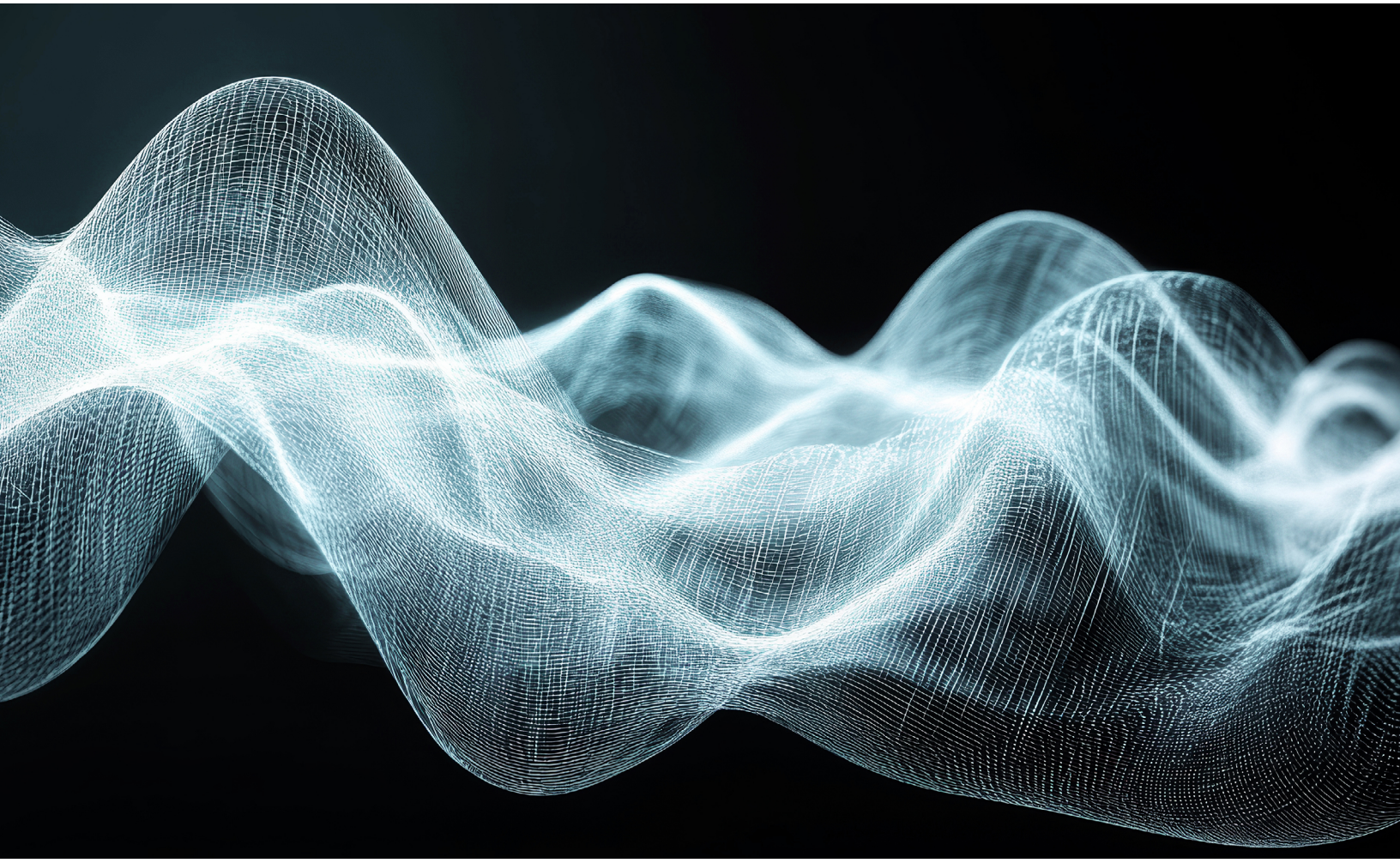

USER GUIDE: AMD EPYC 9005, 9004, 7003, 7002

USING SEV WITH AMD EPYC™ PROCESSORS

TABLE OF CONTENTS

AUDIENCE

This tuning guide is intended for a technical audience such as production deployment, virtualization developers, firmware engineers, and performance engineering teams with:

- A background in configuring servers.
- Access to the system BIOS.

AUTHOR

Diego Gonzalez Villalobos

DATE	VERSION	CHANGES
MARCH 2023	1.0	Initial release
OCTOBER 2023	1.1	
NOVEMBER 2025	2.0	• Added Turin information and SNP upstream support. • Added details about physical address bits when configuring CPU models in QEMU for SNP VM launches. • Added information regarding the SEV rebranding • Added a section on TIO

CHAPTER 1

SECURITY FEATURES

1.1 SEV NAMING

AMD has simplified Secure Encrypted Virtualization (SEV) naming to better communicate its features across the different AMD EPYC™ CPU generations. The new name corresponds to both the hardware technology and the software and firmware enablement for each SEV generation. Every AMD EPYC™ processor family introduced SEV improvements, sometimes with a milestone feature, such as Secure Nested Paging, or with several other enhancements like in the case of AMD EPYC™ 9004.

The chart below illustrates how each SEV generation aligns with its corresponding major feature:

SEV GENERATION	EPYC GENERATION	MILESTONE HARDWARE FEATURE	FEATURE ACRONYM
SEV 1.0	AMD EPYC™ 7001	Legacy Secure Encrypted Virtualization	**LEGACY SEV**
SEV 2.0	AMD EPYC™ 7002	Encrypted State	**ES**
SEV 3.0	AMD EPYC™ 7003	Secure Nested Paging	**SNP**
SEV 3.1	AMD EPYC™ 9004, AMD EPYC™ 8004		
SEV 4.1	AMD EPYC™ 9005	Trusted I/O	**TIO**

Table 1-1: SEV Feature branding

Note: Since both the technology family and its first generation are referred to as "SEV", this guide adopts the following terminology for clarity:

- "Legacy SEV" refers specifically to the first generation of SEV for confidential VMs.
- "SEV refers to the technology family as a whole.

1.2 SECURITY FEATURES BY PROCESSOR GENERATION

AMD EPYC™ Processors have the following security features by generation:

- **AMD EPYC™ 9005**
  - Legacy Secure Encrypted Virtualization (SEV)
  - Encrypted State (ES)
  - Secure Nested Paging (SNP)
  - Trusted I/O (TIO)
  - Secure Virtual Machine Service Model (SVSM)
  - Transparent Secure Memory Encryption (TSME)
  - 1006 Address Space Identifier (ASID) Keys
- **AMD EPYC™ 9004 or AMD EPYC™ 8004**
  - Legacy SEV
  - ES
  - SNP
  - SVSM
  - TSME
  - 1006 ASID Keys
- **AMD EPYC™ 7003**
  - Legacy SEV
  - ES
  - SNP
  - SVSM
  - TSME
  - Configuration Options:
    - 509 ASID Keys (uses 9 address bits, allowing up to 8 TB of DRAM)
    - 253 ASID Keys (uses 8 address bits, allowing up to 16 TB of DRAM)
- **AMD EPYC™ 7002**
  - Legacy SEV
  - ES
  - Configuration Options:
    - 509 ASID Keys (uses 9 address bits, allowing up to 8 TB of DRAM)
    - 253 ASID Keys (uses 8 address bits, allowing up to 16 TB of DRAM)

CHAPTER 2

ENABLING/DISABLING SMEE

This chapter describes how to enable the AMD Secure Memory Encryption (SMEE) feature. SMEE must be enabled to use all SEV features. All the instructions shown in this chapter are based on AMD Custom Reference Boards (CRBs). The exact steps and images may vary by OEM and BIOS version.

2.1 ENABLING SMEE IN BIOS

This section describes how to enable SMEE on AMD EPYC™ Processors.

2.1.1 AMD EPYC™ 9005 SERIES PROCESSORS

SMEE is disabled by default on systems powered by AMD EPYC™ 9005 Series Processors because of incompatibility with certain Linux kernels. To enable SMEE:

1. Access your system **BIOS**:

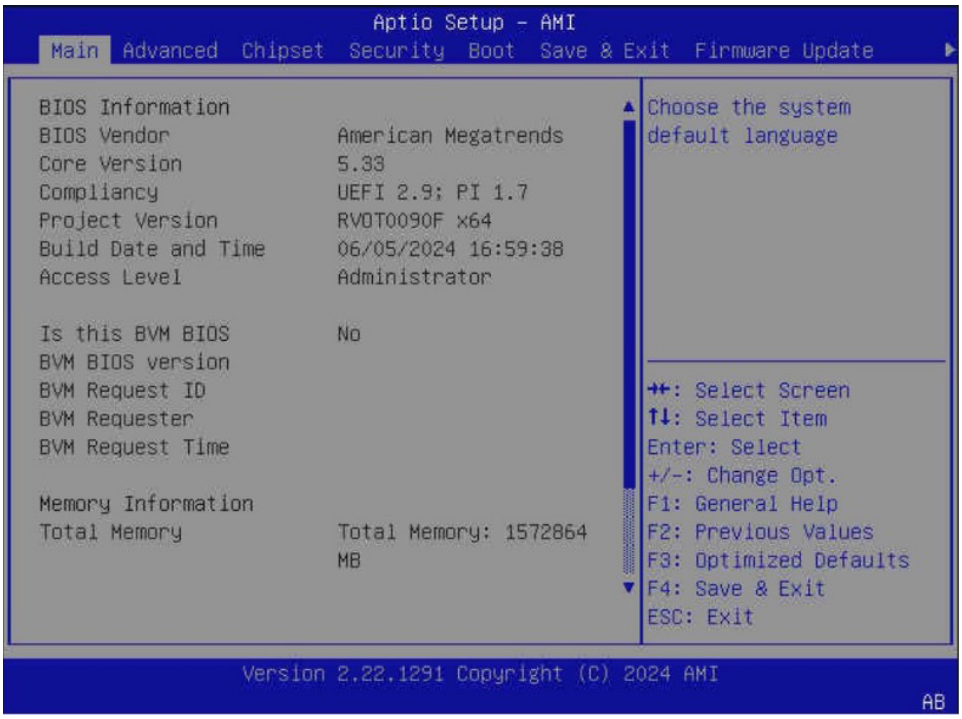

Figure 2-1: System BIOS (AMD EPYC™ 9005 Series Processors)

2. Select the **Advanced** Tab:

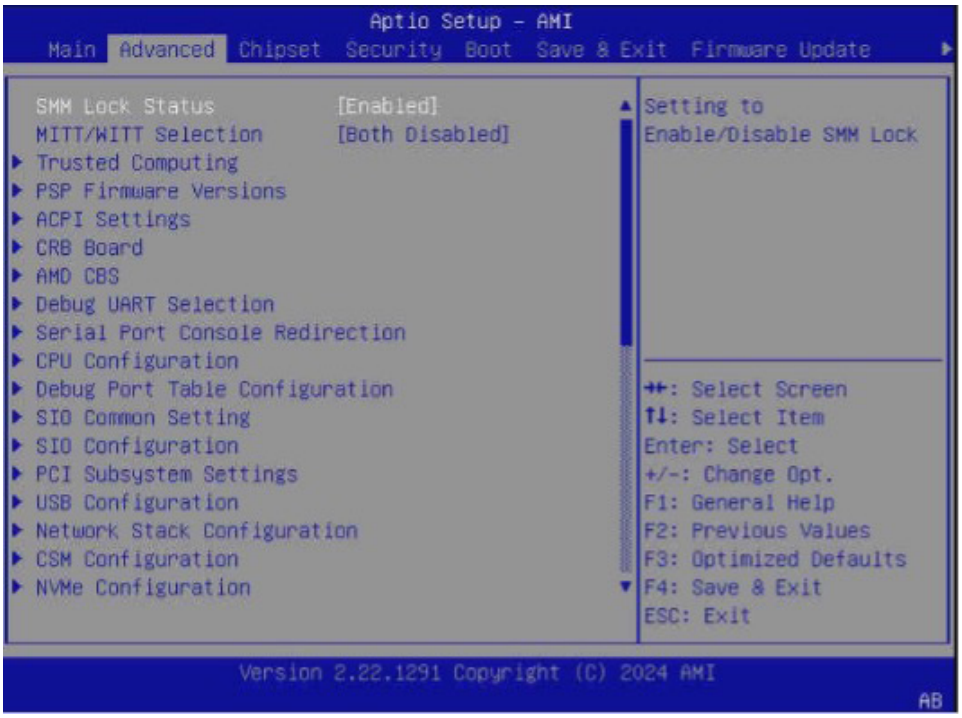

Figure 2-2: BIOS Advanced tab (AMD EPYC™ 9005 Series Processors)

3. Select **AMD CBS**:

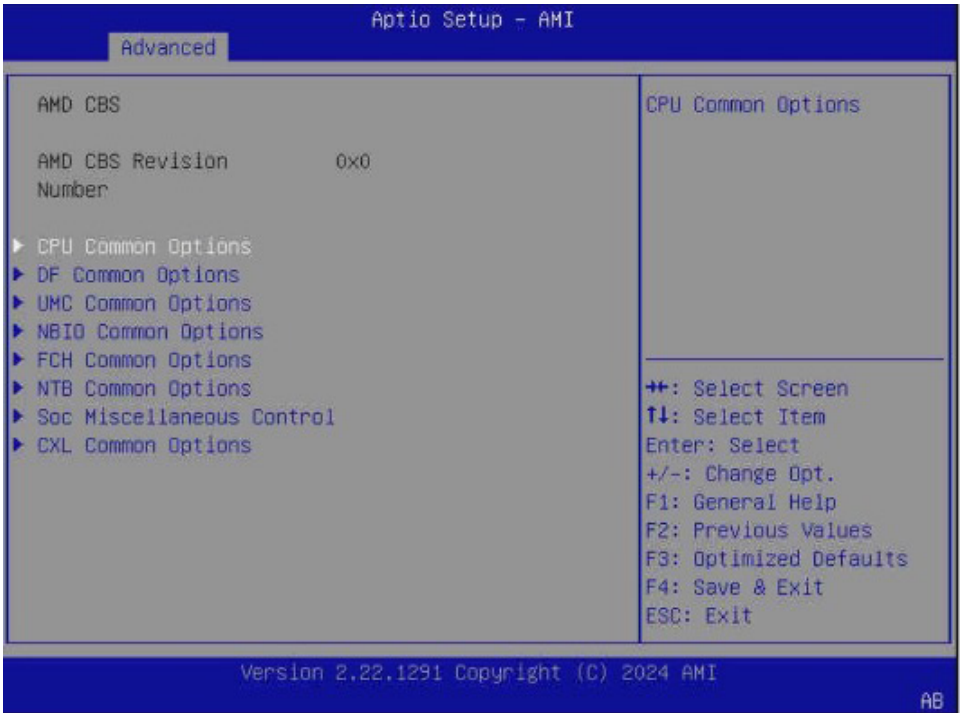

Figure 2-3: AMD CBS tab (AMD EPYC™ 9005 Series Processors)

4. Select **CPU Common Options**:

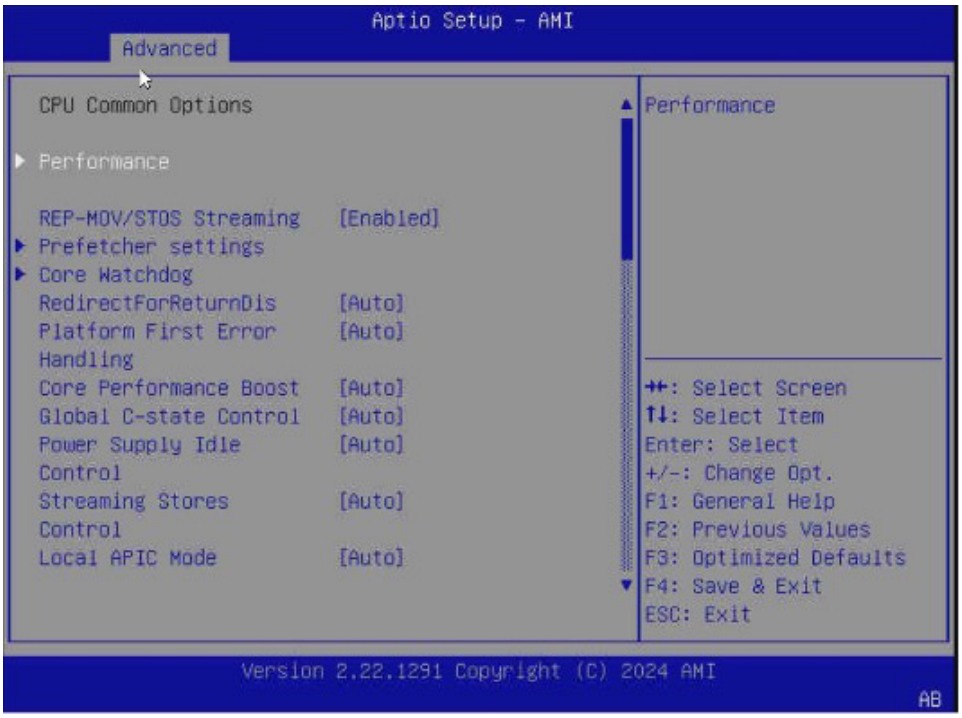

Figure 2-4: CPU Common Options tab (AMD EPYC™ 9005 Series Processors)

5. Scroll down this tab, then select **SMEE**, and then set it to **Enable**:

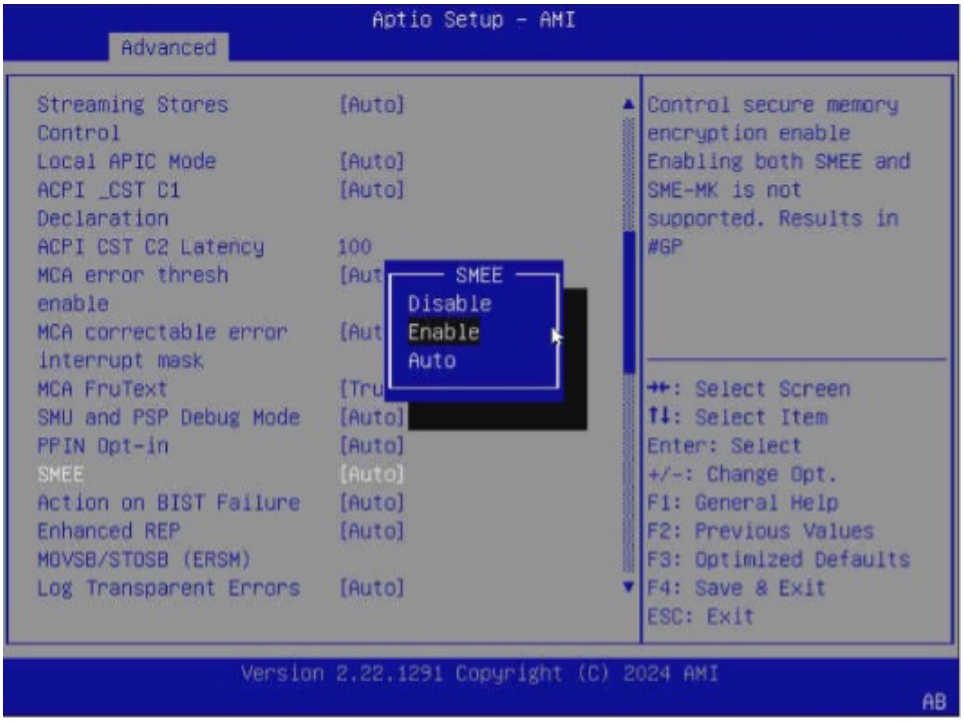

Figure 2-5: SMEE Enabled (AMD EPYC™ 9005 Series Processors)

2.1.2 AMD EPYC™ 9004 SERIES PROCESSORS

SMEE is disabled by default on systems powered by AMD EPYC™ 9004 Series Processors because of an incompatibility with kernels older than 5.6. To enable SMEE:

1. Access your system **BIOS**:

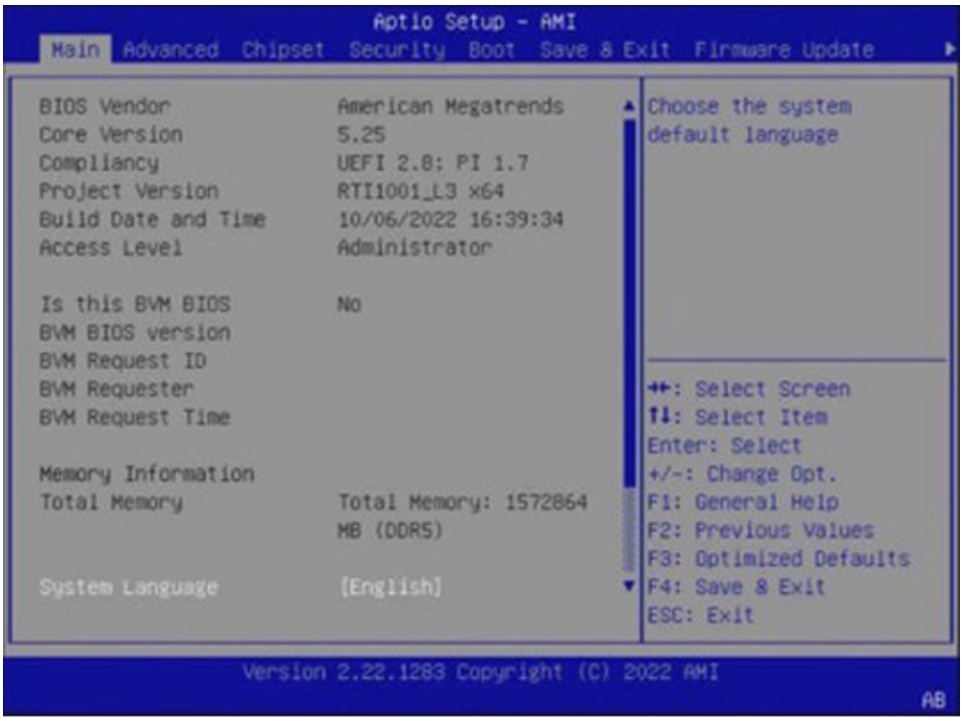

Figure 2-6: System BIOS (AMD EPYC™ 9004 Series Processors)

2. Select the **Advanced** tab:

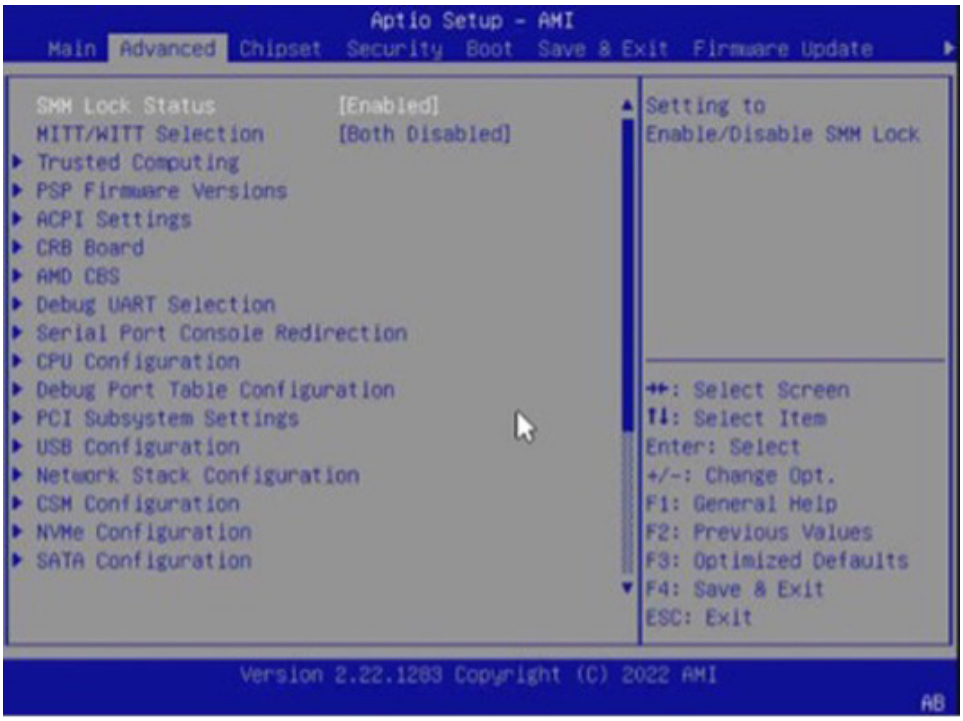

Figure 2-7: BIOS Advanced tab (AMD EPYC™ 9004 Series Processors)

3. Select **AMD CBS**:

Figure 2-8: AMD CBS tab (AMD EPYC™ 9004 Series Processors)

4. Select **CPU Common Options**:

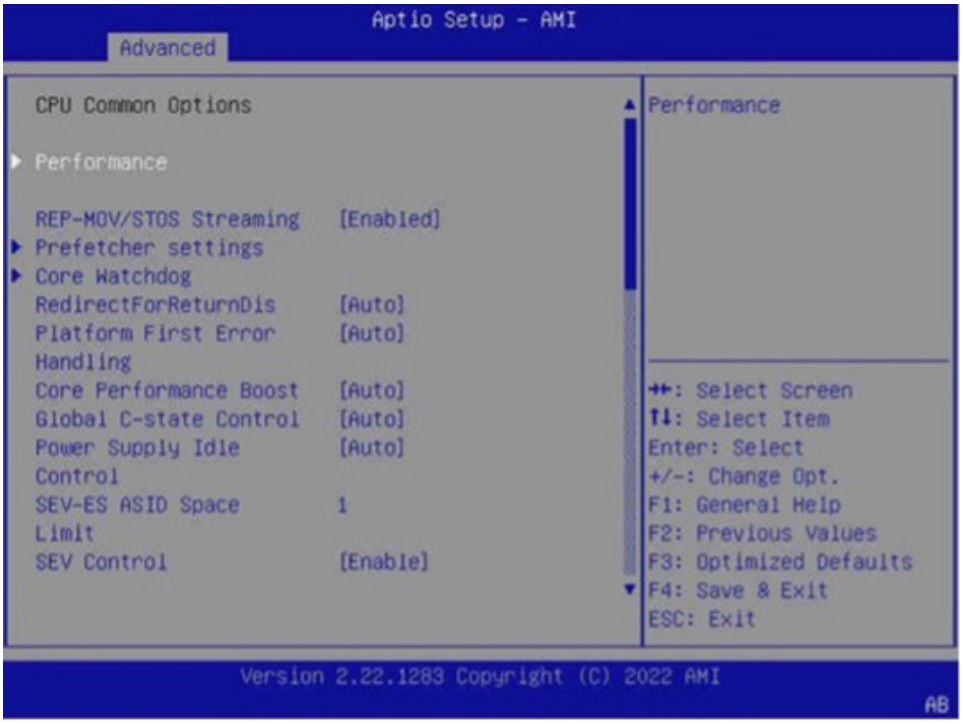

Figure 2-9: CPU Common Options tab (AMD EPYC™ 9004 Series Processors)

5. Scroll down this tab, then select **SMEE**, and then set it to **Enable**:

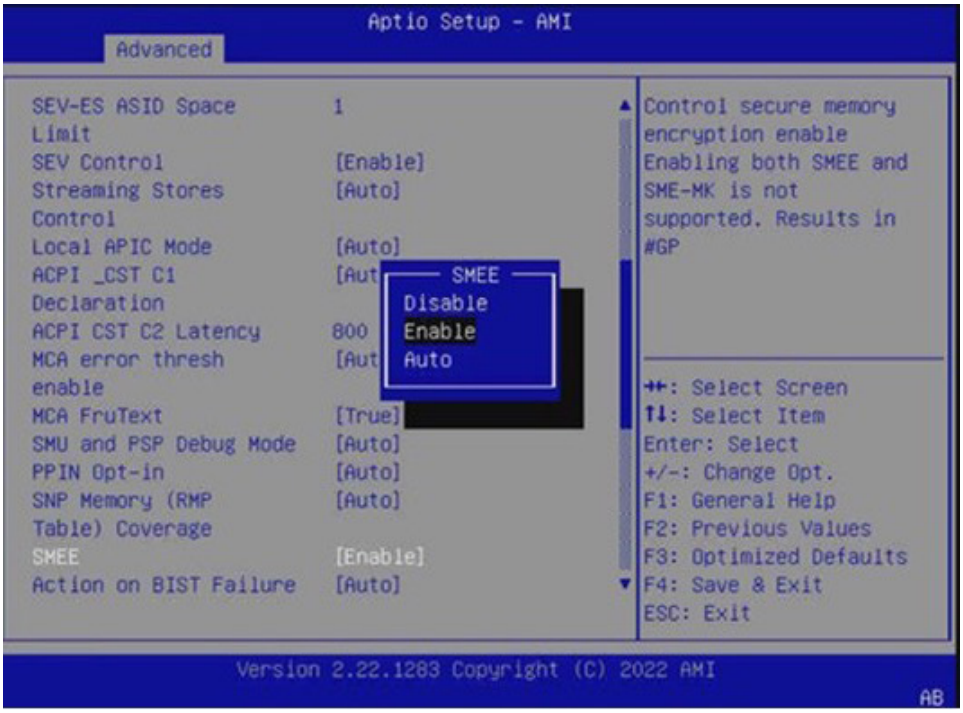

Figure 2-10: SMEE enabled (AMD EPYC™ 9004 Series Processors)

2.1.3 AMD EPYC™ 7003 SERIES PROCESSORS

SMEE is disabled by default on systems powered by AMD EPYC™ 7003 Series Processors because of an incompatibility with kernels older than 5.6. To enable SMEE:

1. Access your system **BIOS**:

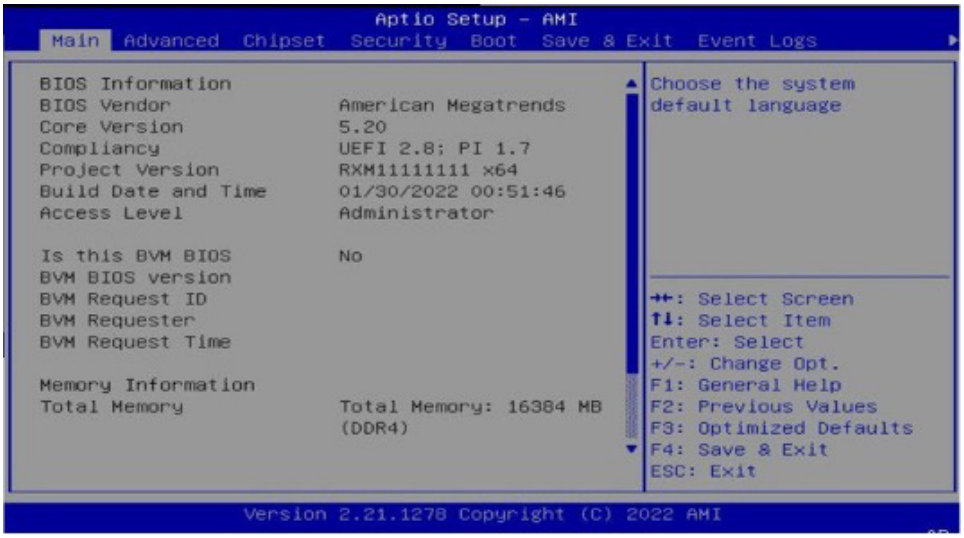

Figure 2-11: System BIOS (AMD EPYC™ 7003 Series Processors)

2. Select the **Advanced** tab:

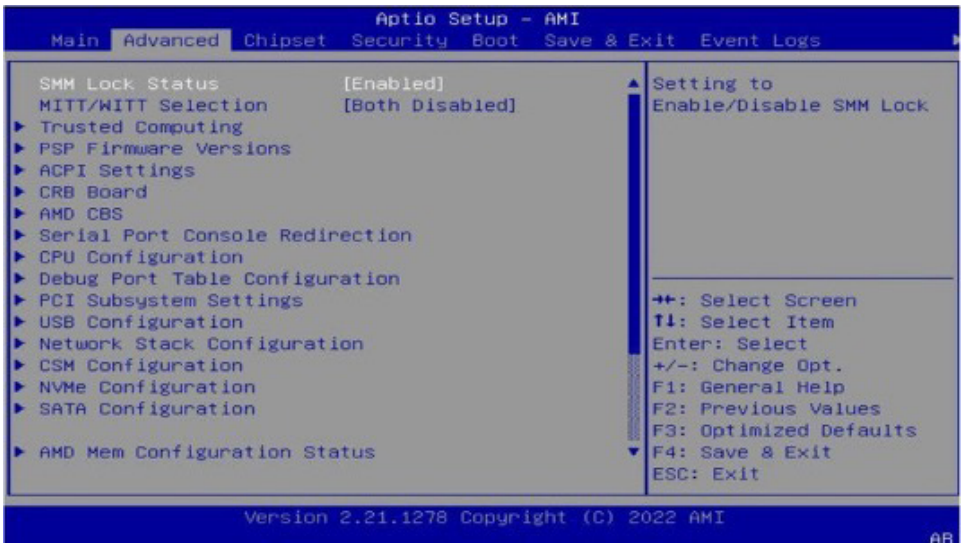

Figure 2-12: BIOS Advanced tab (AMD EPYC™ 7003 Series Processors)

3. Select **AMD CBS**

Figure 2-13: AMD CBS tab (AMD EPYC™ 7003 Series Processors)

4. Select **CPU Common Options**:

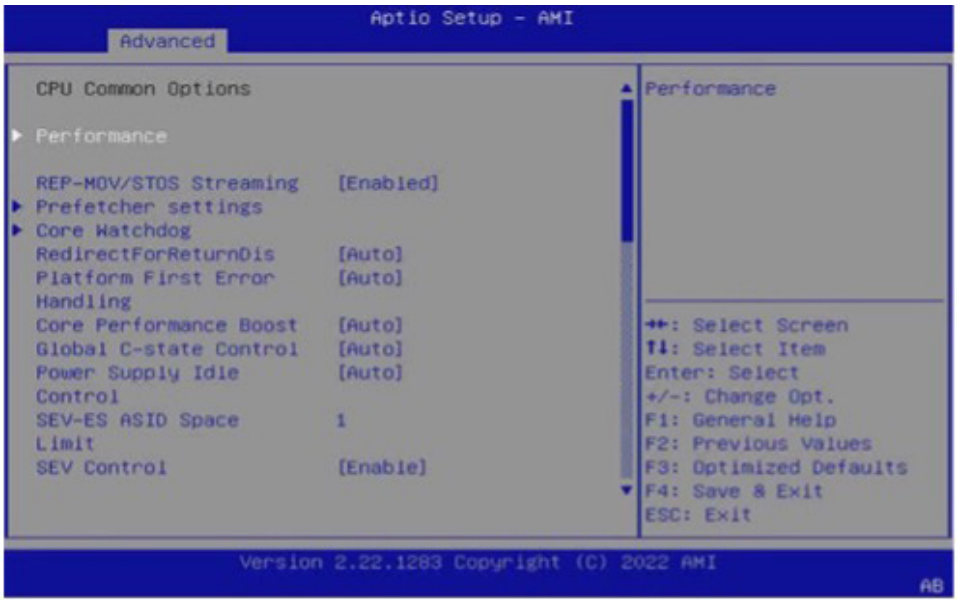

Figure 2-14: CPU Common Options tab (AMD EPYC™ 7003 Series Processors)

5. Scroll down this tab, then select **SMEE**, and then set it to **Enable**:

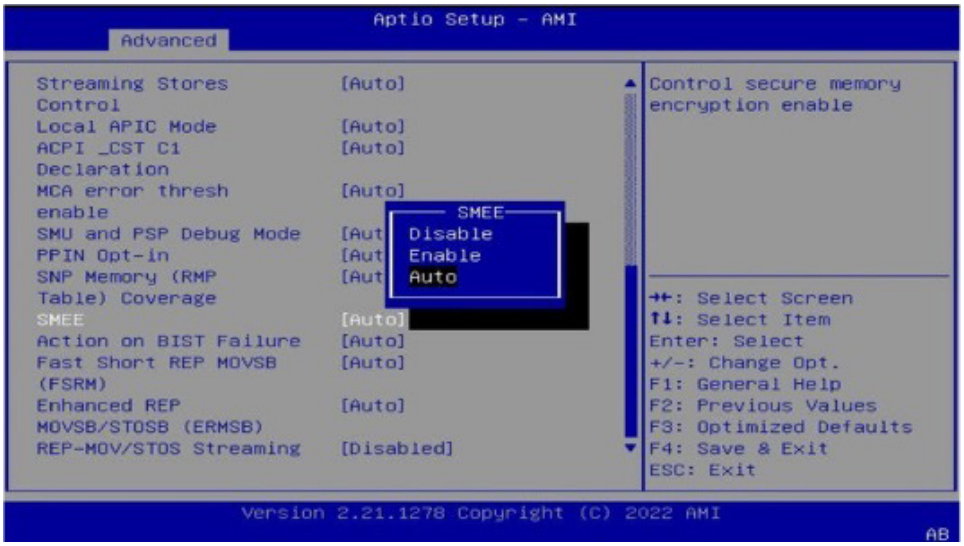

Figure 2-15: SMEE enabled (AMD EPYC™ 7003 Series Processors)

2.1.4 AMD EPYC™ 7002 AND 7001 SERIES PROCESSORS

SMEE is **Enabled** by default on systems powered by AMD EPYC™ 7002 and 7001 Series Processors.

2.2 ENABLING SMEE VIA MSR

To enable SMEE via the processor MSR:

- x86 can set the SMEE bit (bit 23) in the **`SYS_CFG MSR`** before OS boot.
- **`MSRC001_0010 [System Configuration] (Core::X86::Msr::SYS_CFG)`**
- EDK2-based BIOS (non-CBS users) should specifically toggle this bit to enable/disable Legacy SEV if a reciprocal PCD method is not available for that processor family.

Note: This bit must be set on every CPU in the system.
Note: The bit is Write-1-Only, which (cannot be cleared once set, and which is set to 0 on system reset).
Note: AMD EPYC™ 7001 and 7002 Series Processors have SMEE enabled automatically. If SMEE is disabled in BIOS, then you can use MSR to reenable SMEE in the system.

2.3 DISABLING SMEE IN BIOS

This section describes disabling SMEE on AMD EPYC™ Processors.

2.3.1 AMD EPYC™ 9005, 9004 AND 7003 SERIES PROCESSORS

To disable SMEE on a system with an AMD EPYC™ 9004 Series Processor:

1. Access your system **BIOS**.
2. Select the **Advanced** tab.
3. Select **AMD CBS**.
4. Select **CPU Common Options**.
5. Scroll down this tab, then select **SMEE**, and then set it to either **Auto** or **Disabled**.

2.3.2 AMD EPYC™ 7002 SERIES PROCESSORS

You cannot disable SMEE on a system with an AMD EPYC 7002 Series Processor.

2.4 ENABLING/DISABLING SMEE VIA MSR

For people interested in using MSRs, (i.e. writing personal kernel implementations) these are the registers you can configure to enable and disable SMEE:

- x86 can set the SMEE bit (bit 23) in the **`SYS_CFG MSR`** before OS boot.
- **`MSRC001_0010 [System Configuration] (Core::X86::Msr::SYS_CFG)`**
- EDK2-based BIOS (non-CBS users) should specifically toggle this bit to enable/disable Legacy SEV if a reciprocal PCD method is not available for that processor family.

Note: This bit must be set on every CPU in the system.
Note: The bit is Write-1-Only, which (cannot be cleared once set, and which is set to 0 on system reset).
Note: AMD EPYC™ 7001 and 7002 Series Processors have SMEE enabled automatically. If SMEE is disabled in BIOS, then you can use MSR to reenable SMEE in the system.

SMEE cannot be disabled in the MSR; the bit is Write-1-Only. You must either reset the system or disable SMEE in BIOS.

Note: Disabling SEV will allow the use of more than 16TB of system physical address space (DRAM + PCIe + MMIO, etc.) because x bits of physical address space will not be used for ASIDs/c-bit.

- AMD EPYC™ 9005 Series Processors: 52-bit addressing with no c-bit, SMEE/SEV off.
  - 46-bit address with SEV (1006 keys).
- AMD EPYC™ 9004 Series Processors: 52-bit addressing with no c-bit, SMEE/SEV off.
  - 46-bit address with SEV (1006 keys).
- AMD EPYC™ 7003 Series Processors: 48-bit addressing with no c-bit, SMEE/SEV off.
  - 43-bit address with SEV in 509-key mode, 44-bit in 253 key mode.
- AMD EPYC™ 7002 Series Processors: 48-bit addressing with no c-bit, SMEE/SEV off.
  - 43-bit address with SEV in 509-key mode, 44-bit in 253 key mode.
- AMD EPYC™ 7001 Series Processors: 48-bit addressing with no c-bit, SMEE/SEV off.
  - 43-bit address with SME/SEV (16 keys).

2.5 ENABLING/DISABLING TSME ON ALL PROCESSORS

Transparent Secure Memory Encryption (TSME, also known as Secure Memory Encryption) uses a single key to encrypt system memory. The AMD Secure Processor generates this key at boot. TSME requires enablement in the system BIOS and offers transparent memory encryption that can run with any operating system. TSME is separate from SMEE, and you need not run SEV to benefit from TSME. TSME is disabled by default.

2.5.1 ENABLING TSME ON ALL PROCESSORS

TSME is disabled by default on systems powered by AMD EPYC™ 7003 Series Processors because of incompatibility with certain Linux kernels. To enable TSME:

1. Access your system **BIOS**.
2. Select **Advanced**

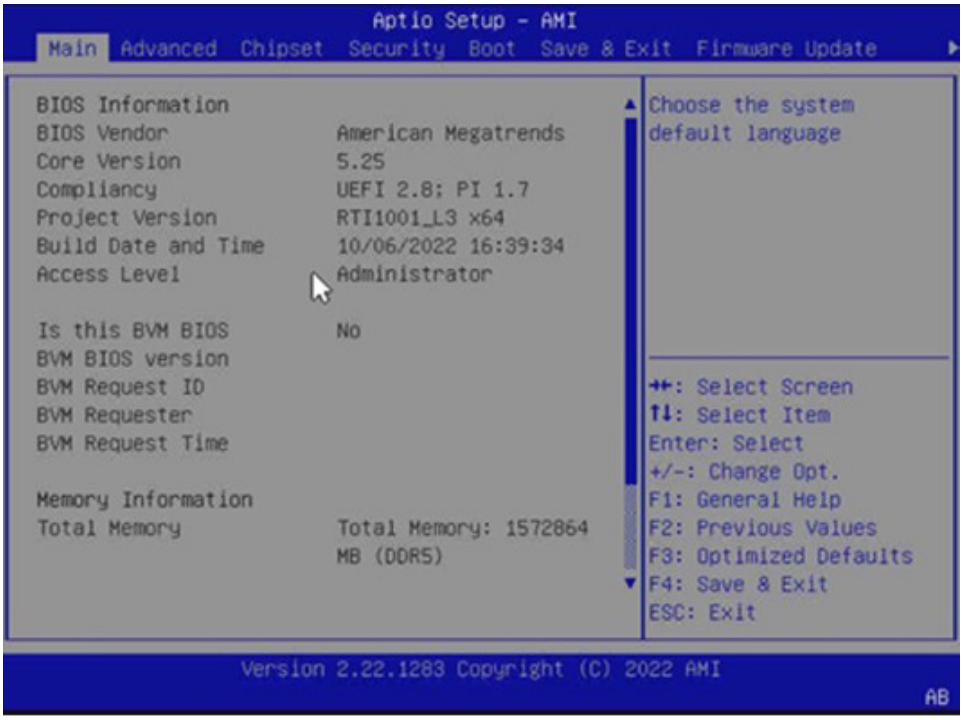

Figure 2-16: BIOS Landing page

3. Select **AMD CBS**:

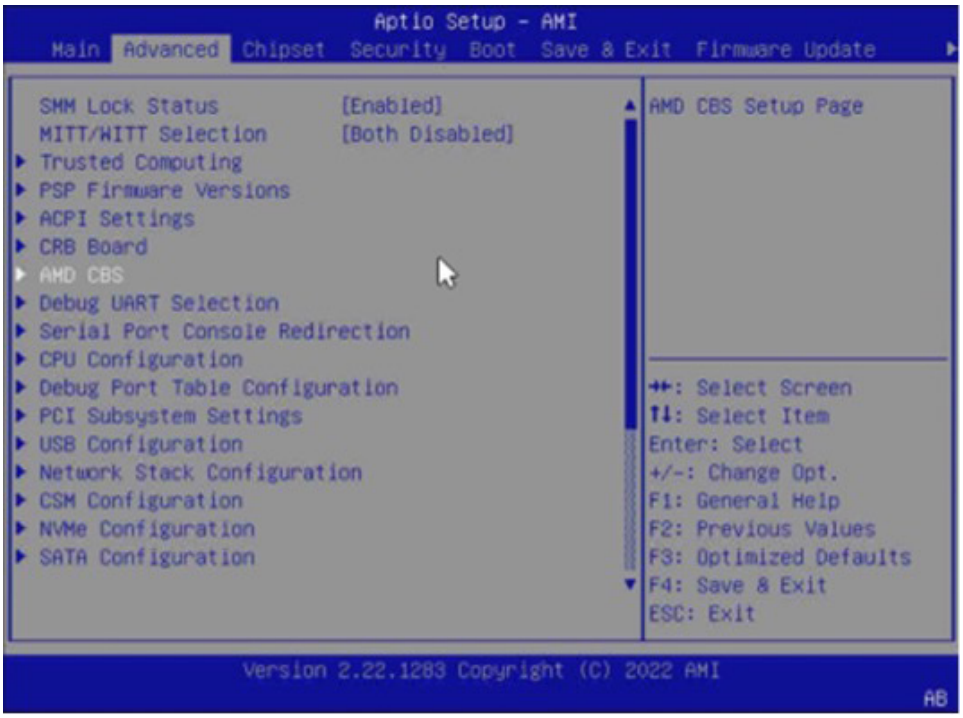

Figure 2-17: BIOS Advanced Tab

4. Select **UMC Common Options**:

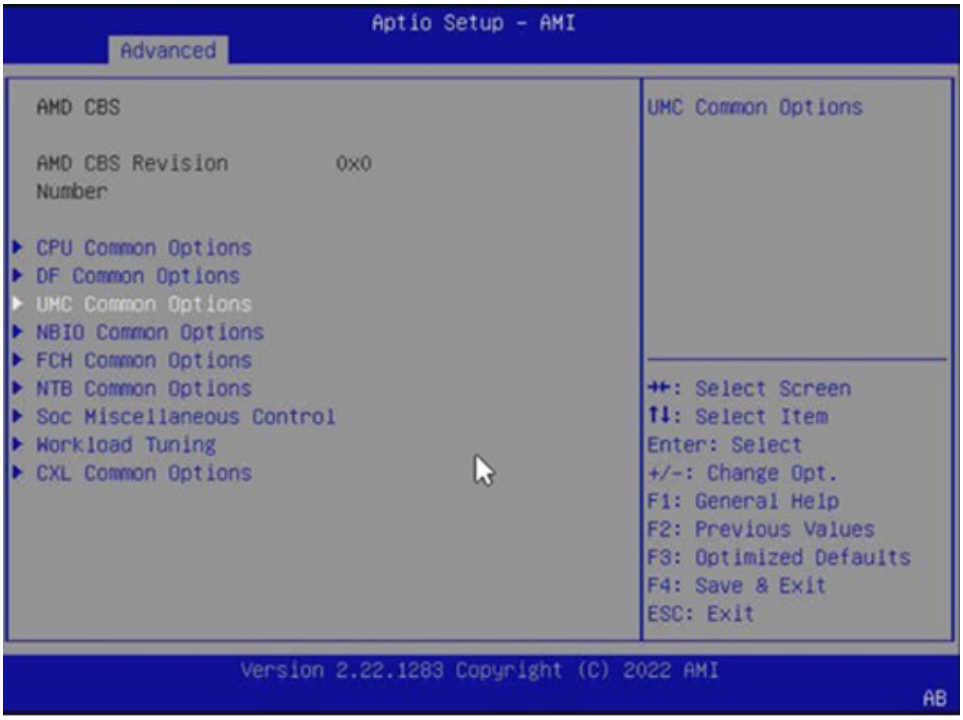

Figure 2-18: AMD CBS

5. Select **DDR Security**:

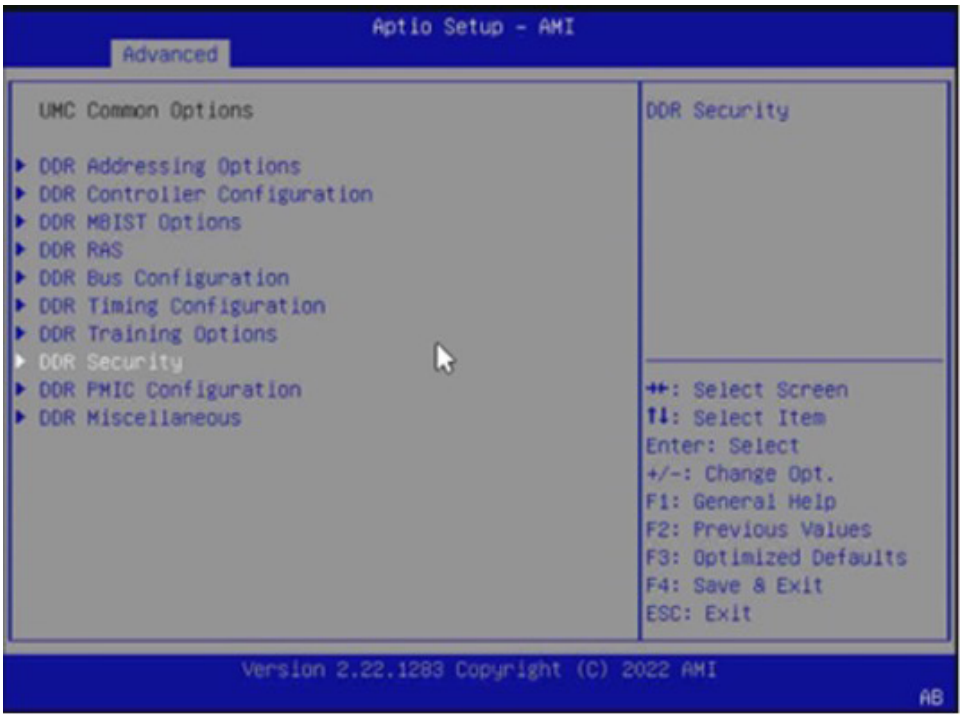

Figure 2-19: UMC Common Options

6. Select **TSME** to **Enabled**:

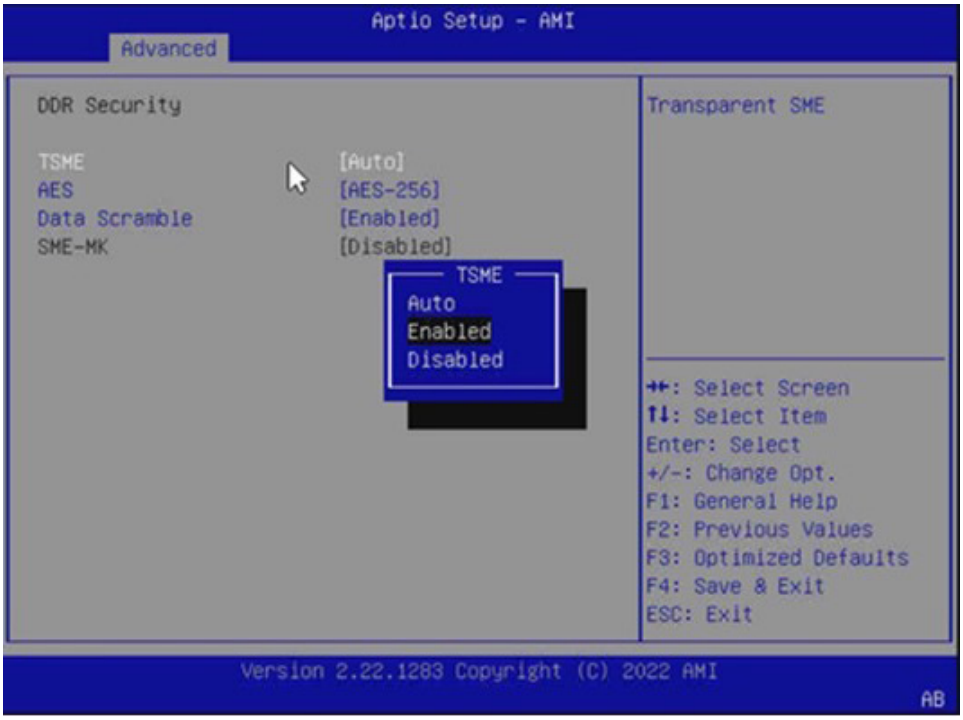

Figure 2-20: DDR TAB: Enable TSME

2.5.2 DISABLING TSME ON ALL PROCESSORS

To disable TSME on an AMD CRB:

- Access your system **BIOS**.
- Select **Advanced** > **AMD CBS** > **UMC Common Options** > **DDR Security**.
- Set **TSME** to either **Auto** or **Disabled**.

CHAPTER 3

CONFIGURING SNP

This guide prioritizes Secure Nested Paging (SNP) as the default virtualization security model for AMD EPYC™ Processors. While Legacy SEV and ES remain supported, SNP offers enhanced protection and is recommended for all new deployments.

This chapter explains how to enable and configure SNP. Because SNP builds upon Legacy SEV and ES, the initial steps will guide users through enabling those features as prerequisites.

Note: SMEE must be enabled to use any SEV-based feature, including SNP.

3.1 CONFIGURING SNP THROUGH BIOS:

3.1.1 AMD EPYC™ 9005 SERIES PROCESSORS

1. In **BIOS**, select **Advanced** > **AMD CBS** > **CPU Common Options**, and then set the **SEV Control** parameter to **Enable**. This will enable both Legacy SEV and ES:

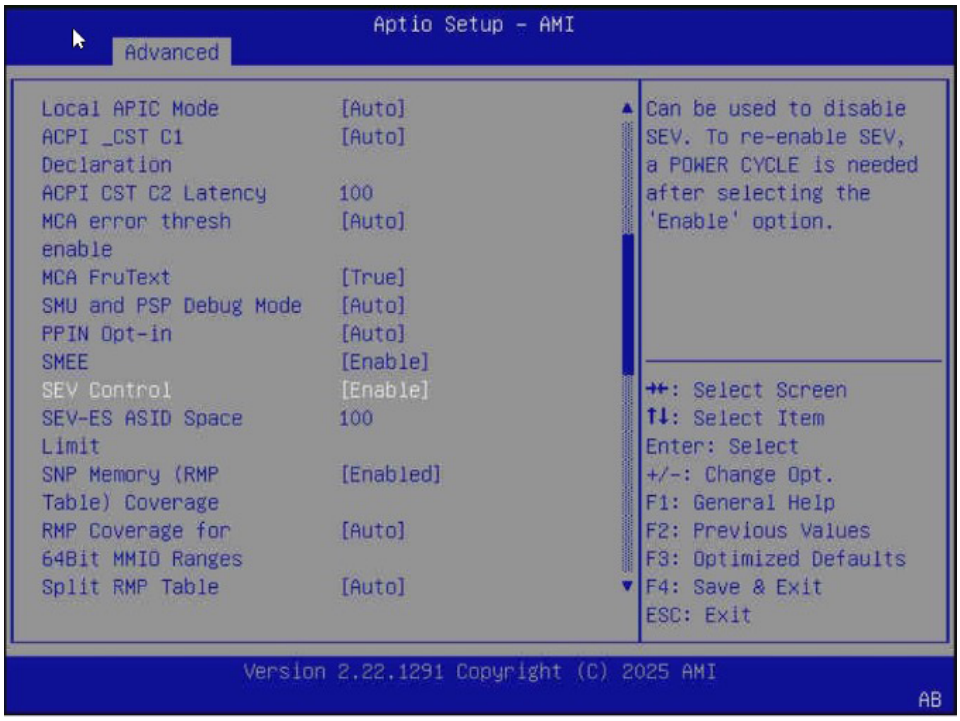

Figure 3-1: Setting SEV Control to Enable (AMD EPY™C 9005 Series Processors)

2. To change the maximum number of ES/SNP ASIDs, select **Advanced** > **AMD CBS** > **CPU Common Options** and then change the **SEV-ES ASID** Count from **Auto** (1006) to 1006 or below. Set **SEV- ES ASID Space Limit** to the desired value based on the types of VMs you will be running. ASIDs less than 'x' are for ES and SNP, and ASIDs greater than or equal to 'x' are for Legacy SEV only. For example, if 5 is input in the field, then there will be 4 available ES and SNP ASIDs and the rest will be Legacy SEV only. If the field is set to 1, then ES and SNP will be disabled because there are no available ASIDs for ES or SNP.

   See the **minSEVASID** question in **"FREQUENTLY ASKED QUESTIONS"** for more information.

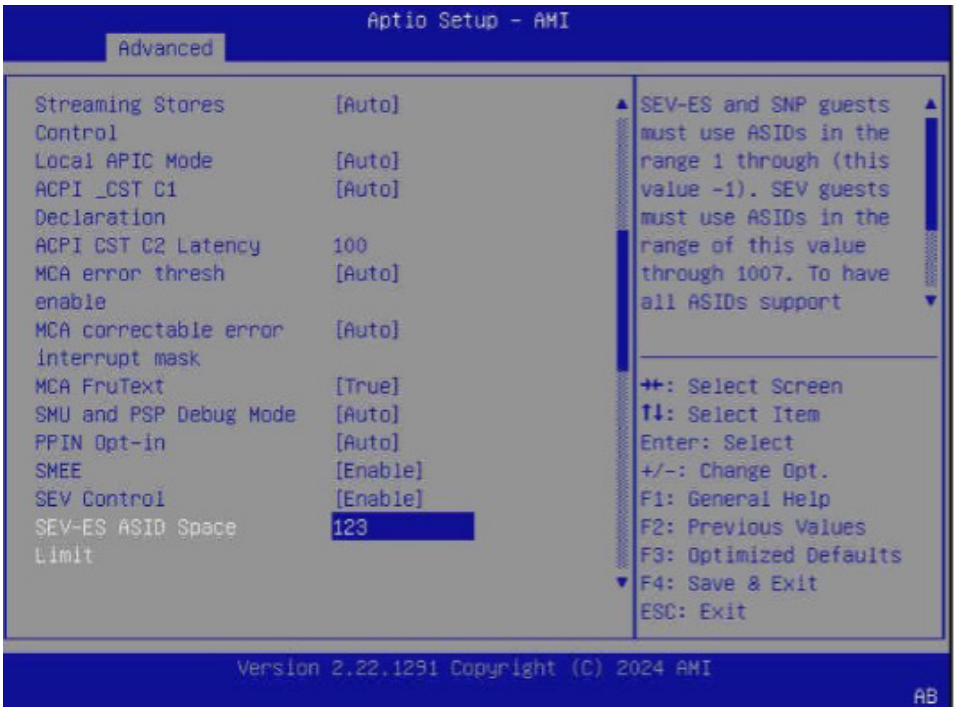

Figure 3-2: Configuring SEV-ES ASID Space Limit (AMD EPYC™ 9005 Series Processors)

3. Change **SNP Memory (RMP Table) Coverage** from **Auto** (which means **Disabled**) to **Enabled**. This will reserve memory for the SNP RMP Table that will cover/protect all of system memory. If needed, you can select Custom to set the RMP to not cover all of memory:

   Note: This is required for Linux hosts. Microsoft hosts do not require this when using SEV-SNP under Hyper-V.

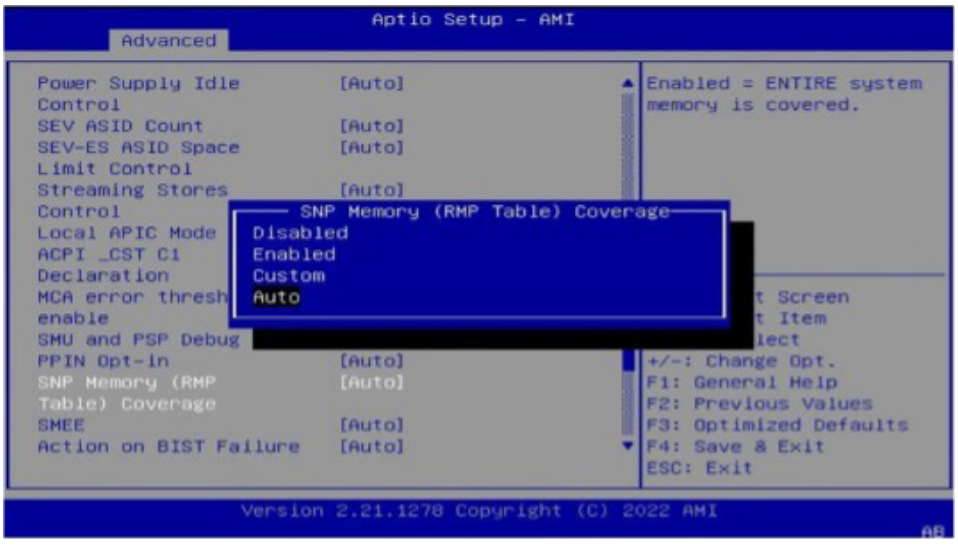

Figure 3-3: Changing SNP Memory (RMP Table) Coverage options (AMD EPYC™ 9005 Series Processors)

4. Go back to the **BIOS** for main page and then to **Advanced** > **AMD CBS** > **NBIO Common Options** > **IOMMU/SECURITY**:

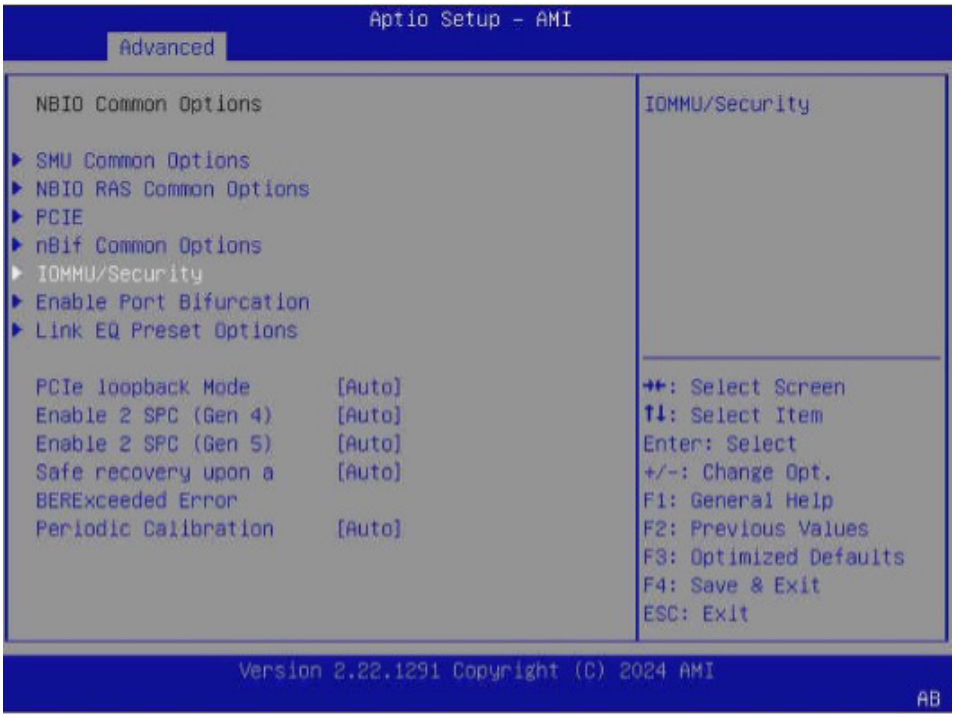

Figure 3-4: IOMMU/SECURITY tab in NBIO common options (AMD EPYC™ 9005 Series Processors)

5. Set **IOMMU** and **SEV-SNP Support** to **Enabled**:

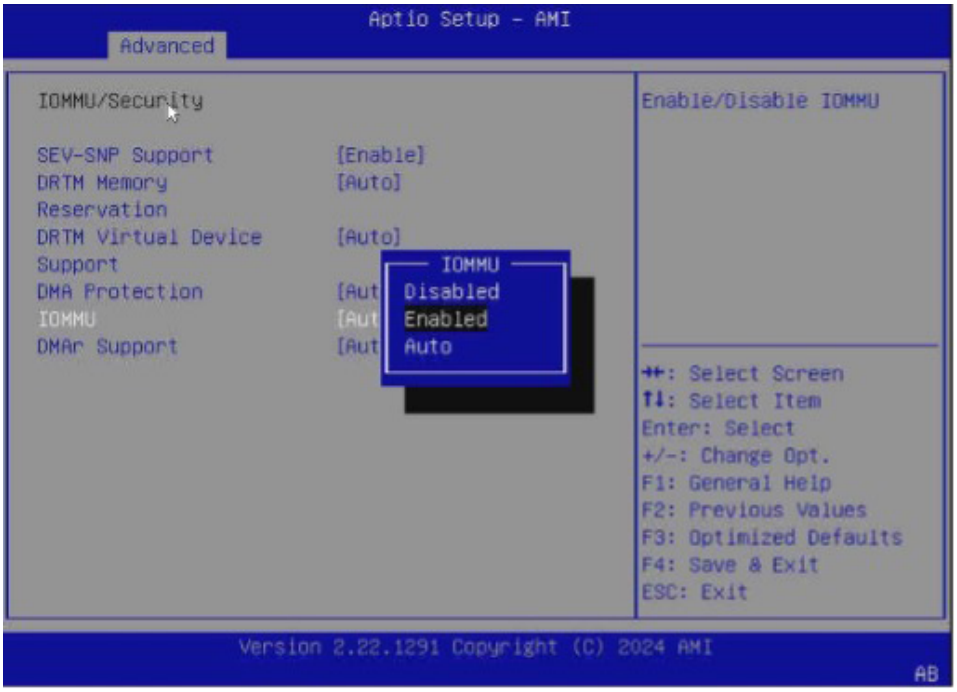

Figure 3-5: Enable IOMMU and SEV-SNP in IOMMU/Security (AMD EPYC™ 9005 Series Processors

3.1.2 AMD EPYC™ 9004 SERIES PROCESSORS

1. In **BIOS**, select **Advanced** > **AMD CBS** > **CPU Common Options**, and then set the **SEV Control** parameter to **Enable**. This will enable both Legacy SEV and ES.

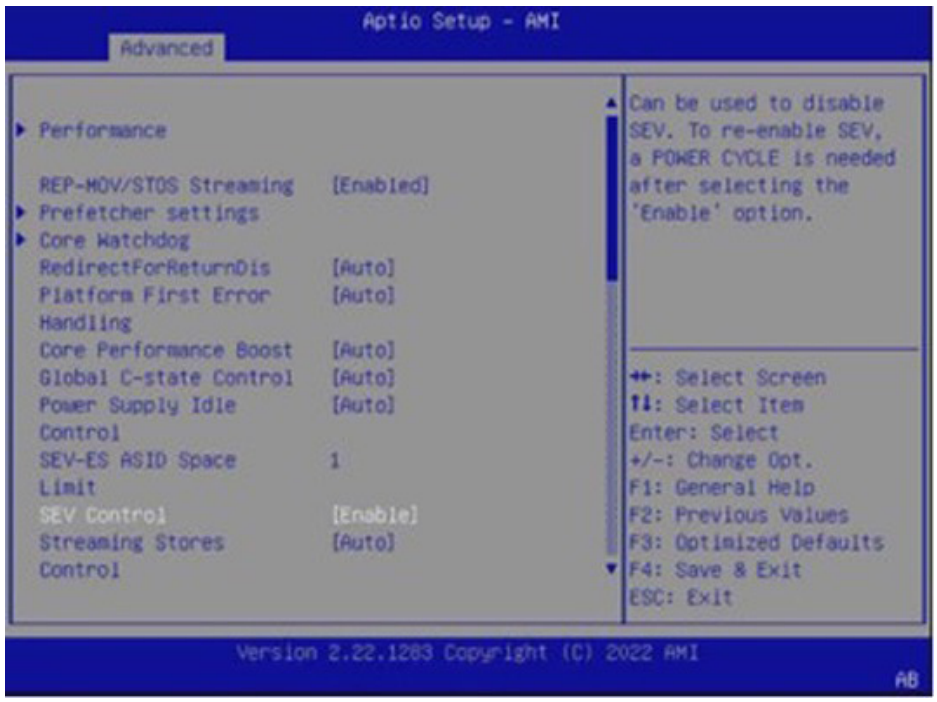

Figure 3-6: Setting SEV-ES Control to Enabled (AMD EPYC™ 9004 Series Processors)

2. To change the maximum number of ES/SNP ASIDs, select **Advanced** > **AMD CBS** > **CPU Common Options** and then change the **SEV-ES ASID Count** from **Auto** (1006) to 1006 or below. Set **SEV- ES ASID Space Limit** to the desired value based on the types of VMs you will be running. ASIDs less than 'x' are for ES and SNP, and ASIDs greater than or equal to 'x' are for Legacy SEV only. For example, if 5 is input in the field, then there will be 4 available ES & SNP ASIDs and the rest will be Legacy SEV only. If the field is set to 1, then ES and SNP will be disabled because there are no available ASIDs for ES or SNP.

   See the **minSEVASID** question in **"FREQUENTLY ASKED QUESTIONS"** for more information.

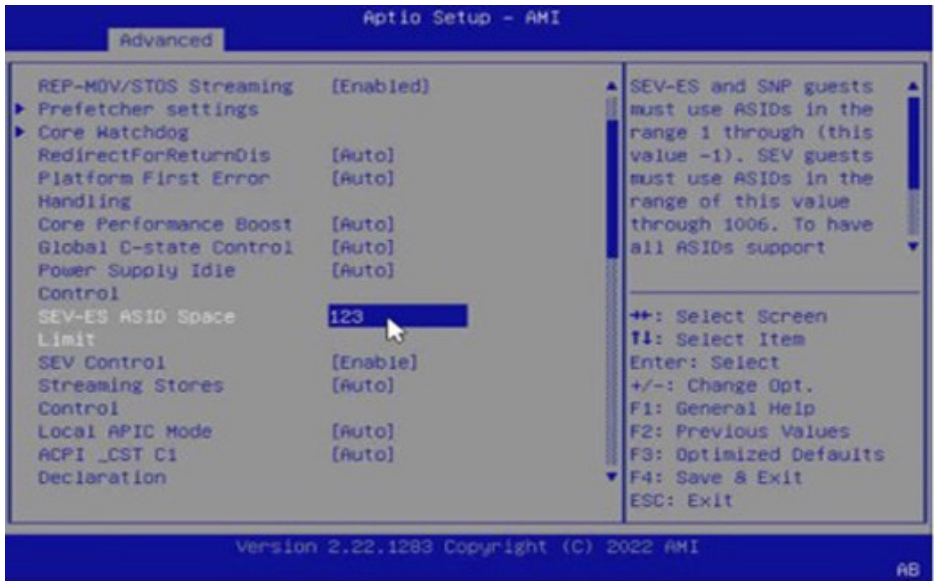

Figure 3-7: Configuring SEV-ES ASID Space Limit (AMD EPYC 9004 Series Processors)

3. Change **SNP Memory (RMP Table) Coverage** from **Auto** (which means **Disabled**) to **Enabled**. This will reserve memory for the SNP RMP Table that will cover/protect all of system memory. If needed, you can select Custom to set the RMP to not cover all of memory.

   Note: This is required for Linux hosts. Microsoft hosts do not require this when using SEV-SNP under Hyper-V.

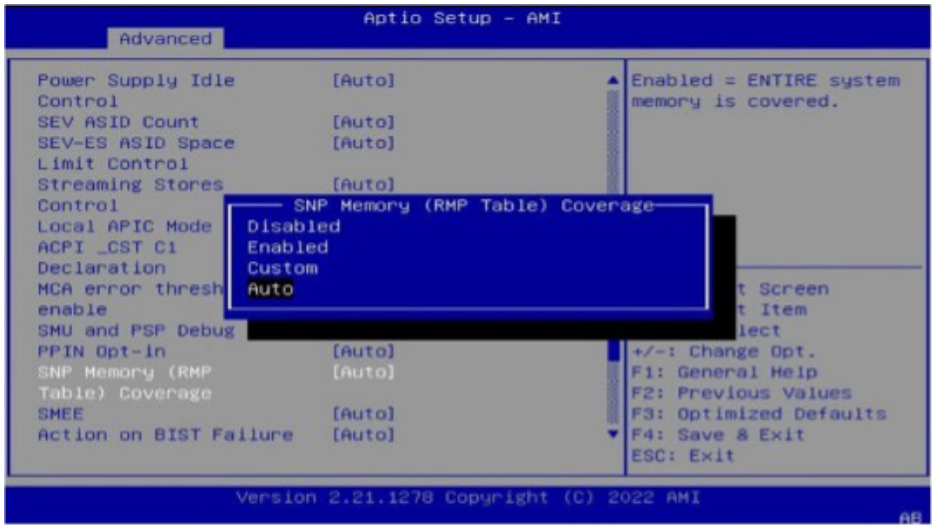

Figure 3-8: Changing SNP Memory (RMP Table) Coverage (AMD EPYC 9004 Series Processors)

4. Go back to the BIOS main page then to **Advanced** > **AMD CBS** > **NBIO Common Options**.

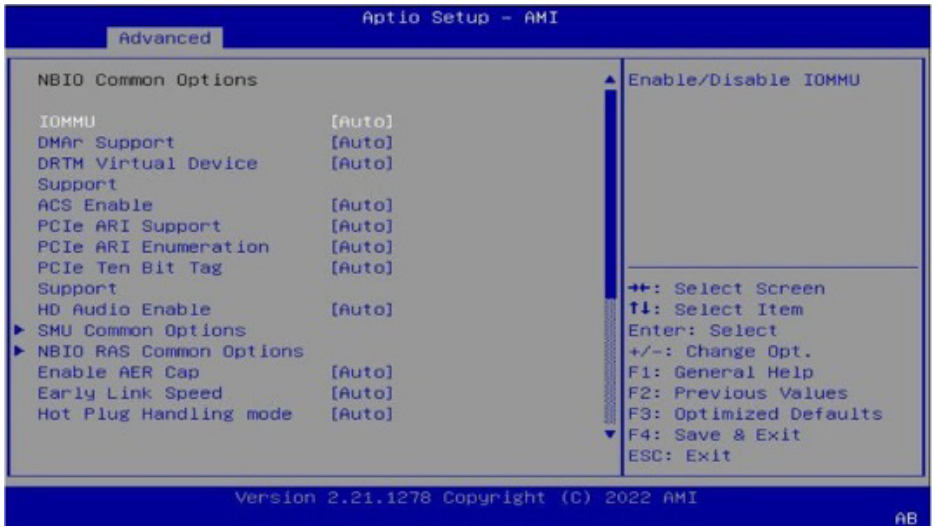

Figure 3-9: NBIO Common Options (AMD EPYC™ 9004 Series Processors)

5. Set **IOMMU** and **SEV-SNP Support** to Enabled.

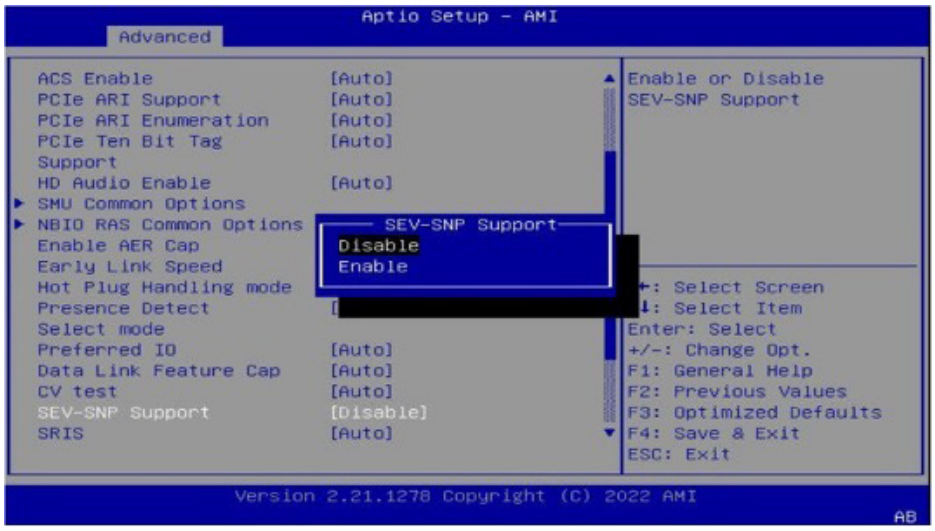

Figure 3-10: Enable SEV-SNP Support in NBIO Common Options (AMD EPYC™ 9004 Series Processors)

3.1.3 AMD EPYC™ 7003 SERIES PROCESSORS

To configure SNP on a system powered by an AMD EPYC™ 7003 Series Processor:

1. In **BIOS**, select **Advanced** > **AMD CBS** > C**PU Common Options**, and then set the **SEV-ES ASID Space Limit Control** parameter to **Manual**.

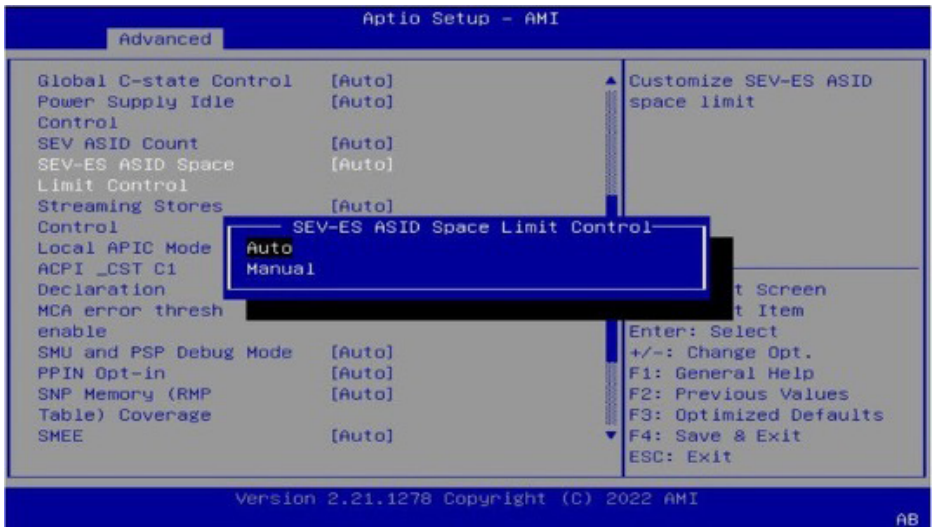

Figure 3-11: Setting SEV-ES Space Limit Control to Manual (AMD EPYC™ 7003 Series Processors)

2. Select **Advanced** > **AMD CBS** > **CPU Common Options**, and then change the **SEV-ES ASID Count** from **Auto** (509/253) to 509/253. Set **SEV-ES ASID Space Limit** to the desired value based on the types of VMs you will be running. ASIDs less than 'x' are for ES and SNP, and ASIDs greater than or equal to 'x' are for Legacy SEV. For example, if 5 is input in the field, then there will be 4 available ES and SNP ASIDs and the rest will be Legacy SEV only. If the field is set to 1, then ES and SNP will be disabled because there are no available ASIDs for ES or SNP.

   In AMD EPYC™ 7003 systems (and 7002), the total DRAM capacity and SEV ASID count share the same physical address space. To support configurations exceeding 8 TB of addressable DRAM, the SEV ASID count must be reduced to 253.
   AGESA automatically enforces this limit at boot if it detects more than 8 TB of installed memory.

   See the **minSEVASID** question in **"FREQUENTLY ASKED QUESTIONS"** for more information.

   Note: If the system detects 8TB or more of DRAM, then BIOS will automatically switch this to 253 ASIDs.

Figure 3-12: Configuring SEV-ES ASID Space Limit (AMD EPYC™ 7003 Series Processors)

3. Change **SNP Memory (RMP Table) Coverage** from **Auto** (which means **Disabled**) to **Enabled**. This will reserve memory for the SNP RMP Table that will cover/protect all of system memory. If needed, you can select Custom to set the RMP to not cover all of memory.

   Note: This is required for Linux hosts. Microsoft hosts do not require this when using SEV-SNP under Hyper-V.

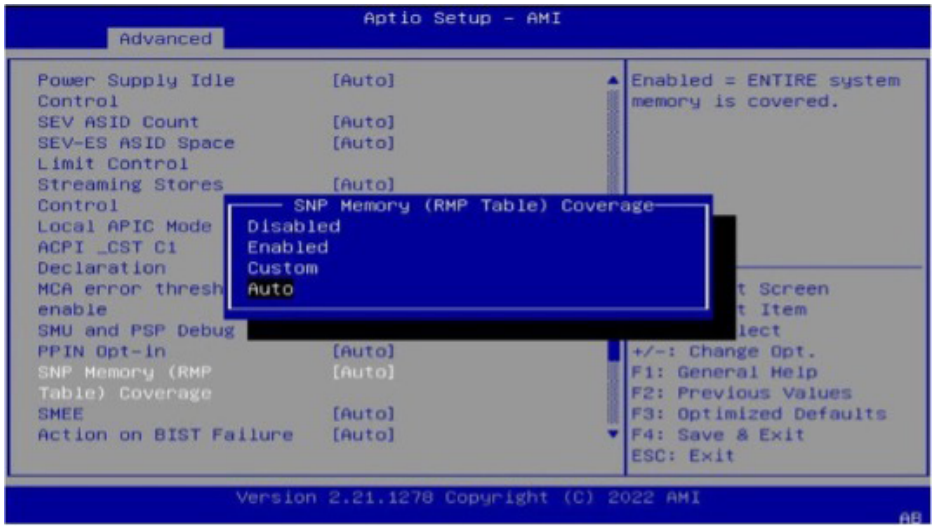

Figure 3-13: Changing SNP Memory (RMP Table) Coverage (AMD EPYC™ 7003 Series Processors)

4. Go back to the BIOS main page then to **Advanced** > **AMD CBS** > **NBIO Common Options**.

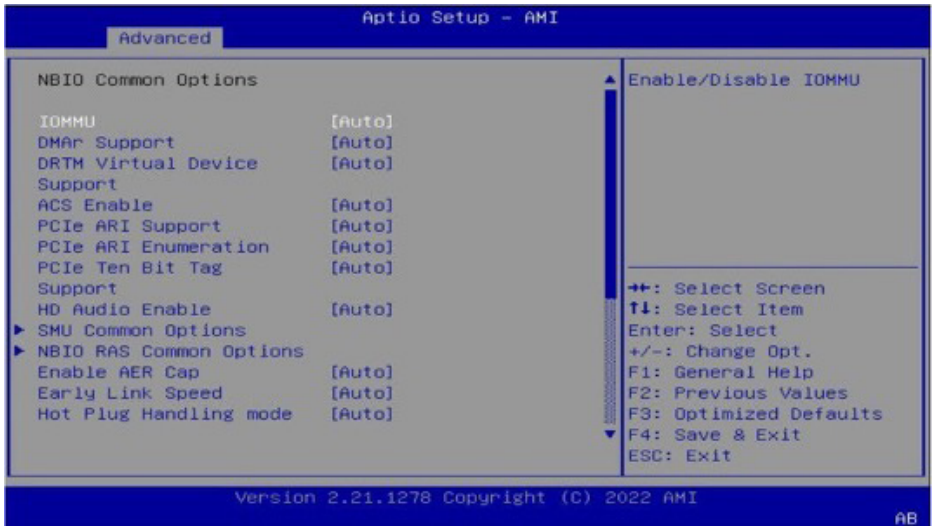

Figure 3-14: NBIO Common Options (AMD EPYC™ 7003 Series Processors)

5. Set **IOMMU** and **SEV-SNP Support** to **Enabled**.

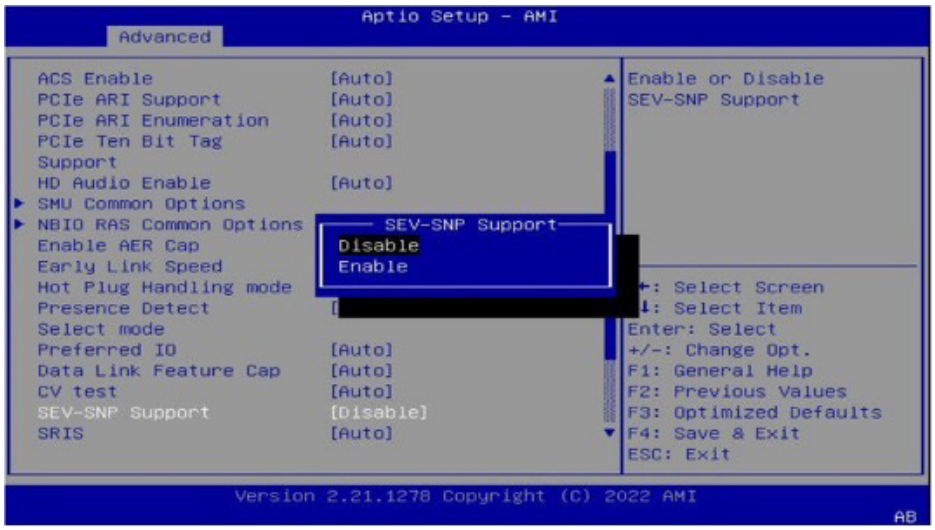

Figure 3-15: Enable SEV-SNP Support in NBIO Common Options (AMD EPYC™ 7003 Series Processors)

3.1.4 AMD EPYC™ 7002 SERIES PROCESSORS

AMD EPYC ™7002 Series Processors do not support SNP; users can follow steps 1-2 from the **"AMD EPYC™ 7003 Series Processors" section 3.1.3** instructions to enable Legacy SEV and ES in their systems.

3.2 DISABLING SNP

Set the **SNP Memory (RMP Table) Coverage** to **Auto/Disabled** in the **BIOS**. Also **Disable SEV-SNP support** in the **NBIO settings** in BIOS. No need to **DISABLE IOMMU** in **BIOS** to disable SNP.

AMD recommends leaving **SNP Memory (RMP Table) Coverage** set to **Auto/Disabled** in the **BIOS**, but there is no harm in leaving it Enabled if the hypervisor eventually wants to enable SNP. When enabled, firmware allocates a Reverse Map Table (RMP) entry for every 4 KiB page of system DRAM. This allows any portion of system memory to be used for SNP protected guests. Enabling full memory coverage reserves approximately 0.4 % of total system DRAM for the RMP and related metadata. This reserved region is not available for the operating system but has no other impact if SNP is not in use.

See the **RMP question in "FREQUENTLY ASKED QUESTIONS"** for more detailed information.

3.3 ENABLING/DISABLING SNP USING MSRS

You can also enable and disable SNP using MSRs. Like SMEE, this is intended for people planning on writing their own kernel implementations.

Before enabling SNP, first zero the RMP memory, and then write the address of the memory into the MSRs:

- **MSRC001_0132 [RMP Base](Core::X86::Msr::LS_RMP_BASE)**
- **MSRC001_0133 [RMP End](Core::X86::Msr::LS_RMP_END)**

Enable SNP by setting the following MSR bits to 1:

- **MSRC001_0010 [System Configuration] (Core::X86::Msr::SYS_CFG)** bit 19 MtrrFixDramModEn
- **MSRC001_0010 [System Configuration] (Core::X86::Msr::SYS_CFG)** bit 24 SecureNestedPagingEn
- **MSRC001_0010 [System Configuration](Core::X86::Msr::SYS_CFG)** bit 25 VmplEn

Please see Sections 15.26.4 and 15.36.1 in Volume 2 of the AMD Architecture Programmer's Manual for more information on RMP programming.

To disable SNP using MSRs:
Do not enable the **SecureNestedPagingEn** MSR bit: **MSRC001_0010 [System Configuration] (Core::X86::Msr::SYS_CFG)** bit 24 via x86.

Note: The system BIOS will never enable SecureNestedPagingEn. It always must be enabled by x86.

The RMP_BASE and RMP_END settings do not matter when SNP is disabled because RMP protection is not in effect.

3.4 ENABLING/DISABLING LEGACY FEATURES

To only enable and disable the legacy features (Legacy SEV & ES) in a server you only need to follow the first 2 steps for every EPYC Processor generation in the **"Configuring SNP through BIOS" section 3.1**. These features must be enabled for SNP to be enabled; you cannot have SNP enabled without Legacy SEV and ES being enabled as well.

To disable all security features, you may simply disable SMEE as described in **"Disabling SMEE in BIOS" section 2.3**.

CHAPTER 4

OS REQUIREMENTS

For the different SEV hardware features, verify that your OS supports the future as a hypervisor and/or as a guest, show in the following tables

Note: TSME is OS-independent and only needs enablement in the BIOS

4.1 SECURE ENCRYPTED VIRTUALIZATION

The following kernels/OS support Legacy SEV:

OS/KERNEL	HOST	GUEST
LINUX 4.15		☑
LINUX 4.16	☑	☑
RHEL 7.6		☑
RHEL 8	☑	☑
FEDORA 28	☑	☑
SLES 15	☑	☑
UBUNTU 18.04		☑
UBUNTU 18.10	☑	☑
ORACLE UEK 5	☑	☑

Table 4-1: Legacy SEV Support

4.2 ENCRYPTED STATE

The following kernel/OS support ES:

OS/KERNEL	HOST	GUEST
LINUX 5.11	☑	☑
FEDORA 34	☑	☑
UBUNTU 22.04	☑	☑
SLES 15 SP3	☑	☑
RHEL 8.4	☑	☑

Table 4-2: ES Support

4.3 SECURE NESTED PAGING

The following kernels/OS support SNP:

OS/KERNEL	HOST	GUEST
LINUX 5.10		☑
LINUX 5.11	☑	☑
FEDORA 34	☑	☑
UBUNTU 22.04	☑	☑
SLES 15 SP3	☑	☑
RHEL 8.4	☑	☑
LINUX 5.19		☑
LINUX 6.11	☑	☑
RHEL 9.1		☑
RHEL 9.5	☑	☑
UBUNTU 22.04		☑
UBUNTU 25.04	☑	☑
SLES 15 SP4		☑
SLES 15 SP7	☑	☑
FEDORA 37		☑
FEDORA 41	☑	☑

Table 4-3: SNP Support

4.4 OS CERTIFICATION

Although the charts above show the minimum software versions required for each SEV generation, some SEV-related features are enabled later at the kernel or operating system level and may not be included with the initial minimum support.
Because of this, it can be difficult to track which features are available across different OS versions. To simplify this, AMD provides a certification suite that validates operating systems with the appropriate kernel and package support for SEV features introduced in later releases.

For details on certification levels and to view currently certified operating systems, visit the SEV Certification Suite:
https://github.com/AMDEPYC/sev-certify

CHAPTER 5

OS ENABLEMENT

5.1 CHECKING SECURITY FEATURE ENABLEMENT

Execute the following command to find all SEV related kernel prompts:

```
$ sudo dmesg | grep SEV
```

For Legacy SEV you should see:

```
SEV supported
```

or

```
kvm_amd: SEV enabled (ASIDs 123 - 1006)
```

Note: Both are valid, depending on the kernel version

For ES you should see:

```
kvm_amd: SEV-ES enabled (ASIDs 1 - 122)
```

For SNP you should see:

```
kvm_amd: SEV-SNP enabled (ASIDs 1 - 122)
```

Along with some additional RMP details:

```
SEV-SNP: RMP table physical range [0x0000018276e00000 - 0x00000183faefffff]
-SNP: Reserving start/end of RMP table on a 2MB boundary [0x00000183fae00000]
```

An example when all features are enabled at the same time would look like:

```
SEV-SNP: RMP table physical range [0x0000018276e00000 - 0x00000183faefffff]
SEV-SNP: Reserving start/end of RMP table on a 2MB boundary [0x00000183fae00000]
ccp 0000:55:00.5: SEV API:1.55 build:37
ccp 0000:55:00.5: SEV-SNP API:1.55 build:37
kvm_amd: SEV enabled (ASIDs 123 - 1006)
kvm_amd: SEV-ES enabled (ASIDs 1 - 122)
kvm_amd: SEV-SNP enabled (ASIDs 1 - 122)
```

In the above example:
The numbers of ASIDS after the SEV enabled notice are ASIDs for Legacy SEV only. The SEV-ES and SEV-SNP ASIDs are shared ASIDs for ES and SNP.

5.2 ENABLING SECURITY FEATURES

Look at chapter 4 to find the minimum requirements at the OS level to enable the security feature you want. We recommend always using the latest available kernel. The following instructions assume install kernel (6.11 or newer) that supports SNP.

To enable SNP in your OS:

1. Follow the procedure described in **"CONFIGURING SNP"** to enable all security features.
2. Verify that the current firmware installed is the newest available (1.55 at the time of publication). If needed, update the firmware as described in **"UPDATING SEV FIRMWARE"**.
3. If needed install a 6.11 (or newer) host kernel on the system, the following kernel options should be set:

   ```
   --enable AMD_MEM_ENCRYPT
   --enable KVM_AMD_SEV
   --disable IOMMU_DEFAULT_PASSTHROUGH
   ```

   If you wish to use the same kernel for your guest VM, make sure the following option is set:

   ```
   --module SEV_GUEST
   ```
4. Verify your security features are enabled using the command shown in the previous section.

*Note: There are certain features that were not included in the 6.11 kernel. For example, the ability to perform extended attestation on SNP enabled guests. To get developmental features in a system follow the instructions in **https://github.com/AMDESE/AMDSEV/tree/snp-latest** to build the development kernel, OVMF and QEMU.*

5.3 ADDITIONAL RESOURCES

Please see the following resources for additional information:

- Kernel.org: **https://www.kernel.org/doc/html/v5.8/virt/kvm/amd-memory-encryption.html**
- RHEL: **https://docs.redhat.com/en/documentation/red_hat_openstack_platform/16.1/html/configuring_the_compute_service_for_instance_creation/assembly_configuring-amd-sev-compute-nodes-to-provide-memory-encryption-for-instances_amd-sev**
- Oracle: **https://blogs.oracle.com/linux/post/using-amd-secure-memory-encryption-with-oracle-linux**
- SUSE: **https://documentation.suse.com/sles/15-SP7/html/SLES-all/cha-vm-security.html**

CHAPTER 6

UPDATING SEV FIRMWARE

You should always use the latest SEV firmware supported by your BIOS to have the latest features and security protection. To update SEV firmware:

1. Update your system **BIOS**.
2. Manually add a new firmware to your OS and then execute the **DownloadFirmware (DLFW)** command. See **"DOWNLOADFIRMWARE (DLFW)" Section 6.1**.
3. Execute the **DownloadFirmwareEX (DLFW_EX)** command. See **"DOWNLOADFIRMWAREEX (DLFW_EX)" Section 6.2**.

The **`DownloadFirmware`** and **`DownloadFirmwareEX`** commands replace the local copy of SEV in DRAM with the new image. Calling the next SEV command loads that new copy into SRAM and runs it. The BIOS copy remains in SpiRom; rebooting the system will run the older BIOS image until you execute these commands again to update to the latest version.

6.1 DOWNLOADFIRMWARE (DLFW)

The DownloadFirmware command allows system administrators to the version of SEV running on the platform without having to reboot the platform or update the BIOS, provided that:

- All SEV/SNP guests are shut down.
- The SEV/SNP platform state is UNINIT.

The Linux CCP driver will automatically check for a new SEV image when initialized. If it finds a new image, then it will execute the **`DownloadFirmware`** command.

1. Download the latest firmware version from **https://developer.amd.com/sev/** (see Figure 6-5).
2. Copy the appropriate firmware file to **`/lib/firmware/amd/`** and then name the file to **`amd_sev_fam[family:02X]h_model[model:02X]h.sbin`** (see Figure 6-1). If needed, you may create an **`/amd`** folder as shown in Figure 6-2, then paste the **`.sbin`** into this folder (see Figure 6-3), and then rename the firmware as shown in Figure 6-4. Only files with the right naming convention in the folder will get applied at boot, so if your update is not being applied, make sure that the naming convention is correct.
3. Reboot the system or reload the ccp to invoke **`DOWNLOADFIRMWARE`** and then you should see your update being applied. To reload the ccp you can run the following commands:

```
sudo rmmod kvm_amd
sudo rmmod ccp
sudo modprobe ccp
sudo modprobe kvm_amd
```

Figure 6-1: Firmware download example

Figure 6-2: lib/firmware folder, with the /amd subfolder created.

Figure 6-3: Pasted .sbin before renaming

Figure 6-4: Pasted .sbin after renaming

Links & Downloads

Link	Description
https://github.com/AMDESE/AMDSEV	Linux open source code under development
Confidential Containers	Confidential Containers (CoCO) Project
Using AMD Secure Memory Encryption with Oracle Linux	Oracle UEK support for SME and SEV.
SUSE: AMD Secure Encrypted Virtualization (AMD-SEV) Guide	Provides a basic understanding of how SEV works, how to enable and configure it, and some of the limitations and restrictions that its use causes as compared to non-encrypted virtualization.
ask_ark_naples.cert	ASK/ARK certificates for EPYC 7xx1 (Naples)
ask_ark_rome.cert	ASK/ARK certificates for EPYC 7xx2 (Rome)
ask_ark_milan.cert	ASK/ARK certificates for EPYC 7xx3 (Milan)
ask_ark_genoa.cert	ASK/ARK certificates for EPYC 9xx4 (Genoa)
ask_ark_prod_turin.cert	ASK/ARK certificates for EPYC 9xx5 (Turin)
amd_sev_fam17h_model01h_0.17.48.zip	SEV Firmware | SEV firmware 0.17.48 [hex 00.11.30] for EPYC 7xx1 (Naples)
amd_sev_fam17h_model3xh_0.24.20.zip	SEV Firmware | SEV firmware 0.24.20 [hex 00.18.14] for EPYC 7xx2 (Rome)
amd_sev_fam19h_model0xh_1.55.36.zip	SEV Firmware | SEV firmware 1.55.36 [hex 1.37.24] for EPYC 7xx3 (Milan)
amd_sev_fam19h_model1xh_1.55.49.zip	SEV Firmware | SEV firmware 1.55.49 [hex 1.37.31] for EPYC 9xx4 (Genoa)
amd_sev_fam1ah_model0xh_1.55.65.zip	SEV Firmware | SEV firmware 1.55.65 [hex 1.37.41] for EPYC 9xx5 (Turin)
CEK certificate web page	Interactive tool for obtaining CEK certificate. Also available as https://kdsintf.amd.com/cek/id/<GetIDValue>
https://github.com/AMDESE/sev-tool	AMD SEV Tool for managing SEV platform certificates

Figure 6-5: SEV firmware download links

6.2 DOWNLOADFIRMWAREEX (DLFW_EX)

The DownloadFirmwareEX command only applies to 3rd Gen AMD EPYC™ Processors and later. This command allows system administrators to upgrade the version of SEV running on the platform without having to reboot the platform or update the BIOS. SNP guests may remain running during the update, but all SEV guests must be shut down. The exception is that you may be required to shut down the guests or uninitialize the SNP platform in certain cases, such as if a security bug was found in a previous version and the running guests cannot be upgraded securely.

The minimum version requirements for this command are:

- PSP Bootloader: 00.13.00.60 (Milan PI 1004 BIOS).
- SEV uapp version: 1.2B.2B (around Milan PI 1007 BIOS).

If you are running a SEV version that does not support DLFW_EX, then you will have to first shut down your guests and then run the DLFW workflow (**"DownloadFirmware" section 6.1**) to upgrade to the SEV version that supports **`DownloadFirmwareEX`** and then use **`DownloadFirmwareEX`** going forward.

DLFW_EX is currently being upstreamed into the kernel for support, expecting to land on kernel 6.20. For more information on tooling that can be used to use this command to update servers on demand look at **"ATTESTATION"**.

CHAPTER 7

LAUNCHING ENCRYPTED VMs

7.1 LAUNCHING A VM WITH SNP ENCRYPTION

To launch a VM with SNP enabled, enable SNP in the system as described in **"Configuring SNP"**, and then verify that you have the following minimum versions:

PROJECT	VERSION
LIBVIRT	10.5.0
QEMU	9.1
OVMF	Commit newer than (EDK2-STABLE 2025-02)

Table 7-1: Minimum project versions to support SEV-SNP-encrypted VMs

Once the right packaging and OS versions are installed, launch an SNP guest using QEMU.
The following is an example launch command:

```
qemu-system-x86_64 \
-enable-kvm \
-cpu EPYC-v4,phys-bits=52 \
-machine q35 \
-no-reboot \
-vga none \
-vnc :0\
-bios /path/to/ovmf/SNPGUEST.amdsev.fd
-drive file=SNPGUEST.qcow2,if=none,id=disk0,format=qcow2 \
-device virtio-scsi-pci,id=scsi0,disable-legacy=on,iommu_platform=on,romfile= \
-object memory-backend-memfd,id=ram1,size=2048M,share=true,prealloc=false
-machine memory-backend=ram1
-device scsi-hd,drive=disk0 \
-machine memory-encryption=sev0,vmport=off \
-object sev-snp-guest,id=sev0,cbitpos=51,reduced-phys-bits=1
```

The following are important points to notice around SNP VM launches:

- SNP needs a specific AMD version of OVMF to provide SNP capabilities. This OVMF is a new format that combines CODE and VARS in a single file. This file will contain an amdsev tag in the file name: **/path/to/ovmf/OVMF.amdsev.fd**
- This file will usually be found in either **/usr/share/ovmf/OVMF.amdsev.fd** or **/usr/share/edk2/ovmf/OVMF.amdsev.fd** in most distros.
- The user should copy and rename the file to match its guest image name to avoid reusing the file. In the above example the OVMF was renamed to: **SNPGUEST.amdsev.fd**
- Need to use the -bios flag and point to the AMD unsplit OVMF: **-bios /path/to/ovmf/SNPGUEST.amdsev.fd**
- VGA must be set to none. For devices that normally include an option ROM, you must explicitly disable it by setting **romfile=** with no value:

  ```
  -vga none
  -device virtio-scsi-pci,id=scsi0,disable-legacy=on,iommu_platform=on,romfile=
  ```

- Make sure memfd is being used:

  ```
  -object memory-backend-memfd,id=ram1,size=2048M,share=true,prealloc=false”
  -machine memory-backend=ram1
  ```

- And to enable SNP make sure the sev-snp-guest object is being used:

  ```
  -machine memory-encryption=sev0,vmport=off \
  -object sev-snp-guest,id=sev0,cbitpos=51,reduced-phys-bits=1
  ```

Since SNP is only supported from processor series 7003 and newer, the c-bit (cbitpos) will always be 51.

The **sev-snp-guest** object allows users to specify their desired SNP policy. The policy is defined as a hexadecimal value using the policy parameter. The available SNP policy bit definitions are as follows:

BIT(S)	NAME	DESCRIPTION
63:26	Reserved, should be 0	
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. 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; should be 0.	
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.

Table 7-2: SNP guest policy

For more information on the other parameters that can be passed to the sev-snp-guest object please visit: **https://www.qemu.org/docs/master/system/i386/amd-memory-encryption.html**

Execute the following command on the guest to confirm that SNP is enabled:

```
dmesg | grep SEV
Memory Encryption Features active: AMD SEV SEV-ES SEV-SNP
```

Note: Launching a VM with SNP encryption includes Legacy SEV and ES encryption. That is why this is the recommended way of launching VMs.

Note: LibVirt added support for SNP in the 10.5.0 release. For complete information in launching and managing SNP in LibVirt please consult the SUSE SNP guide: ***https://documentation.suse.com/sles/15-SP7/html/SLES-all/cha-vm-security.html***

7.2 LAUNCHING A VM WITH LEGACY FEATURES

To launch a VM with Legacy SEV encryption only, enable SEV in the system as described in **"Enabling/Disabling Legacy Features" section 3.4**, and then verify that you have the following minimum versions:

PROJECT	VERSION
LIBVIRT	4.5
QEMU	2.12
OVMF	Commit newer than (75b7aa9528bd 2018-07-06)

Table 7-3: Minimum project versions to support SEV-encrypted VMs

Once the right packaging and OS versions are installed, launch an Legacy SEV guest using QEMU. The following is an example launch command:

```
$ qemu-system-x86_64 \
-enable-kvm \
-cpu EPYC-v4,phys-bits=52(or host-phys-bits=true) \
-machine q35 \
-no-reboot \
-vga std \
-vnc :0
-drive file=distro.iso=cdrom -boot d \
-drive if=pflash,format=raw,unit=0,file=/usr/share/OVMF/OVMF_CODE.fd,readonly=on \
-drive if=pflash,format=raw,unit=1,file=encryptedImage.fd \
-drive file=encryptedImage.qcow2,if=none,id=disk0,format=qcow2
-object sev-guest,id=sev0,policy=0x3,cbitpos=47,reduced-phys-bits=1
-device virtio-scsi-pci,id=scsi0,disable-legacy=on,iommu_platform=on
-device scsi-hd,drive=disk0
-machine memory-encryption=sev0,vmport=off
```

Note: QEMU defaults to 40 physical address bits when using architected CPU models like "EPYC-v4" above. This can be somewhat restrictive for large guests with >1TB memory, or guests with multiple GPUs with large amounts of memory passed through. Because of this, it is recommended to set "phys-bits=52" (or higher if needed) as is done above. Also note that because 1 guest physical address bit will be used to differentiate between shared vs. private accesses in the guest's page tables, this may reduce the amount of physical memory the guest is able to access accordingly, so this should also be accounted for when setting "phys-bits".

*Note: You may need to edit these commands to suit your needs and use cases. For example, different distros may have different QEMU launch commands. Please see the guides listed in **"Additional Resources" Section 5.3** for more information.*

Some important aspects to note about Legacy SEV launch:

- Legacy SEV can use the standard OVMF, it does not need to point to the amdsev version of OVMF as SNP does.
- Users can reuse the **`OVMF_CODE.fd`** file, but it is recommended to copy and rename the **`OVMF_VARS.fd`** file, in the example above it was changed to **`encryptedImage.fd`**.
- The cbitpos parameter in the sev-guest object line, can change depending on the processor generation AMD EPYC™ 7002 Processors have a c-bit value of 47. Newer AMD EPYC™ Processors have a c-bit value of 51.
- To launch guests with legacy encryption only (just Legacy SEV or ES) instead of using the **`sev-snp-guest`** object, the user would use the **`sev-guest`** object. This object comes with its own set of parameters and policy.

On the guest, execute the following command to verify that SEV is enabled:

```
dmesg | grep SEV
Memory Encryption Features active: AMD SEV
```

The **sev-guest** object allows users to specify their desired Legacy SEV policy. The policy is defined as a hexadecimal value using the **policy parameter**. The policy for Legacy SEV is different from the policy used in SNP. If you are using the **sev-guest** object, you'll be using the Legacy SEV policy which is defined as follows:

OFFSET	BIT(S)	NAME	DESCRIPTION
000H	0	NODBG	Debugging of the guest is disallowed when set.
	1	NOKS	Sharing keys with other guests is disallowed when set.
	2	ES	ES is required when set.
	3	NOSEND	Sending the guest to another platform is disallowed when set.
	4	DOMAIN	The guest must not be transmitted to another platform that is not in the domain when set.
	5	SEV	The guest must not be transmitted to another platform that is not SEV capable when set.
	15:6	Reserved; should be 0.	
002H	7:0	API_MAJOR	The guest must not be transmitted to another platform with a lower firmware version.
003H	7:0	API_MINOR	

Table 7-3: SEV policy bits

To launch a VM with only Legacy SEV & ES encryption you would need to make sure you have the following minimum versions:

PROJECT	VERSION
LIBVIRT	4.5
QEMU	6.0
OVMF	Commit newer than (EDK2-STABLE 2020-21-02)

Table 7-4: Minimum project versions to support ES-encrypted VMs

Once the right versions are installed, you can use the same QEMU command line as Legacy SEV, and then just make sure ES is enabled by passing the right policy bits. Looking at table 7-3, bit 2 must be enabled to boot the VM with ES encryption. A valid ES configuration would look like:

```
-object sev-guest,id=sev0,policy=0x5,cbitpos=47,reduced-phys-bits=1
```

Once the guest boots, you can execute the same dmesg command to confirm enablement:

```
dmesg | grep SEV
Memory Encryption Features active: AMD SEV SEV-ES
```

*Note: Please see **https://libvirt.org/kbase/launch_security_sev.html** for instructions on SEV VMs with LibVirt.*

CHAPTER 8

ATTESTATION

After launching a confidential guest, users will also want to attest this guest. Attestation allows guest owners to prove that their confidential guest is set-up correctly and they have the protection benefits that SEV promises. To perform attestation users would have to build applications around the attestation functions that the SEV and SEV-SNP IOCTLs provide. The VirTEE open-source community project provides pre-built tools for users to perform attestation in their confidential virtual machines and provides an API library for users to build their own applications around SEV. VirTEE simplifies attestation for users unfamiliar with kernel development and allows everyone to make sure their virtual machines are protected. These are the tools and libraries that VirTEE provides for SEV users:

- **SEV Library** - A Rust crate that provides an implementation of the SEV APIs and SNP ABIs, simplifying the communication with the Linux kernel for developers.
- **SNPHOST** – Client tool to manage and set-up SNP enabled hosts.
- **SNPGUEST** – Client tool to manage, attest and calculate launch measurements for SNP guests.
- **SEVCTL** – Client tool to manage Legacy SEV guests and hosts.

CHAPTER 9

COCONUT-SVSM

SNP's Virtual Machine Permission Level (VMPL) feature permits the inclusion of components in the guest that can run with a higher privilege than the guest operating system. This offers an environment for secure, privileged code modules to run without interference from the bulk of the guest. These modules are not part of the guest OS and may be designed to be configurable, offering compatibility with a wide variety of guest configurations. This is what a Secure VM Service Module (SVSM) aims to do. The SVSM is an environment that can host privileged modules within the guest.

COCONUT-SVSM is a project that aims to enable the SVSM capabilities on confidential VM's taking advantage of the VMPL features. COCONUT-SVSM is still being developed, but users can already use it to launch SVSM capable of VMs, with secure service modules such as Virtual TPMs (vTPM).

9.1 COCONUT-SVSM HARDWARE REQUIREMENTS

AMD EPYC™ Series 7003 Processor or newer. (SNP capable Processors)

9.2 COCONUT-SVSM OS REQUIREMENTS

The following kernels support COCONUT-SVSM:

OS/KERNEL	HOST	GUEST
IN DEVELOPMENT	☑	☑

Table 10-1: Coconut-SVSM requirements

9.3 COCONUT-SVSM HARDWARE ENABLEMENT

To enable COCONUT-SVSM in the hardware, the user just must enable **SNP** in the **BIOS** as described in **"CONFIGURING SNP"** .

9.4 COCONUT-SVSM OS ENABLEMENT

Since the SVSM capabilities are still in development, there are several different components that need to be built separately.

- Verify that the current firmware installed is the newest available (1.55 at the time of publication) for SNP-compatible AMD EPYC™ 7003,9004, and 9005 Series Processor. If needed, update the firmware as described in **"UPDATING SEV FIRMWARE"**.
- Follow the instructions in the SVSM quick start guide which will walk the user through building:
  - Linux host kernel with SVSM support
  - Linux guest kernel with SVSM support
  - EDK2 with SVSM support
  - A modified QEMU which supports launching guests configures using IGVM

The quick start can be found here:
https://github.com/coconut-svsm/svsm/blob/main/Documentation/docs/installation/INSTALL.md

9.5 LAUNCHING A COCONUT-SVSM VM

To launch an SVSM enabled guest a similar QEMU command line will be used as that to launch a regular SNP guest (look at **"LAUNCHING A VM WITH SNP ENCRYPTION" Section 7.1**). The only difference is that instead of providing an OVMF file with the **`-bios flag`**, an IGVM file needs to be used instead. (generated in the last step of the previous section). The IGVM file contains the firmware that would usually be in OVMF. The path to the IGVM file is provided in the **`sev-snp-guest`** object. The **`-bios`** flag is no longer used:

Example:

```
-object sev-snp-guest,id=sev0,cbitpos=51,reduced-phys-bits=1,igvm-file=/path/to/
coconut-qemu.igvm
```

When launching the SVSM enabled guest, the following message should appear on the terminal during boot:

```
[Stage2] COCONUT Secure Virtual Machine Service Module (SVSM) Stage 2 Loader
```

At this point an SVSM-enabled guest should be launched.

CHAPTER 10

TRUSTED I/O

TIO (Trusted I/O) extends SNP capabilities to provide hardware based I/O encryption and integrity protection for devices assigned to virtual machines. This chapter provides detailed instructions for enabling and configuring TIO on AMD EPYC™ Processors.

10.1 TIO HARDWARE REQUIREMENTS

TIO is supported by AMD EPYC™ 9005 Processors and newer.

External devices need TEE Device Interface Security Protocol (TDISP) available to be compatible with TIO.

10.2 TIO OS REQUIREMENTS

TIO is officially supported on Turin systems with PI 1006 firmware or later. Install the latest firmware on your Turin platform to ensure TIO is available on your system.

TIO is not yet supported in the upstream Linux kernel. To enable it, you must build and install a custom kernel on both the host and the guest. The latest development kernel is available here:

TIO Kernel: **https://github.com/AMDESE/linux-kvm/tree/tsm**

A custom version of QEMU is also needed to launch a guest with TIO capabilities. This customized QEMU build can be found here:

TIO QEMU: **https://github.com/AMDESE/qemu/tree/tsm**

Build scripts are available to help with building and managing custom kernels as well as generating custom QEMU images. These scripts provide a simplified way to build the necessary TIO components and are available here:

TIO build scripts: **https://github.com/AMDESE/AMDSEV/tree/tsm**

External TDISP-capable devices may require vendor-specific drivers to interoperate with TIO. Contact your device OEM for driver availability and support

10.3 TIO HARDWARE ENABLEMENT

To enable TIO in your system follow these set-up instructions:

1. TIO builds upon SNP functionality. Before enabling TIO-specific settings, first enable SNP as described in **Chapter 3 "Configuring SNP"**.
2. Once SNP is enabled, in **BIOS**, return to **Advanced** > **AMD CBS** > **CPU Common Options**, and then set the **RMP Coverage for 64bit MMIO Ranges** parameter to **Enabled**.

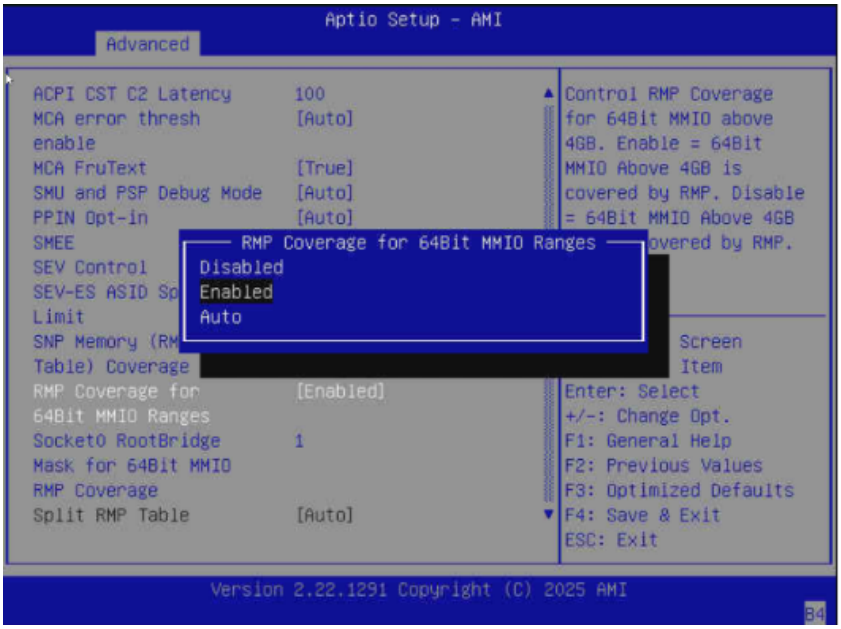

Figure 10-1: Enable RMP Coverage for 64Bit MMIO Ranges

3. Set the **Socket0 RootBridge Mask for 64Bit MMIO RMP Coverage** to the slot where the external device will be attached. It is recommended to set this value to 0xFF, as it covers all available slots and has no negative impact.

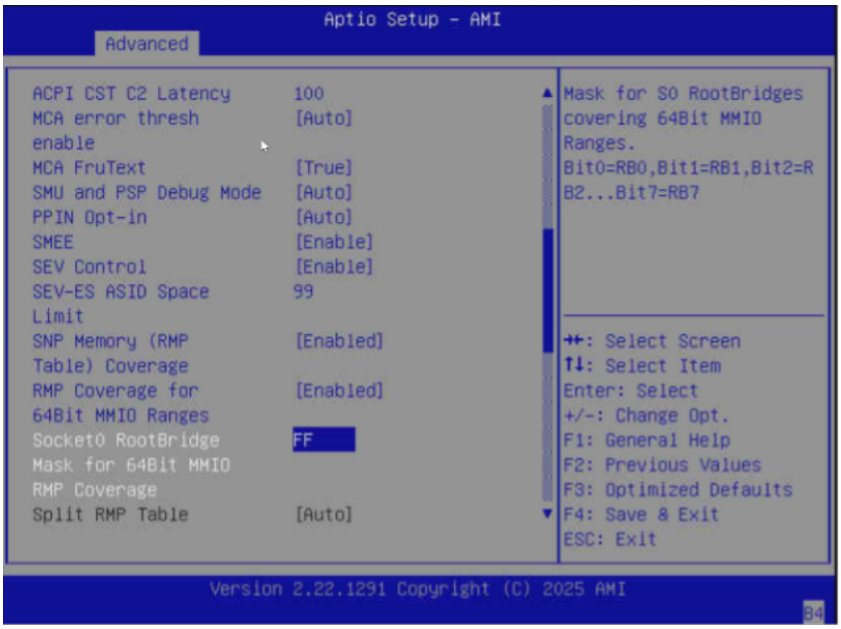

Figure 10-2: Set the Socket0 RootBridge Mask for 64Bit MMIO RMP Coverage to FF

4. Set **Segmented RMP** Table to **Enabled**

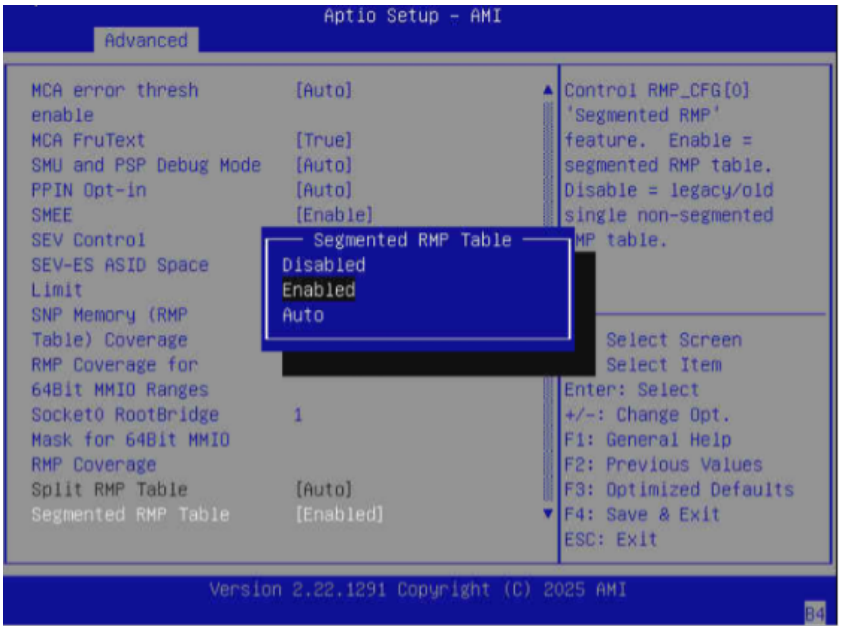

Figure 10-3: Set Segmented RMP Table to Enabled

5. Go back to the BIOS main page then to **Advanced** > **AMD CBS** > **NBIO Common Options** > **IOMMU/Security** and set **SEV-TIO** to **Enabled**.

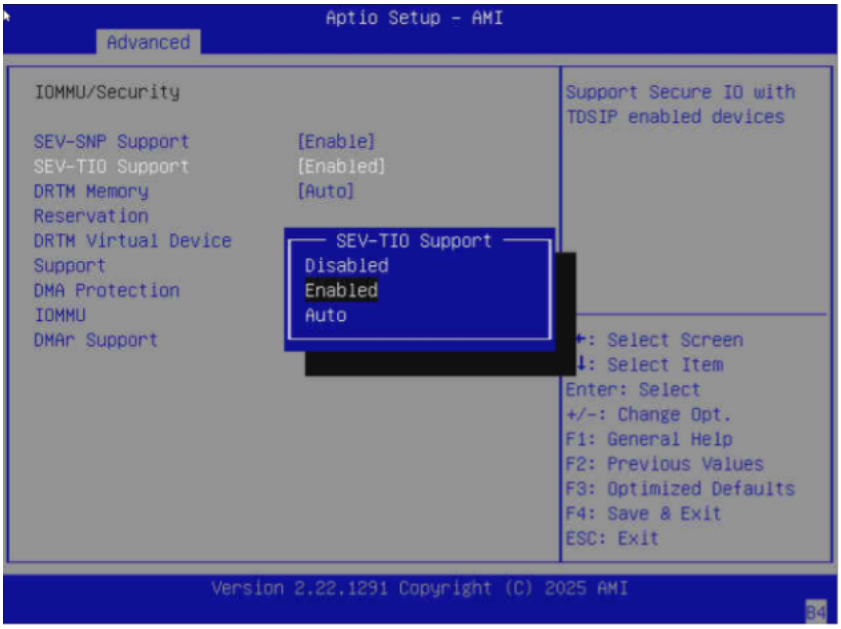

Figure 10-4: Set SEV-TIO Support to Enabled

Note: For AMD EPYC™ Turin processors, BIOS layouts may vary depending on the processor model (e.g., 9655 vs 9755). The options to be enabled remain the same, but their location in the BIOS menu structure may differ.

10.4 TIO OS ENABLEMENT

10.4.1 SET-UP TIO IN OS

Once TIO is enabled in hardware the following steps will show how to enable TIO at the OS level:

1. Build and Install TIO Kernel
   Get the kernel from the **TIO kernel repository** and follow the building instructions in the README to install the kernel in the host system. Build scripts can also be found in the build repository to build and install the kernel instead. Make sure to reboot after installing the new kernel.

2. If not loaded at boot, load the required kernel modules using the following commands:
   ```
   sudo modprobe ccp
   sudo modprobe tpm
   sudo modprobe kvm_amd
   ```
   To see if a kernel module is already loaded you can run the following command:
   ```
   lsmod | grep <module_name>
   ```

3. Once the TIO kernel is installed and the modules are loaded, run the following command to confirm TIO was enabled in the system:
   ```
   sudo dmesg | grep -i SEV
   ```
   Telemetry similar to this one should appear on the console:
   ```
   AMD-Vi: Extended features (0xa5bf732fa2295afe, 0x53f): PPR X2APIC NX GT [5] IA
   GA PC GA_vAPIC SNP SEV-TIO
   SEV-TIO support is present
   SEV-TIO status: EN=1 INIT_DONE=1 rq=4096..4096 rs=4096..4096 scr=16384..16384
   out=16384..16384 dev=65536 tdi=4096 algos=0
   ```

10.4.2 SET-UP EXTERNAL DEVICE

The following steps will show how to set up an external device to use with TIO. For ease of use these are variables that will be referenced in the following steps:

- **%TDISP-DEVICE-NAME%**: Name of the device we want to securely attach using TIO
- **%BUS:DEVICE:FUNCTION%**: The device's bus device and function. This can change on each system depending various factors such as what slit it is plugged into. An example output would be: 0000:e1:00.0
- **%TDISP-DEVICE-DRIVER%**: To do most of the enablement commands the driver for the device has to be loaded. The name of this driver will be different depending on the device, so please check with this device's OEM to get more information on this driver.

1. Load your device driver to the kernel modules:
   ```
   sudo modprobe %TDISP-DEVICE-DRIVER%
   ```

2. Identify the TDISP device and the bus device function.
   ```
   lspci -D | grep -i %TDISP-DEVICE-NAME%
   ```
   Expected output:
   ```
   %BUS:DEVICE:FUNCTION% %TDISP-DEVICE-NAME%
   ```

3. Enable SR-IOV Virtual Functions
   ```
   sudo bash -c 'echo 1 > /sys/bus/pci/devices/%BUS:DEVICE:FUNCTION%/sriov_numvfs'
   ```

4. Connect the device to the Trusted Security Manager (TSM):
   ```
   sudo bash -c 'echo tsm0> /sys/bus/pci/devices/%BUS:DEVICE:FUNCTION%/tsm/
   connect'
   ```

5. Verify TSM connection:
   sudo dmesg | grep - %BUS:DEVICE:FUNCTION%
   Expected output showing secure connection:
   %BUS:DEVICE:FUNCTION%: 2=SECURE <-> %BUS:DEVICE:FUNCTION%: 2=SECURE ret=0/0

6. Verify TSM interface:
   sudo cat /sys/bus/pci/devices/%BUS:DEVICE:FUNCTION%/tsm/connect
   Expected output:
   tsm0

7. Configure hugepages:
   sudo bash -c 'echo 3000 > /sys/kernel/mm/hugepages-2048kb/nrhugepages'
   This sets hugepage limit to 3000 pages of 2MB each

8. Optional: The following 2 commands can be used to dump the status of the device. This provides information about the TSM status and the PCIe IDE. These are not required for set-up but running them successfully are another indicator that the device is set up correctly. The commands use tools that are in the development kernel repository, so they can be found wherever that code was stored:
   ~/path-to-kernel-repo/linux-kvm/tools/crypto/tsm/tsmsysfs.py --dev /sys/bus/pci/devices/%BUS::DEVICE:FUNCTION%/tsm/dsm_status
   ~/path-to-kernel-repo/linux-kvm/tools/crypto/tsm/ide.sh %BUS:DEVICE:FUNCTION%

9. Load the VFIO module to the kernel
   sudo modprobe vfio_pci

10. Set-up VFIO for the device running the following commands:
    echo "%BUS:DEVICE:FUNCTION %" > /sys/bus/pci/devices/%BUS:DEVICE:FUNCTION%/driver/unbind
    echo "vfio-pci" > /sys/bus/pci/devices/%BUS:DEVICE:FUNCTION%/driver_override
    echo "%BUS:DEVICE:FUNCTION%" > /sys/bus/pci/drivers/vfio-pci/bind
    echo "" > /sys/bus/pci/devices/%BUS:DEVICE:FUNCTION%/driver_override

11. Optional: change owner of the following files to the user to avoid permission issues.
    chown user:user /dev/vfio/* /dev/vfio/devices/vfio* /dev/iommu
    At this point the device and the operating system should be set up and ready to launch a VM with TIO enabled.

10.5 LAUNCHING A TIO VM

To launch a TIO-enabled guest, a QEMU command line similar to that used for a standard SNP guest is used (look at **"LAUNCHING A VM WITH SNP ENCRYPTION" Section 7.1**). The main differences are that the protected device will be passed through using VFIO and using iommufd for device management.

1. Build a QEMU compatible with TIO. To build a custom QEMU please visit:
   https://github.com/AMDESE/qemu/tree/tsm
   Specific instructions for the latest development QEMU version should be found in the repository.

   The build scripts in the build repository can also be used to build the custom QEMU, and even use a launch script that simplifies launching TIO guests.

2. There is currently no mainstream guest kernel support for TIO, so after the custom QEMU build is complete, the guest image must be launched without encryption enabled. Once the guest is running, the TIO kernel built in **section 10.4.1** must be installed into the guest image.

3. Once the custom QEMU is built and the TIO kernel is installed in the guest image, launch a VM with TIO enabled and devices attached using a command like the following:

```
qemu-system-x86_64 \
 -enable-kvm \
 -cpu EPYC-v4,phys-bits=52 \
 -machine q35,confidential-guest-support=sev0,memory-backend=ram1\
 -no-reboot \
 -vga none \
 -bios /path/to/ovmf/TIOGUEST.amdsev.fd \
 -drive file=TIOGUEST.qcow2,if=none,id=disk0,format=qcow2 \
 -device virtio-scsi-pci,id=scsi0,disable-legacy=on,iommu_platform=on,romfile= \
 -object memory-backend-memfd,id=ram1,size=2048M,share=true,prealloc=false"
 -device scsi-hd,drive=disk0 \
 -machine memory-encryption=sev0,vmport=off \
 -object sev-snp-guest,id=sev0,cbitpos=51,reduced-phys-bits=1,convert-in-
place=true,gmem-allocator=hugetlb,gmem-page-size=2097152 \
 -object iommufd,id=i0 \
 -device pcie-root-port,id=r0,slot=0,addr=6 \
 -device vfio-pci,id=vfio,host=%BUS:DEVICE:FUNCTION%,bus=r0,id=v0,iommufd=i0
```

   The following commands being important for TIO:

   - In the **-object sev-snp-guest** setting, notice that the convert-in-place setting, the **gmem-allocator-hugetlb** and the **gmem-page-size** settings are added. This enables huge pages and in place conversion needed for TIO.
   - The **-object iommufd,id=0** creates an IOMMU file descriptor for managing IOMMU.
   - Then the **-device pcie-root-port** adds PCIe root port to the guest's PCI topology.
   - Lastly **-device vfio-pci** passes through the device to the guest using VFIO, using the created root port and the iommufd. Change the **%BUS:DEVICE:FUNCTION%** to that of the encrypted device.

4. Once inside the guest, load the device driver

```
sudo modprobe %TDISP-DEVICE-DRIVER%
```

5. Finally in the guest you can run the following command to see if the device has been attached successfully and TIO is enabled:

```
ls /sys/class/tsm/tsm* >/dev/null 2>&1 && echo "SEV-TIO enabled. TSM Device
Exists" || echo "SEV-TIO disabled. TSM Device Not Found"
```

   Expected output:

```
SEV-TIO enabled. TSM Device Exists
```

   This confirms that the guest can communicate with TIO-protected devices through the VFIO interface!

CHAPTER 11

CONFIDENTIAL CONTAINERS

Users can now launch Legacy SEV, ES and SNP encrypted containers using the open-source project Confidential Containers (CoCo). Please visit **https://github.com/confidential-containers/confidential-containers/blob/main/quickstart.md** for information and instructions on how to set-up confidential containers.

FREQUENTLY ASKED QUESTIONS

What is MinSEVASID?
MinSEVAsid is the minimum ASID that lets you run Legacy SEV guests, everything below that is for ES and SNP guests. For example, if MinSEVAsid is set to 8, then ASIDs 1-7 can only be assigned to ES or SNP guests, and ASIDs 8-(max) can only be used for Legacy SEV guests.

How do I map more than 8TB/16TB of physical address space?
To map to more than 8TB of physical address space (DRAM + PCIe + MMIO, etc), change SEV ASID Count to 253 in the BIOS. AMD EPYC™ 7003 and 7002 AGESA will automatically change this setting to 253 if more than 8TB of physical address space is detected during boot. You must disable SME (which also disables SEV) to map to more than 16TB of physical address space. See **"Disabling SMEE in BIOS" section 2.3**.

How many bits are being used by ASIDs and where is the C-bit on my generation of platform?
See **"Enabling/Disabling SMEE via MSR" section 2.4** to find the number of bits being used by ASIDs. 2nd Gen AMD EPYC™ Processors have the c-bit in bit 47. 3rd, 4th and 5th Gen AMD EPYC™ Processors have the c-bit in bit 51. If in doubt, check the following CPUID function to find the c-bit position:

```
CPUID_Fn8000001F_EBX [AMD Secure Encryption EBX (Core::X86::Cpuid::SecureEncryptionEbx)
```

Where is the SEV documentation?
See: **https://developer.amd.com/sev/**.

Does the APM vol 2 support SEV and SNP?
Yes. See **https://www.amd.com/content/dam/amd/en/documents/processor-tech-docs/programmer-references/24593.pdf**.

What is SEV 2?
SEV 2.0 now refers to the 2nd generation of SEV. This means SEV released with the 2nd Gen AMD EPYC™ Processors. For more information on SEV branding look at **"Security Feature Branding" section 1.1**.

How big will my RMP be for a given amount of memory?
Each RMP entry is 16 bytes, and 256 RMP entries can fit in a 4K page. So, for 512 GB of DRAM:

```
512*1024*1024*1024 bytes / 4096 = 134,217,728 4K pages
134,217,728 4K pages * 16 Bytes per RMP entry = 2,147,483,648 Bytes for all RMP entries
2,147,483,648 Bytes for all RMP entries / (1024*1024) = 2,048 MB = 2GB (approx.)
```

How do I disable SEV?
The easiest way is to disable SMEE in the BIOS (see **"Disabling SMEE in BIOS" section 2.3**). If you want to still use TSME but not SEV, then you can blacklist the ccp kernel driver, so it doesn't load SEV. The last option is to remove the SEV binary from the BIOS, but that is not recommended.

How do I disable SNP?
Don't reserve memory for the RMP in the BIOS, and don't set the SNP_EN MSR from x86. See **"Disabling SNP" section 3.2** for more information.

How do I check if TSME is enabled?
You can check the kernel message to see if TSME is enabled by executing the command:

```
dmesg | grep SME
AMD Memory Encryption Features active: SME
```

Additionally, a SNP guest can request an attestation report, and bit 1 (tsme_en) of the PLATFORM_INFO field contains the tsme_en info. Look at **"ATTESTATION"** for more information.

Which version of SEV firmware did 'x' support get added?
For information on features added in SEV firmware please consult the **SEV API**.
For information on features added in SNP firmware please consult the **SNP ABI**.

PERFORMANCE DATA

You can find an external report discussing SNP performance here:
https://prowessconsulting.com/resources/microsoft-confidential-compute-performance/