAILED	Update of the firmware internal state or a guest context page has failed.

8.23 SNP_PAGE_SWAP_OUT

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

8.23.1 Parameters

Table 90. 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

Byte Offset	Bits	In/Out	Name	Description
				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 *Section 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	Deprecated page type of the source page. The FW will ignore this.
	0	In	PAGE_SIZE	Indicates page size. 0 indicates a 4 KB page. 1 indicates a 2 MB page.

8.23.2 Actions

If SnpPageSwapDisablePolicy feature bit is set, the firmware checks the Guest policy bit PAGE_SWAP_DISABLE (bit 25). If this bit is set (enabled) then the firmware returns POLICY_FAILURE.

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 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 any of these checks fails, the firmware returns INVALID_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.

8.23.2.1 Data Pages

The actions in this section are performed only when RMP.VMSA is 0 and it is not a Metadata page.

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 VEK to decrypt the contents of the source page, and it also uses the guest's OEK to wrap the contents with Aead_Wrap() (0) without AAD. The firmware checks that incrementing the OekIvCount would not cause an overflow. If overflow would occur, the firmware returns AEAD_OFLOW. Otherwise, the firmware increments the OekIvCount and uses that new value as the IV. The firmware then writes the produced ciphertext into the destination page.

The firmware then constructs a MDATA structure as described in *Table 91 below*. 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 91. Metadata Entry (MDATA) for Data Pages

MDATA Field	Value
SOFTWARE_DATA	SOFTWARE_DATA.
IV	Constructed from OekIvCount.
AUTH_TAG	Authentication tag generated by Aead_Wrap().
PAGE_SIZE	RMP.Page_Size of the source page.
VALID	1
METADATA	0

MDATA Field	Value
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.

8.23.2.2 Metadata Page

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 checks that incrementing the OekIvCount would not cause an overflow. If overflow would occur, the firmware returns AEAD_OFLOW. Otherwise, the firmware increments the OekIvCount and uses that new value as the IV. The firmware then writes the produced ciphertext into the destination page.

The firmware then constructs a MDATA structure as described in *Table 92 below*. 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 92. Metadata Entry (MDATA) for Metadata Pages

MDATA Field	Value
SOFTWARE_DATA	SOFTWARE_DATA.
IV	Constructed from OekIvCount.
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.

MDATA Field	Value
PAGE_VALIDATED	0
VMPL0	0h
VMPL1	0h
VMPL2	0h
VMPL3	0h

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

8.23.2.3 VMSA Pages

The actions in this section are performed only when RMP.VMSA is 1.

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 checks that incrementing the OekIvCount would not cause an overflow. If overflow occurs, the firmware returns AEAD_OFLOW. Otherwise, the firmware increments the OekIvCount and uses that new value as the IV. The firmware then writes the produced ciphertext into the destination page.

The firmware then constructs a MDATA structure as described in *Table 93 below*. 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 93. Metadata Entry (MDATA) for Data Pages

MDATA Field	Value
SOFTWARE_DATA	SOFTWARE_DATA.
IV	Constructed from OekIvCount.
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.

MDATA Field	Value
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.

8.23.3 Status Codes

Table 94. 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.
AEAD_OFLOW	An overflow in the IV counter was detected.
UPDATE_FAILED	Update of the firmware internal state or a guest context page has failed.
POLICY_FAILURE	The guest policy prevents this command from operation.

8.24 SNP_PAGE_SWAP_IN

This command swaps an SNP-protected page back in.

8.24.1 Parameters

Table 95. 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 *Section 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	Deprecated page type of the source page. The FW will ignore this.
	0	In	PAGE_SIZE	Indicates page size. 0 indicates a 4 KB page. 1 indicates a 2 MB page.

8.24.2 Actions

If SnpPageSwapDisablePolicy feature bit is set, the firmware checks the Guest policy bit PAGE_SWAP_DISABLE (bit 25). If this bit is set (enabled) then the firmware returns POLICY_FAILURE.

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 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 and destination pages. If any of these checks fails, the firmware returns INVALID_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 any of these checks fails, the firmware returns INVALID_PAGE_SIZE.

The metadata entry determines the source page type according to *Table 96 below*.

Table 96. 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 then checks that that the VALID bit in the metadata entry is set. If the 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 determined page type.

8.24.2.1 Data Pages

The actions in this section are performed only when the 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.

8.24.2.2 Metadata Pages

The actions in this section are performed only when the 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 plain text 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.

8.24.2.3 VMSA Pages

The actions in this section are performed only when the 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 bit 9 of SEV_FEATURES of the VMSA is 1, the firmware sets the GUEST_TSC_SCALE and GUEST_TSC_OFFSET fields of the VMSA as follows:

```
GUEST_TSC_SCALE := GCTX.DesiredTscFreq / (mean native frequency)
GUEST_TSC_OFFSET := GCTX.PspTscOffset
```

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

8.24.3 Status Codes

Table 97. Status Codes for SNP_PAGE_SWAP_IN

Status	Condition
SUCCESS	Successful completion.

Status	Condition
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.
UPDATE_FAILED	Update of the firmware internal state or a guest context page has failed.
POLICY_FAILURE	The guest policy prevents this command from operation.

8.25 SNP_PAGE_RECLAIM

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

8.25.1 Parameters

Table 98. Layout of the CMDBUF_SNP_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.

8.25.2 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. If the page size is 2 MB, the firmware then checks that PAGE_PADDR is 2 MB aligned. If not, the firmware returns INVALID_ADDRESS.

The firmware transitions the provided page according to *Table 99 below*.

Table 99. 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.

8.25.3 Status Codes

Table 100. Status Codes for SNP_PAGE_RECLAIM

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.

8.26 SNP_PAGE_UNSMASH

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

8.26.1 Parameters

Table 101. 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.

8.26.2 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 meets the following requirements.

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

If any of the above checks fails, 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.

8.26.3 Status Codes

Table 102. 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.

8.27 SNP_GUEST_REQUEST

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

8.27.1 Parameters

Table 103. 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 104. 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	127:64	–	Reserved. Must be zero.
	63:0	MSG_SEQNO	The sequence number for this message. Used to construct the IV.
30h	7:0	ALGO	The AEAD used to encrypt this message. See *Table 105 below*.
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 106 below*.
35h	7:0	MSG_VERSION	The version of the payload.
36h	15:0	MSG_SIZE	The size of the payload in bytes.
38h	31:0	–	Reserved. Must be zero.
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 105. AEAD Algorithm Encodings

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

Table 106. Message Type Encodings

Value	Message Type	Message Version
0	Invalid	–
1	SNP_MSG_CPUID_REQ	1
2	SNP_MSG_CPUID_RSP	1
3	SNP_MSG_KEY_REQ	3
4	SNP_MSG_KEY_RSP	3
5	SNP_MSG_REPORT_REQ	1
6	SNP_MSG_REPORT_RSP	1
7	SNP_MSG_EXPORT_REQ	1
8	SNP_MSG_EXPORT_RSP	1
9	Reserved (~~MSG_IMPORT_REQ~~)	–
10	Reserved (~~MSG_IMPORT_RSP~~)	–
11	Reserved (~~MSG_ABSORB_REQ~~)	–
12	Reserved (~~MSG_ABSORB_RSP~~)	–
13	SNP_MSG_VMRK_REQ	1
14	SNP_MSG_VMRK_RSP	1
15	SNP_MSG_ABSORB_NOMA_REQ	2
16	SNP_MSG_ABSORB_NOMA_RSP	2
17	SNP_MSG_TSC_INFO_REQ	1
18	SNP_MSG_TSC_INFO_RSP	1
27	SNP_MSG_CONFIG_VIOMMU_REQ	1
28	SNP_MSG_CONFIG_VIOMMU_RSP	1
31	SNP_MSG_GET_CSR_REQ	1
32	SNP_MSG_GET_CSR_RSP	1
37	SNP_MSG_GET_TPM_INFO_REQ	1
38	SNP_MSG_GET_TPM_INFO_RSP	1
All other encodings reserved.		–

8.27.2 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. The firmware checks that the response message will not cross a 4kB system physical page boundary when written. If either of these checks fails, the firmware returns INVALID_ADDRESS.

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

The firmware constructs the incoming 96-bit IV. The firmware sets bits IV[63:0] to the MSG_SEQNO and bits IV[95:64] to 0h.

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: Constructed 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.

Message types of SNP_MSG_IMPORT_REQ, SNP_MSG_IMPORT_RSP, SNP_MSG_ABSORB_REQ, and SNP_MSG_ABSORB_RSP are unsupported. If these types are used, 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 constructs 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 always encrypts the response with the same key that the guest used to encrypt the request even if a guest message triggers a key change.

The firmware constructs the IV by setting IV[63:0] to MSG_SEQNO and setting IV[95:64] to 0h.

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.

8.27.3 Status Codes

Table 107. 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 or a feature unsupported.
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.

Status	Condition
AEAD_OFLOW	The message sequence number was incorrect, or the guest's message count would overflow.
BAD_MEASUREMENT	The message failed to authenticate.
UPDATE_FAILED	Update of the firmware internal state or a guest context page has failed.

8.28 SNP_DBG_DECRYPT

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

8.28.1 Parameters

Table 108. 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.

8.28.2 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**: 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.

8.28.3 Status Codes

Table 109. 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.
UPDATE_FAILED	Update of the firmware internal state or a guest context page has failed.

8.29 SNP_DBG_ENCRYPT

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

8.29.1 Parameters

Table 110. 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.

8.29.2 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: *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.

8.29.3 Status Codes

Table 111. 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.
INACTIVE	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.
UPDATE_FAILED	Update of the firmware internal state or a guest context page has failed.

8.30 SNP_PAGE_SET_STATE

This command transitions Firmware pages to the HV-fixed page state. This operation cannot be undone until there is an RMP reinitialization or a system reset.

This command is available starting in version 1.52.

8.30.1 Parameters

Table 112. Layout of the CMDBUF_SNP_PAGE_SET_STATE Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length of this command buffer in bytes.
4h	31:0	–	–	Reserved.
8h	63:0	In	LIST_PADDR	sPA of the RANGE_LIST structure. See *Table 113 below*.

Table 113. Layout of the RANGE_LIST Structure

Byte Offset	Bits	Name	Description
0h	31:0	N	Number of elements in the RANGE_ARRAY.
4h	31:0	–	Reserved. Should be zero.
8h – (8h + N*16)		RANGES	Array of N elements of type RANGE. See *Table 114 below*.

Table 114. Layout of the RANGE Structure

Byte Offset	Bits	Name	Description
0h	63:12	BASE	Specifies the sPA of the first byte of the range. The address is formed by setting bits [63:12] to BASE and bits [11:0] to 0h.
	11:0	–	Reserved. Must be zero.
8h	31:0	PAGE_COUNT	Number of 4kB pages in this range.
Ch	31:0	–	Reserved. Must be zero.

8.30.2 Actions

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

If LIST_PADDR is an invalid sPA, the firmware returns INVALID_ADDRESS. If a range in RANGES contains an invalid address, the firmware returns INVALID_ADDRESS.

Each RANGE element must be 2MB aligned. If not, the firmware returns INVALID_PARAM.

The firmware checks that the pages containing the ranges enumerated in the RANGES structure are either in the Default or Firmware page state with RMP.Page_Size set to 0. If not, the firmware returns INVALID_PAGE_STATE. If the PAGE_COUNT field of a range is zero, the firmware ignores the range as if it were not provided.

The firmware ignores pages that are in the Default page state. For pages in the Firmware state, the firmware transitions the page to the HV-fixed page state.

If the firmware detects an error while processing the ranges, it will revert any changes made to the RMP before returning an error status code.

8.30.3 Status Codes

Table 115. Status Codes for SNP_PAGE_SET_STATE

Status	Condition
SUCCESS	Successful completion.

Status	Condition
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_ADDRESS	An address is invalid or misaligned.
INVALID_PAGE_STATE	A page is not in the correct state.

8.31 SNP_VLEK_LOAD

This command loads a VLEK to supplant the VCEK in scenarios where the hypervisor wishes to use a signing key that is not chip unique.

The hypervisor retrieves the wrapped VLEK hashsticks from the AMD KDS service by providing the CHIP_ID and TCB version similarly to the retrieval of the VCEK. The KDS will then provide a wrapped VLEK. The hypervisor then invokes this command to load the retrieved VLEK.

The version of the VLEK indicated in the TCB_VERSION of the WRAPPED_VLEK structure must be equal to the ReportedTcb at the time of load. If any firmware command causes the ReportedTcb to change, the previously loaded VLEK is deleted and a VLEK associated with the new ReportedTcb must be reloaded.

On SNP_SHUTDOWN, the VLEK is deleted.

This command is available in version 1.54. For firmware versions 1.55 and beyond, this command is present when the VLEK feature bit is set.

8.31.1 Parameters

Table 116. Layout of the CMDBUF_SNP_VLEK_LOAD Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length of this command buffer in bytes.
4h	7:0	In	VLEK_WRAPPED_ VERSION	Version of the wrapped VLEK hashstick structure. Must be 0h.
5h-7h		In	–	Reserved.
8h	63:0	In	VLEK_WRAPPED_ PADDR	sPA of a wrapped VLEK hashstick. The wrapped VLEK hashstick format is specified in *Table 117 below*.

Table 117. Layout of the WRAPPED_VLEK_HASHSTICK Structure (Version 0h)

Byte Offset	Bits	Name	Description
0h	95:0	IV	IV used to wrap chip-unique key.

Byte Offset	Bits	Name	Description
Ch-Fh		–	Reserved. Must be zero.
10h–18Fh		VLEK_WRAPPED	VLEK hashstick wrapped with a chip-unique key using AES-256-GCM.
190h	63:0	TCB_VERSION	The TCB version associated with this VLEK hashstick.
198h-19Fh		–	Reserved. Must be zero.
1A0h	127:0	VLEK_AUTH_TAG	AES-256-GCM authentication tag of the wrapped VLEK hashstick and TCB_VERSION.

8.31.2 Actions

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

The firmware reads the wrapped VLEK hashstick structure pointed at by VLEK_WRAPPED_PADDR. If TCB_VERSION of the VLEK structure is not equal to the ReportedTcb, the command fails with BAD_SVN.

The firmware uses AES-256-GCM to unwrap the VLEK and to authenticate the provided VLEK hashstick structure. The VLEK_WRAPPED field is encrypted, and bytes 190h–19Fh are additionally authenticated data. If the authentication fails, the firmware returns BAD_MEASUREMENT.

The firmware saves the unwrapped VLEK hashstick to internal memory.

On failure, this command does not delete the existing VLEK, if present.

8.31.3 Status Codes

Table 118. Status Codes for SNP_VLEK_LOAD

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
BAD_SVN	ReportedTcb is not equal to TCB_VERSION.
BAD_MEASUREMENT	Failed to authenticate wrapped VLEK hashstick.
INVALID_PARAM	Unexpected length of wrapped VLEK hashstick.

8.32 SNP_TSC_INFO

The SNP_TSC_INFO command outputs GUEST_TSC_SCALE, GUEST_TSC_OFFSET, and TSC_FACTOR for a given guest.

8.32.1 Parameters

Table 119: CMDBUF_SNP_TSC_INFO Command Structure

Byte Offset	Bits	Name	Description
0h	31:0	LENGTH	Length of this command buffer in bytes.
4h	31:0	–	Reserved.
8h	63:12	GCTX_PADDR	Bits 63:12 of the system physical address of a page donated to the firmware by the hypervisor to contain the guest context.
	11:0	–	Reserved. Must be zero.
10h	63:0	TSC_INFO_PADDR	System physical address of a location the firmware will write the TSC_INFO structures. See *Table 38*.

8.32.2 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 TSC_INFO_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS.

The firmware checks that TSC_INFO_PADDR page is a Firmware or Default page. If not, the firmware returns INVALID_PAGE_STATE.

8.32.3 Status Codes

Table 120. Status Codes for SNP_TSC_INFO

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.

8.33 SNP_HV_REPORT_REQ

The Host OS can directly request that the firmware construct a guest attestation report.

A Host OS requests an attestation report by constructing an SNP_HV_REPORT_REQ as specified in *Table 121 below*.

8.33.1 Parameters

Table 121. CMDBUF_SNP_HV_REPORT_REQ Message Structure

Byte Offset	Bits	Name	Description
00h	31:0	LENGTH	Length of this command buffer in bytes.
4h	31:2	–	Reserved.
	1:0	KEY_SEL	Selects which key to use for generating the signature. 0: Sign with VLEK if VLEK is installed. Otherwise, sign with the key indicated by CsVcekPref. 1: Sign with legacy VCEK. 2: Sign with VLEK. 3: Sign with chip secret VCEK.
08h	63:12	GCTX_PADDR	Bits 63:12 of the sPA of the guest context page.
	11:0	–	Reserved. Must be zero.
10h	63:0	HV_REPORT_PADDR	sPA of a location the firmware will write the SNP_MSG_REPORT_RSP Message Structure.

8.33.2 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_RUNNING states. If not, the firmware returns INVALID_GUEST_STATE.

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

The firmware checks that HV_REPORT_PADDR page is a Firmware or Default page. If not, the firmware returns INVALID_PAGE_STATE.

Upon receiving a request for an attestation report, the firmware constructs the report according to *Table 27*.

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 an associated migration agent, the REPORT_ID_MA is filled in with the report ID of the migration agent.

If MaskChipKey is 0, the firmware signs the attestation report with either the VCEK or VLEK based on the key selection made in KEY_SEL. The firmware will return INVALID_KEY if any of the following conditions occur:

- KEY_SEL is 0, the VLEK is not loaded, and VcekDis is 1.
- KEY_SEL is 1 and VcekDis is 1.
- KEY_SEL is 2 and the VLEK is not loaded.
- KEY_SEL is 3, and VcekDis is 1 or CsVcekEn is 0.

The KEY_SEL field will be processed as follows based on the CsVcekEn and CsVcekPref settings in the target guest context.

- ‘00’: If VLEK is installed, sign with VLEK. If no VLEK is installed, and CsVcekPref is 1, use chip secret VCEK. Otherwise, use legacy VCEK.
- ‘01’: Legacy VCEK used for signing.
- ‘10’: VLEK used for signing.
- ’11’: If CsVcekEn is 1 then chip secret VCEK used for signing.

The firmware uses the systemwide ReportedTcb value as the TCB version to derive the VCEK, VLEK, or CSVCEK. This value is set by the hypervisor. The firmware guarantees that the ReportedTcb value is never greater than the installed TCB version.

If MaskChipKey is 1, the firmware writes zeroes into the SIGNATURE field instead of signing the report.

8.33.3 Status Codes

Table 122. Status Codes for CMDBUF_SNP_HV_REPORT_REQ

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_GUEST	The guest is invalid.
INVALID_GUEST_STATE	The guest is not in the correct state.
INVALID_ADDRESS	An address is invalid for use by the firmware or is misaligned.
INVALID_PAGE_STATE	A page was in the incorrect state.
INVALID_PARAM	Incorrect parameter or MBZ fields are not zero.
INVALID_KEY	The selected key is invalid.

8.33.4 Return Information

See *Table 27* for the returned data.

8.34 SNP_INIT_START

Initializes the RMP while SYS_CFG[SNPE] is not set. The detailed description of the actions performed by the firmware upon invocation of this command is described in *Section 5.2*. This command takes no arguments.

Software does not need to suspend executions of legacy (non-SEV) guests or set VM_HSAVE_PA to 0h during this command. However, software and devices must not write to the RMP starting at invocation of this command until SNP_INIT_FINISH command completes successfully and the SNP_PREINIT_STATE is returned to UNINIT. This exclusion also applies to the SNP x86 instruction set.

***Note**: Execution of the SNP x86 instruction set will fault if SYS_CFG[SNPEn] is not set.*

8.34.1 Parameters

There are no parameters for this command.

8.34.2 Actions

The firmware checks that SNP is in the UNINIT state. If not, the firmware returns INVALID_PLATFORM_STATE. FW checks that the RMP is initialized. If not, the firmware

returns INVALID_PLATFORM_STATE. Firmware checks that the SNP_PREINIT_STATE is UNINIT. If not, the firmware returns INVALID_PLATFORM_STATE.

The firmware ensures the following system requirements are met:

- SYS_CFG[SNPEn] is set across all cores.
- SYS_CFG[MFDM] must be set across all cores.
- HWCR[SmmLock] (MSR C001_0015) must be set identically across all cores.

The following MSRs must be set identically across all cores:

- All MTRRs
- IORR_BASE
- IORR_MASK
- TOM
- TOM2

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

The firmware 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.

Except for SYS_CFG[SNPEn], the firmware saves each of the checked MSR values and the locations of the event log, PPR log, and completion wait buffers of the IOMMU to its internal memory.

The firmware makes the RMP read-only. Software must suspend guests when this command is invoked. The sequence of actions taken by FW are described in *Section 5.2.1* for non-segmented RMP and *Section 5.2.2* for segmented RMP (RST). The firmware marks all pages as hypervisor except:

- The pages containing the RMP are set to the Firmware state.
- The pages containing the IOMMU event log, PPR log, and completion wait buffers are read-only and cannot be assigned to a guest.
- All MMIO ranges are set to the HV-fixed state.

The firmware also initializes any microarchitectural data structures within the RMP.

The firmware sets the RmpInstallPending internal flag and sets the SNP_PREINIT_STATE to SNP_PREINIT_CONTINUE_PENDING.

8.34.3 Status Codes

Table 123. Status Codes for SNP_INIT_START

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform and/or the RMP table is not in the UNINIT state.
INVALID_ADDRESS	An invalid address was provided.
INVALID_CONFIG	An invalid configuration exists.

8.35 SNP_INIT_CONTINUE

Continues the initialization of the RMP. The detailed description of the actions performed by FW upon invocation of this command is described in *Section 5.2*. Before invoking this command, software must set ensure that the SNP_PREINIT_STATE is SNP_PREINIT_CONTINUE_PENDING. Guests cannot run while this command is executing.

8.35.1 Parameters

Table 124. Layout of the CMDBUF_SNP_INIT_CONTINUE Message Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length of this command buffer.
4h	31:3	–	–	Reserved. Must be zero.
	2	In	TIO_EN	0: Trusted I/O is not enabled 1: Trusted I/O is enabled
	1	In	LIST_PADDR_EN	0: LIST_PADDR is not valid 1: LIST_PADDR is valid
	0	In	CANCEL	Results in RMP being returned to the uninitialized state.
8h	63:0	In	LIST_PADDR	SPA of the RANGE_LIST structure. See *Table 113*.
10h-39h	–	–	–	Reserved. Must be zero.

8.35.2 Actions

The firmware checks that SNP is in the UNINIT state. If not, the firmware returns INVALID_PLATFORM_STATE. The firmware checks that the RmpInstallPending and RstInstallPending are cleared. If not, the firmware returns INVALID_PLATFORM_STATE.

Firmware checks that the SNP_PREINIT_STATE is SNP_PREINIT_CONTINUE_PENDING. If not, the firmware returns INVALID_PLATFORM_STATE.

If LIST_PADDR_EN is 1, then the firmware checks that LIST_PADDR is a valid SPA. If not, the firmware returns INVALID_ADDRESS. If a range in RANGES contains an invalid address, the firmware returns INVALID_ADDRESS. For this check, a range overlapping the RMP is not considered an invalid address. Depending on the setting of TIO_EN, the firmware will initialize regions in the IOMMU.

If CANCEL is set, the firmware performs the following actions and then returns SUCCESS.

- Removes the write protection on the RMP.
- Marks that the RMP must be re-initialized before SNP_INIT_EX can successfully complete.
- SNP_PREINIT_STATE is set to UNINIT
- Clears SYS_CFG[SNPEn] and SYS_CFG[VMPLEn] if they were set.
- SNP enforcement of IOMMU protection is disabled

The firmware performs the following checks:

- SYS_CFG[SNPEn] is set on all cores.
- SYS_CFG[VMPLEn] is set on all cores.

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

The firmware initializes the IOMMU to perform RMP enforcement. The firmware then forces a TLB invalidation across all cores and IOMMUs.

Finally, the FW enables VM_HSAVE_PA so that guests can resume and sets the SNP_PREINIT_STATE to SNP_PREINIT_FINISH_PENDING.

8.35.3 Status Codes

Table 125. Status Codes for SNP_INIT_CONTINUE

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the UNINIT state.
INVALID_ADDRESS	An invalid address was provided.
INVALID_CONFIG	An invalid configuration exists.

8.36 SNP_INIT_FINISH

SNP_INIT_FINISH is the last command invoked in creation of an RMP. The detailed description of the actions performed by the firmware upon invocation of this command is described in *Section*

5.2. Before invoking this command, software must ensure that the SNP_PREINIT_STATE is SNP_PREINIT_FINISH_PENDING. Guests cannot run while this command is executing.

8.36.1 Parameters

Table 126. Layout of the CMDBUF_SNP_INIT_FINISH Message Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length of this command buffer in bytes.
4h	31:1	–	–	Reserved. Must be zero.
	0	In	CANCEL	Results in RMP being returned to the uninitialized state.
8h-1fh	–	–	–	Reserved. Must be zero.

8.36.2 Actions

The firmware checks that SNP is in the UNINIT state. If not, the firmware returns INVALID_PLATFORM_STATE. FW checks that the RmpInstallPending and RstInstallPending are cleared. If not, the firmware returns INVALID_PLATFORM_STATE. The firmware checks that the SNP_PREINIT_STATE is SNP_PREINIT_FINISH_PENDING. If not, the firmware returns INVALID_PLATFORM_STATE.

If CANCEL is set, the firmware performs the following actions and then returns SUCCESS.

- Removes the write protection on the RMP.
- Marks that the RMP must be re-initialized before SNP_INIT_EX can successfully complete.
- SNP_PREINIT_STATE is set to UNINIT.
- Clears SYS_CFG[SNPEn] and SYS_CFG[VMPLEn] if they were set.
- SNP enforcement of IOMMU protection is disabled.

The firmware performs the following checks:

- SYS_CFG[SNPEn] is set on all cores.
- SYS_CFG[VMPLEn] is set on all cores.

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

The firmware initializes the IOMMU to perform RMP enforcement. The firmware then forces a TLB invalidation across all cores and IOMMUs.

Finally, the FW enables VM_HSAVE_PA so that guests can resume and sets the SNP_PREINIT_STATE to SNP_PREINIT_STATE_UNINIT.

8.36.3 Status Codes

Table 127. Status Codes for SNP_INIT_FINISH

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the UNINIT state.
INVALID_ADDRESS	An invalid address was provided.
INVALID_CONFIG	An invalid configuration exists.

8.37 SNP_VERIFY_MITIGATION

The SNP_VERIFY_MITIGATION command allows the Host OS/HV (Host) to initiate commands to the SEV firmware to perform vulnerability mitigation operation initiation, and to request vulnerability mitigation supported and vulnerability mitigation verification status. The command action executions are specified and then applied to specific designated vulnerability mitigations.

8.37.1 Parameters

Table 128. Layout of the CMDBUF_SNP_VERIFY_MITIGATION Structure

Byte Offset	Bits	In/Out	Name	Description
00h	31:0	In	LENGTH	Length of this structure in bytes.
04h	15:0	In	SUBCOMMAND	Mitigation sub-command for the firmware to execute.
06h-07h	–	–	–	Reserved.
08h	63:0	In	VECTOR	VECTOR is only valid for MIT_REQ_VERIFY sub-command. The VECTOR Bit field contains a single bit specifying the vulnerability mitigation to process and/or validate. Must be a single bit within the vector field value.
10h	31:2	–	–	Reserved. MBZ.
	1	In	SRC_PADDR_EN	0: SRC_PADDR is not valid 1: SRC_PADDR is valid If SRC_PADDR_EN is set (valid) then SRC_PADDR must be set. If SRC_PADDR is not set, then SRC_PADDR must be zero.

Byte Offset	Bits	In/Out	Name	Description
	0	In	DST_PADDR_EN	0: DST_PADDR is not valid 1: DST_PADDR is valid If DST_PADDR_EN is set (valid) then DST_PADDR must be set. If DST_PADDR is not set, then DST_PADDR must be zero.
14h-17h	–	–	–	Reserved. MBZ.
18h	63:12	In	SRC_PADDR	Bits 63:12 of the sPA of the 4 KB region to contain optional input data. ***Note**: SRC_PADDR is intended for future use.*
	11:0	–	–	Reserved. Must be zero.
20h	63:12	In	DST_PADDR	Bits 63:12 of the sPA of the 4 KB region page to contain optional output data.
	11:0	–	–	Reserved. Must be zero.
28h-3Fh	–	–	–	Reserved.

Table 129. Command Descriptions

SNP_VERIFY_MITIGATION Sub-Command	Sub-Command Number	VECTOR	Description
MIT_REQ_STATUS	0h	MBZ	Command to request the firmware to return information regarding the currently supported (available) mitigations, and then the verified (processed and completed) mitigations. If DST_PADDR_EN is set, DST_PADDR will be populated with the SNP_VERIFY_MITIGATION_DST_PADDR structure.
MIT_REQ_VERIFY	1h	64-bit field with single bit set	Command to request the firmware to initiate the mitigation verification operations for a specific mitigation. If DST_PADDR_EN is set, then the command additionally populates DST_PADDR with the SNP_VERIFY_MITIGATION_DST_PADDR structure.

Table 130. Layout of the SNP_VERIFY_MITIGATION_DST_PADDR Structure

Byte Offset	Bits	Name	Description
0h	63:0	MIT_VERIFIED_VECTOR	Bit vector of vulnerability mitigations verified.

Byte Offset	Bits	Name	Description
8h	63:0	MIT_SUPPORTED_VECTOR	Bit vector of vulnerability mitigations supported.
10h	31:0	MIT_FAILURE_STATUS	The firmware provides status of the verification operation: 0h: No error condition 1h: Invalid SoC version or invalid platform 2h: Microcode version check failure 3h: Verification failure Possible verification failures include: • WBINVD not executed during the mitigation process. • Incorrect ucode loaded. • Ucode not loaded on all CPU cores. 4h: Other or hardware internal failure 5h: Shutdown required 6h: RMP re-initialization required 7h: Committed TCB is too low 8h: Hardware registers are in use 9h-FFFFh: Reserved.
14h-FFFh	–	–	Reserved.

Table 131. Layout of the SNP_VERIFY_MITIGATION SRC_PADDR Structure

Byte Offset	Bits	Name	Description
0h-FFFh	–	–	Reserved.

8.37.2 Actions

The firmware checks that the COMMAND value is equal to a valid command range. If not, the firmware returns INVALID_PARAM.

If DST_PADDR_EN is set to 1, then the firmware checks that the DST_PADDR should be set for this command. If DST_PADDR should not be set, then the firmware returns INVALID_PARAM. If DST_PADDR is a valid parameter, then the firmware checks that the DST_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS. The same checks also apply to SRC_PADDR_EN and SRC_PADDR.

If DST_PADDR is valid and RMP tables are enabled, then the firmware checks that the page is either a Firmware or Default page. If not, then the firmware returns INVALID_PAGE_STATE. If DST_PADDR is valid and RMP tables are not enabled, then the DST_PADDR page can be in any page state and is not checked by the firmware.

If the command is MIT_REQ_VERIFY, then the firmware executes the mitigation verification process for the requested vector bit. If the verification process is successful, the firmware sets the corresponding vector bit to 1 in the internal completed vector value. If the verification process fails, the firmware sets the corresponding vector bit to 0 in the internal completed vector value.

The firmware will check the SNP firmware state based on the mitigation requirements, and if the SNP firmware state is incorrect, then the firmware will return INVALID_PLATFORM_STATE.

If DST_PADDR_EN is set, then the firmware sets the internal completed vector value (MIT_VERIFIED_VECTOR), the internal mitigation available value (MIT_SUPPORTED_VECTOR), and the failure status (MIT_FAILURE_STATUS) into the DST_PADDR locations according to the SNP_VERIFY_MITIGATION_DST_PADDR structure.

8.37.3 Status Codes

Table 132. Status Codes for SNP_VERIFY_MITIGATION

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	An address is invalid or misaligned.
INVALID_PARAM	MBZ fields are not zero.
INVALID_PLATFORM_STATE	SEV-SNP firmware state is invalid.
INVALID_PAGE_STATE	A page is not in the correct state.

8.38 FEATURE_INFO

This command returns the contents of CPUID Fn8000_0024 to provide information regarding the features present in the SEV firmware currently loaded. The caller provides the value of the ECX input index and returns the CPUID subfunction of that index. The CPUID instruction treats all subfunctions of Fn8000_0024 as reserved.

The output of this command may be provided to the guest via the CPUID page in SNP_LAUNCH_UPDATE as well as through the CPUID reporting guest message SNP_MSG_CPUID_REQ.

This command is available if bit 3 of offset 8h of the output buffer of SNP_PLATFORM_STATUS is 1.

8.38.1 Parameters

Table 133. Layout of the CMDBUF_SNP_FEATURE_INFO Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length of this command buffer in bytes.
4h	31:0	In	ECX_IN	Subfunction index of CPUID Fn8000_0024.
8h	63:0	In	FEATURE_INFO_PADDR	System physical address of the FEATURE_INFO structure to write the output to.

Table 134. Layout of the FEATURE_INFO Structure (Version 1h)

Byte Offset	Bits	Name	Description
0h	31:0	EAX	See *Section 4.13*
4h	31:0	EBX	
8h	31:0	ECX	
Ch	31:0	EDX	

8.38.2 Actions

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

The firmware checks the FEATURE_INFO_PADDR is a valid address, is 8-byte aligned, and does not cross a page boundary. If any of these checks fail, the firmware returns INVALID_PARAM.

If the SNP firmware state is INIT, the FEATURE_INFO_PADDR page must be either a Firmware page 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 firmware writes feature information into the FEATURE_INFO structure at FEATURE_INFO_PADDR. The information written into FEATURE_INFO is determined by ECX_IN. *Section 4.13* specifies the returned information for each ECX_IN.

Fn8000_0024_x00 is always valid when this command is present. To determine if other indices are valid, the MaxIndex (Fn8000_0024_EAX_x00[31:0]) indicates the maximum valid sub-function index. If ECX_IN is greater than MaxIndex, then this command will return INVALID_PARAM and will not write to the FEATURE_INFO structure.

8.38.3 Status Codes

Table 135. Status Codes for SNP_FEATURE_INFO

Status	Condition
SUCCESS	Successful completion.
INVALID_PARAM	Incorrect parameter or out of range ECX_IN value.
INVALID_PAGE_STATE	Incorrect page state.

8.39 SNP_CSVCEK_CERT

This command returns the certificate chain associated with the chip secret VCEK.

8.39.1 Parameters

Table 136. Layout of the CMDBUF_SNP_CSVCEK Structure

Byte Offset	Bits	In/Out	Name	Description
00h	31:0	IN	LENGTH	Length in bytes of this command buffer.
04h	31:0	–	Reserved	MBZ
08h	63:0	In	CERT_SLA	A scatter list address pointing to a buffer to be used for returning the CSVCEK certificate chain. See Chapter 4 of [SEV-TiO].

8.39.2 Actions

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

The firmware checks that CERT_SLA is a valid scatter list address. If not, the firmware returns INVALID_ADDRESS.

The firmware checks that the data pages of the buffer pointed at by CERT_SLA are all in the Firmware or Default page states. If not, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the capacity of the buffer provided is large enough to hold a the CSVCEK certificate chain. If not, the firmware returns INVALID_LENGTH.

The certificate chain is returned with an object header, as defined in Chapter 4 of [SEV-TIO]. The header is given in *Table 137*.

Table 137. DATA_OBJECT_HEADER for CSVCEK Certificate Object

Byte Offset	Bits	Name	Description
0h	31:0	DOBJ_ID	10h.
4h	31:0	DOBJ_LENGTH	Length of the data object in bytes, including this header.
8h	15:0	DOBJ_VERSION	Version of the data object structure. Bit[7:0]: Minor version. Bit[15:8]: Major version.
Ah	47:0	–	Reserved.
10h	–	DOBJ_PAYLOAD	

8.39.3 Status Codes

Table 138. Status Codes for CMDBUF_SNP_SNP_CSVCEK

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
INVALID_LENGTH	A provided buffer was too small.

8.40 SNP_GET_CORIM

This command will provide a list of CoRIM IDs as a sequence of byte strings, each of size 16 as per *https://datatracker.ietf.org/doc/html/draft-ietf-rats-corim#name-uuid*.

***Note**: A single CoRIM can encompass multiple measurements.*

8.40.1 Parameters

Table 139. CMDBUF_SNP_GET_CORIM Message Structure

Byte Offset	Bits	In/Out	Name	Description
00h	31:0	IN	LENGTH	Length of this command buffer in bytes.
04h	31:0	–	Reserved	Mbz.
08h	63:0	IN	CORIM_ID_SLA	A scatter list address pointing to a buffer to be used for returning the CoRIM ID list. See Chapter 4 of [SEV-TIO].

8.40.2 Actions

The sequence of actions taken by firmware is as follows:

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

The firmware checks that CORIM_ID_SLA is a valid scatter list address. If not, the firmware returns INVALID_ADDRESS.

The firmware checks that the data pages of the buffer pointed at by CORIM_ID_SLA are all in the Firmware or Default page states. If not, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the capacity of the buffer provided is large enough to hold the CoRIM ID list. If not, the firmware returns INVALID_LENGTH.

The CoRIM ID list is returned with an object header, as defined in Chapter 4 of [SEV-TIO]. The header is given in *Table 140 below.* The VCEK-generated signature (P384) is appended to the end of the list and is included in the data object payload. The VCEK selected for signing (chip secret or legacy) will be based on what is supported by the platform (as indicated by the CsVcek bit in FEATURE_INFO). If the platform supports chip secret VCEK derivation, then that will be the keypair leveraged for signing.

Table 140. DATA_OBJECT_HEADER for CoRIM ID List Object

Byte Offset	Bits	Name	Description
0h	31:0	DOBJ_ID	11h.
4h	31:0	DOBJ_LENGTH	Length of the data object in bytes, including this header.
8h	15:0	DOBJ_VERSION	Version of the data object structure. Bit[7:0]: Minor version. Bit[15:8]: Major version.
Ah	47:0	–	Reserved.
10h	15:0	NUM_CORIM_ID	Number of CoRIM IDs in returned list
12h – 1Fh		–	Reserved.
20h	–	DOBJ_PAYLOAD	

8.40.3 Status Codes

Table 141. Status Codes for CMDBUF_SNP_GET_CORIM

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform state was invalid.
INVALID_ADDRESS	The address provided was invalid.
INVALID_PAGE_STATE	The provided page was not in the expected page state.
INVALID_LENGTH	The buffer provided was too small.

8.41 SNP_GET_CSR

The SNP_GET_CSR command allows the hypervisor to retrieve a null-signed CSR for the LDevID certificate. The hypervisor takes on responsibility for managing the certificate chain that results from this CSR.

8.41.1 Parameters

Table 142. CMDBUF_SNP_GET_CSR Message Structure

Byte Offset	Bits	Name	Description
00h	31:0	LENGTH	Length of this command buffer in bytes.
4h	31:1	–	Reserved.
	0	CSR_SELECT	1: Return LDevID CSR 0: Return FMC_Alias CSR
08h-10h	63:0	HV_CSR_PADDR	sPA of a location that the firmware will write the CSR.

8.41.2 Actions

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

The firmware checks that HV_CSR_PADDR is a valid sPA. If not, the firmware returns INVALID_ADDRESS. The firmware checks that HV_CSR_PADDR page is a Firmware or Default page. If not, the firmware returns INVALID_PAGE_STATE.

The CSR (as indicated by CSR_SELECT) is returned to the address specified by HV_CSR_PADDR. This address must be 4 kB page aligned. The firmware writes the CSR to HV_CSR_PADDR. The format of the returned CSR is given in *Table 143 below*.

Table 143. Layout of the CSR

Byte Offset	Bits	Name	Description
0h	15:0	NUM_BYTES	Number of bytes in CSR
2h	47:0	–	Reserved.
08h	–	CSR	Certificate signing request.

8.41.3 Status Codes

Table 144. Status Codes for CMDBUF_SNP_GET_CSR

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_ADDRESS	The address provided was invalid.
INVALID_PAGE_STATE	The provided page was not in the expected page state.

8.42 SNP_GET_TPM_INFO

The SNP_GET_TPM_INFO host command returns the TPM_INFO structure for a specific guest.

8.42.1 Parameters

Table 145. Layout of the CMDBUF_GET_TPM_INFO structure.

Byte Offset	Bits	Name	Description
0h	31:0	VERSION	Version of this structure.
08h	63:12	GCTX_PADDR	Bits 63:12 of the SPA of the guest context page.
	11:0	–	Reserved. Must be zero.
10h	127:0	NONCE	The nonce to include in the TPM_INFO structure.
20h	63:0	TPM_INFO_PADDR	The SPA of destination buffer to write the TPM_INFO structure to.

8.42.2 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 TPM_INFO_PADDR are valid sPAs. If either check fails, the firmware returns INVALID_ADDRESS. The firmware checks that the TPM_INFO_PADDR is aligned to a 4 KB boundary. If not, 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 page containing TPM_INFO_PADDR is a Firmware or Default page. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the guest is in the GSTATE_LAUNCH or GSTATE_RUNNING states. If not, the firmware returns INVALID_GUEST_STATE.

If the platform does not have a TPM to report, the command returns a TPM_INFO structure with the PRESENT bit cleared and the following fields zeroed.

If the platform does have a TPM, the firmware constructs a TPM_INFO object with the following

- The NONCE field is set to the NONCE value here.
- The GUEST field is cleared.
- The PRESENT bit is set.
- The REPORT_ID set to the REPORT_ID of the targeted guest.
- The EK_DIGEST is set to the collected EK digest.
- The NV_INDEX is set to the NV index used to retrieve the EK.
- The SIGNATURE is the signature over the TPM_INFO object.
- The SIGNING_KEY is set to indicate the key that signed this object.

The firmware writes the TPM_INFO structure to TPM_INFO_PADDR.

8.42.3 Status Codes

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_ADDRESS	The address provided by the firmware was invalid or misaligned.
INVALID_GUEST	The guest is invalid.
INVALID_PAGE_STATE	The provided page was not in the expected page state.

Appendix A Common Algorithms

A.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)

A.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

Appendix B Digital Signatures

The SNP firmware uses digital signatures to sign objects such as the attestation report and to validate signatures such as the ID block. The supported algorithms and their encodings are described in *Table 146*, *Table 147*, *Table 148*, and *Table 149*.

Table 146: Encoding for Signing Algorithms

Signing Algorithm	Encoding
ECDSA P-384 with SHA-384	1h
All other encodings are reserved.	

Elliptic curves are defined in *Table 147 below*.

Table 147. ECC Curve Identifier Encodings

ECC Curve	Encoding
P-384	2h
All other encodings are reserved.	

The ECDSA P-384 with SHA-384 signature format is defined in *Table 148 below*.

Table 148. Format for an ECDSA P-384 with SHA-384 Signature

Byte Offset	Bits	Name	Description
000h	575:0	R	R component of this signature. Value is zero-extended little-endian encoded.
048h	575:0	S	S component of this signature. Value is zero-extended little-endian encoded.
090h–1FFh		–	Reserved.

The ECDSA P-384 public key format is defined in *Table 149 below*.

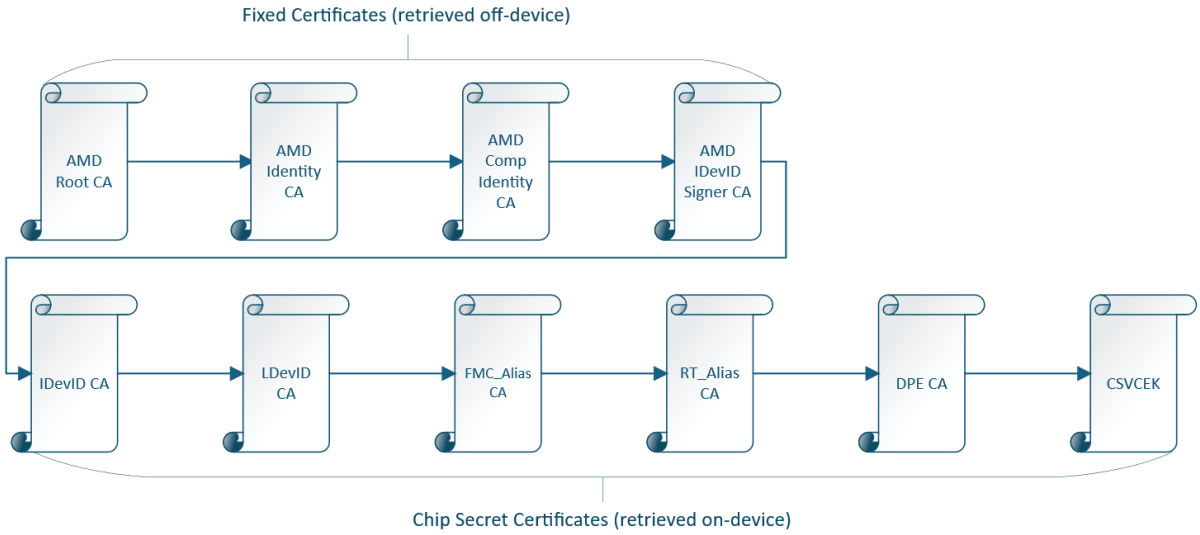
Table 149. Format for an ECDSA P-384 Public Key

Byte Offset	Bits	Name	Description
000h	31:0	CURVE	Curve ID. 2h indicates P-384. All other encodings are reserved.
004h	575:0	QX	X component of this signature. Value is zero-extended little-endian encoded.
04Ch	575:0	QY	Y component of this signature. Value is zero-extended little-endian encoded.
094h–403h		–	Reserved. Must be zero.

Appendix C CSVCEK Certificate Chain

The chip-secret VCEK (CSVCEK) certificate chain can be retrieved on device. This section provides the certificate chain that endorses the CSVCEK and is intended to be used by a verifier when validating the SNP guest report. The certificate chain is shown in *Figure 3*.

Figure 3. CSVCEK Certificate Chain



The AMD Root CA through the AMD IDevID Signer CA certificates are fixed per family and are distributed through *https://kds-dev.amd.com/*. The IdevID through DPE certificates follow the formats described in *Caliptra 1.x*.

The CSVCEK certificate format is provided below. The IDevID through the CSVCEK certificates are retrieved on-device though corresponding SNP ABI calls.

Table 150. CSVCEK Certificate Fields

Certificate Field	Entry
Version	V3
Serial Number	20-byte value, where first byte is h04 and all remaining bytes are randomly-generated
Issuer	CN = DPE Leaf, serialNumber =
Signature Algorithm	secp384r1
Signature Hash Algorithm	sha384
Validity	Not before: January 1, 2023

Certificate Field	Entry
	Not after: December 31, 9999
Subject	CN = SEV-VCEK O = Advanced Micro Devices L = Santa Clara ST = California C = US
Subject Public Key Info	ECDSA on curve P-384
Extensions	See Table 11 of Versioned Chip Endorsement Key (VCEK) Certificate and KDS Interface Specification for prior family extensions. Will re-use the following FW OIDs from Turin: • teeSPL • fmcSPL • snpSPL In addition, hwID, structVersion and productName (“Venice”) will be included.

Appendix D Certificate Signing Requests

There are two certificate signing requests (CSRs) that are returned in SNP – the LDevID CSR (returned to hypervisor) and the FMC_Alias CSR (returned to guest). The hypervisor and guest respectively can use these CSRs to replace the CSVCEK certificate chain with their own. The CSR formats are provided below.

Table 151. LDevID CSR Format

CSR Field	Entry
Version	V1
Signature Algorithm	secp384r1
Signature Hash Algorithm	sha384
Subject	CN = LDEVID_
Signature	Null

Table 152. FMC_Alias CSR Format

CSR Field	Entry
Version	V1
Signature Algorithm	secp384r1
Signature Hash Algorithm	sha384
Subject	CN = Caliptra 1.0 FMC Alias