Seamless Firmware Servicing (SFS) Specification

Publication #	**58604** Revision: **0.76**
Issue Date:	**February 2026**

Advanced Micro Devices

Specification Agreement

This Specification Agreement ("Agreement") is a legal agreement between Advanced Micro Devices, Inc. ("AMD") and "You" as the recipient of the attached AMD Specification ("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. You agree that You shall not use nor procure others to use the contents of this Agreement for (i) modifying any existing patent or patent application or creating any continuation, continuation in part or other extension of any patent or patent application, nor (ii) analyzing, assessing any patent or patent application (which shall include the creation or modification of any patent claim charts or infringement analyses).

4. 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.

5. 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.

6. 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.

7. 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/*. This Section 7 is applicable solely to Specifications shall not apply to any Specifications that are released publicly.

8. 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.

9. 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
February 2026	0.76	Updated FWID table
January 2026	0.75	Updated uCode flows
January 2026	0.74	Added IB/OOB error messages. Added instruction to round up to 64k for total size of image.
August 2025	0.73	Add LMS token and signing information. Add a note about uCode updates
January 2025	0.72	Updated GET_FW_VERSIONS command, added OOB information and other editorial updates.
June 2024	0.71	Added Section 5.2.2
May 2024	0.70	Initial public release

Chapter 1 Introduction

1.1 Purpose and Audience

This document describes the architecture, assumptions, test methodology, and maintenance of the Seamless Firmware Servicing (SFS) support process. This document facilitates customer personnel involved in developing, testing, and deploying SFS support.

1.2 Applicability

SFS is introduced in Family 1Ah, Models 00h-0Fh and 10h-1Fh. Any references to Out-Of-Band methods and requirements for SFS are introduced in Family 1Ah, Models 50h-5Fh and future SOCs to be listed here.

SFS is intended for large fleet operators where firmware versions are carefully controlled and managed.

1.3 Glossary

AGESA: AMD Generic Encapsulated Software Architecture

APCB: AGESA Platform Configuration Block

SFS: Seamless Firmware Servicing

PSP: Platform Security Processor

ASP: AMD Security Processor, the replacement for the earlier program's PSP

SEV: Secure Encrypted Virtualization

SMM: System Management Module

SNP: Secure Nested Paging

SPDM: Security Protocol and Data Model

TEE: Trusted Execution Environment

IB: In-Band (SFS via the running host SW via a driver)

OOB: Out-Of-Band (SFS via, typically, a BMC not interacting with host SW)

Chapter 2 Background

2.1 Motivation for SFS

AMD created SFS as a secure method to allow non-persistent updates to running firmware and settings without requiring a BIOS reflash and/or system reset. This approach improves system stability by allowing patches to address a few selected “high-benefit/low-risk-to-mitigate” issues on running systems. SFS can improve overall system health without increasing maintenance downtime.

In addition, SFS patching can mitigate some security issues which can reduce the frequency of unplanned maintenance events.

SFS does not patch code that is normally part of customer BIOS and runs on x86, such as SMM, AGESA, or UEFI; nor is it able to update run-once code, such as AGESA boot loader (ABL). The intent of SFS is solely for the purposes stated above.

2.2 Scope of SFS Support

AMD AGESA releases typically contain dozens to hundreds of boot-time and runtime improvements in addition to introducing additional capabilities.

AMD limits the scope of SFS to address priority runtime concerns that affect AMD-provided binary firmware components included in an AGESA release.

SFS does not address anything that runs on the x86 processors. Examples that SFS runs on include ASP firmware, modules, and register settings. Other microprocessors can receive updates.

SFS updates are non-persistent and do not last across reboots. The intent is that the SFS update packages are interim steps between BIOS releases. SFS update packages can extend the time for a customer to validate new BIOS releases in preparation for updating the (typically) flash storage that contains the BIOS.

The design of the SFS update packages apply to a particular PI release with a specified base patch and firmware level; this ensures the update package execution process meets the proper base conditions.

SFS update packages include release notes to indicate what the update patches or resolves.

2.3 SFS Flow

The process of creating an SFS update package follows the flow indicated below (*Figure 1*).

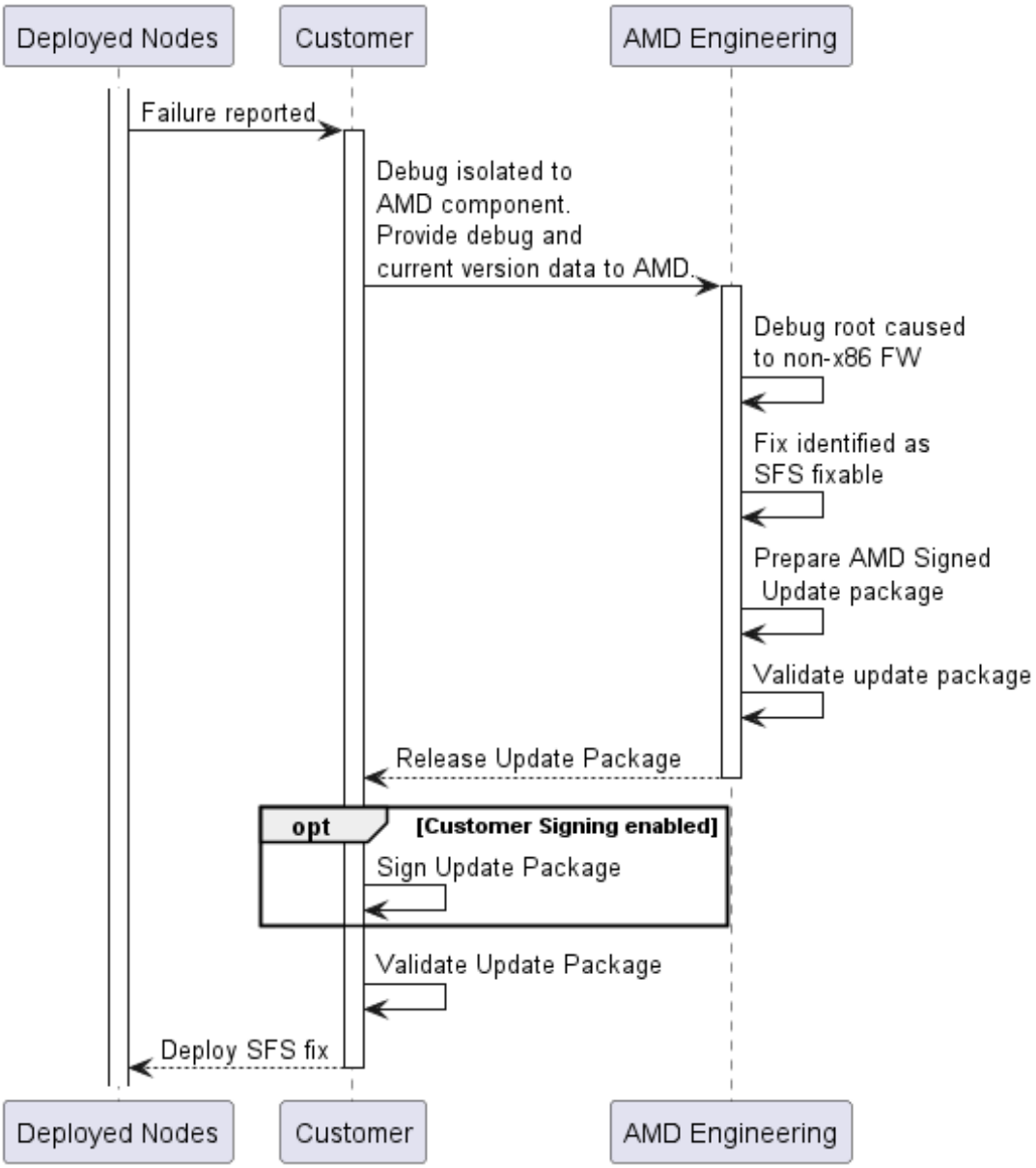

Figure 1. SFS Flow

2.4 Responsibilities

Each customer applying patches is responsible for their decision to proceed with the patches. It is incumbent upon the customer to test and validate any given patch within their own environment. The customer is also responsible for an SFS Deployment Agent (SFSDA) that runs in their environment to initiate and track SFS updates. Updates can be sent to the ASP in one of two methods: In-Band (IB) via the PSP/ASP driver to communicate with AMD's PSP/ASP to execute the patch process or Out-Of-Band (OOB) via the BMC or a node controller that talks to the SOC.

Chapter 3 SFS Operation

SFS is an AMD firmware feature that facilitates its development and distribution of fixes for selected high priority system issues. Datacenter operators can apply SFS patches without rebooting their system.

AMD works to identify the fixes included in each patch then distributes these patches in a ‘Seamless Firmware Servicing Update Package(s).’

Successfully deploying an SFS update package on a system can cause three distinct types of version numbers to change:

a) The SFS Patch Level (SFSPL) identifies the SFS update package currently applied to the system. An unpatched system has a value of zero. To prevent rollback, only SFS packages with a SFSPL greater than the system's current value can deploy (i.e., the SFSPL must monotonically increase). The SFSPL is the overall runtime patch level. The list of firmware versions combined with the SFSPL uniquely identifies the AMD firmware running on a system.

b) Individual firmware components maintain their own firmware and Security Patch Level (SPL) versions. A newer SFS package will always advance the version, but the functionality introduced in one patch may be reverted in a later patch, if needed.

c) SFS may execute run-once code to modify registers and internal SRAM data that exist outside of a versioned 'firmware component.’ To track these changes, SFS uses the concept of the System Patch Level (SYSPL). The SYSPL is the version number for a virtual firmware component that consists of all run-once code executed by SFS since the last reboot. SYSPL increases monotonically. While the effects of run-once code may be reverted, the history of it executing is indelible.

All that information is available to SFS for it to ensure that any to-be-applied-update applies only to the correct level of existing SOC firmware and operational state. That data is also available to the host OS via an SFS command to get the version levels of the firmware (and SPL, SYSPL, and SFSPL).

For security reasons, AMD cryptographically signs the entire SFS package. As an added feature, customers can opt to co-sign the package. Co-signing enables customer approval of any SFS packages deployed within their datacenters. This requires customers to send (during early boot) a public key token to the ASP to apply a patch.

Conceptually, an SFS update package delivers in the form shown below (*Table 1*). For additional details, see *Chapter 5*.

Optional SFS Co-Signing Header (Customer) 256bytes
SFS Package Signing Header (AMD) 256 bytes
SFS Update Package Executable Binary
SFS Package Signature (AMD)
Optional SFS Package Signature (Customer)

Table 1. Patch Signing and General Layout

3.1 SFS Security / Threat Model

The AMD Security Processor (ASP) verifies AMD's signatures on all AMD-owned boot firmware that it retrieves from the ROM. The ASP acts as AMD's root of trust for detection (see *FIPS PUB SP800-193, section 4.1*) for all AMD signed firmware. SFS allows runtime updates to AMD firmware, and SFS extends the ASP's role as the root of trust for detection by requiring the ASP to verify AMD's signature(s) before applying each SFS update package.

AMD's policy has been that the system designer (OEM/ODM) owns the firmware root of trust for update for all the firmware, not AMD (see *FIPS PUB SP800-193 section 4.1* for the difference between root of trust for detection vs update). Unless AMD-provided Platform Secured Boot (PSB) solution is enabled, the system designer (OEM/ODM) is also responsible for root of trust for detection of the system designer owned/signed/provided firmware.

When systems first begin executing OEM/ODM-provided x86 code (i.e., early BIOS code), that code has full read/write access to the ROM's firmware store. This early OEM/ODM trusted code secures the system by enabling restrictions on ROM chip access. Enabling features such as AMD ROM Armor ensures only the OEM/ODM-controlled SMM handler can write to the ROM chip. Only after this lockdown does the OEM's firmware allow untrusted third-party code to execute (such as adapter card ROMs, disk-based boot loaders, and network PXE images). In later stages of execution, OEM/ODM-provided logic then verifies updates before allowing the ROM to apply the updates.

Since SFS offers a means to update running firmware, SFS provides the OEM/ODM the option to co-sign update packages so the OEM/ODM can maintain their role as root of trust for update. During the early boot phase, when the ROM is still in read/write mode, the OEM/ODM may present their public signing key to SFS. Once presented, the ASP will require updates signed by AMD and countersigned by the OEM/ODM's signing key. Maintenance of customer co-signing keys is the responsibility of the customer. AMD does not countersign the customer keys.

***NOTE**: If an attacker manages to subvert the early trusted code, such that they could replace the co-signing key used, the attacker could easily use their escalated privileges to overwrite the*

baseline firmware components in the ROM, which negates any advantage to being able to co-sign updates.

If AMD Platform Secure Boot (PSB) is enabled, the code that installs the SFS co-signature should be part of the code protected by PSB.

3.2 SFS Behavior

Systems boot in an unpatched state, with a baseline firmware level (whatever signed AMD firmware is running). At runtime, customers may choose to apply SFS patches.

3.2.1 Application of SFS Packages

Application of SFS packages must occur at run-time and needs to be reapplied after any system reboot. SFS does not persist over reset. In multi-socket systems, the first sockets' ASP receives the patch communicated to it. Each patch determines if it is on a single or multi-socket system then ensures the patch applies to the local socket correctly, and (as needed) to the additional sockets.

If there are multiple updates, application of the packages must occur sequentially and in numerical order.

3.2.1.1 Enablement

BIOS developers enable SFS by compiling their BIOS with appropriate values for:

- `APCB_TOKEN_UID_PSP_SFS_ENABLE`
- `APCB_TOKEN_UID_PSP_SFS_COSIGN_REQUIRED`

Both options are off by default. Customers must opt-in and configure the BIOS to allow SFS and if desired co-sign the update packages.

Additionally, the following item should be set per customer needs:

- `APCB_TOKEN_UID_PSP_SFS_IB_OR_OOB`

This defaults to OFF (or 0) which means In-Band is the default method.

The security processor firmware consumes this information when authenticating whether to allow application of the patch. *Table 2* below explains the combinations of these two flags.

SFS enabled?	Co-signing enabled?	Behavior
No	N/A	SFS will not accept commands. BIOS will not accept a public key.
Yes	No	SFS will accept update packages and requires no customer co-signature (SFS will refuse a co-signed package)
Yes	Yes	SFS will accept update packages and require co-signing

Table 2. SFS/Co-signing Flags

In early boot, it is a requirement that the customer BIOS pass their signing key to the ASP, if set up. If both keys are enabled, the SFS package authenticates first against the customer key, and then against the AMD SFS key. This allows customers to create a BIOS with a co-signing key that runs against (for example) a test set of systems and then another one with a different co-signing key that runs on production systems only. This separation prevents SFS update packages, used for the test phase, from deploying on production systems.

Prior to OS start, the customer BIOS provides a public signing key by sending a message (below) to the BIOS-PSP interface. The firmware verifies that the signature of an SFS update package matches the customer in control of the machine during early boot.

3.2.1.2 SET_CUST_SFS_SIGNATURE

Signing requires the customer provide their public key (aka RSA public key) used for SFS package verification during early boot, specifically before BIOS_CMD_LOCK_DF_REG (MboxBiosCmdLockDFReg). The customer cannot update the signature after this stage.

The x86 should pass in a pointer to an `MBOX_BUFFER_HEADER`, defined as:

```
typedef struct
{
  UINT32    TotalSize;  ///< Total Size of MBOX_BUFFER (including this field)
  UINT32    Status;     //< Status value
} MBOX_BUFFER_HEADER;
```

The ReqBuffer follows immediately in memory behind the MBOX_BUFFER_HEADER as defined below:

```
typedef struct OEM_CO_SIGN_KEY
{
    UINT32 KeySize; //!< Indicator of 2K or 4K key in bits.
    //!< Struture which holds OEM Co sign key and its data
    UINT8  OemCoSignKey[SIZE_4K_OEM_CO_SIGN_KEY];
} OEM_CO_SIGN_KEY;
```

3.2.1.3 RSA_TOKEN_FORMAT

Customers can define the `OemCoSignKey` using the `RSA_TOKEN_FORMAT`.

To co-sign an update package received from AMD with RSA signing, the customer generates an RSA key pair (4096-bit) and signs the package with private key using RSA-PSS signature scheme `RSASSA_PKCS1_PSS_MGF1_SHA384` for a 4096-bit key (total size 1090 bytes)).

Byte Offset	Content	Description
00h	Version_ID	Version for the key format structure. Set the version as one.
04h	This_KEY_ID	Customer-created key ID
14h – 35h	Reserved	Must be zero
36h	ALGORITHM	Must be zero for RSA
37h	Reserved	Must be zero
38h	EXP_SIZE	Public exponent size in bits
3Ch	MOD_SIZE	Modulus size in bits
40h	EXP	N = 256 or 512 bytes depending on public exponent size
40h + N	MOD	N = 256 or 512 bytes depending on modulus size

Table 3. RSA Token Format

3.2.1.4 LMS_TOKEN_FORMAT

Customers can define the `OemCoSignKey` using the `LMS_ONLY_TOKEN_FORMAT`.

To co-sign an update package received from AMD with LMS (Only) signing, the customer generates an LMS key pair (using tree height of 20_8) and signs the package with private key using LMS signing.

Byte Offset	Content	Description
00h	Version_ID	Version for the key format structure. Set the version as one.

Byte Offset	Content	Description
04h	This_KEY_ID	Customer-created key ID
14h	Reserved	Must be zero
24h	Reserved	Must be zero
28h	Reserved	Must be zero
36h	Algorithm	3 -> LMS (not LMS/HSS)
37h	Reserved	Must be zero
38h -	PUBLIC_EXP	56 bytes that are the LMS public exponent
Xxh – FFh	Reserved	Must be zero

Table 4. LMS Token Format

3.2.1.5 Examining the Token

AMD provides a command for BIOS to get the SHA-384 hash of the provided customer co-signing token so the platform BIOS (pre-SMM lockdown) can determine what key has been sent to the ASP.

3.2.2 Versioning

Each executable patch checks the existing System Patch Level and/or specific firmware version(s) then determines if it is applicable to that existing patch/firmware level and the other firmware running in the system.

3.2.3 Patchable Items

An update package can patch a register or other value, replace a complete firmware image in a microprocessor, or update portions (e.g., drivers) of a firmware package.

3.2.3.1 SEV/SNP Update Note

AMD SFS will not commit any SEV update which would prevent rollback to previous images. If a customer wishes to commit an update, they can do so via the normal (x86 based) SNP_COMMIT command (see AMD Document 56860, SEV Secure Nested Paging Firmware ABI Specification).

The implication is that if an SEV FW update is applied, either the host must commit it or a second SEV FW update cannot be applied.

Per existing SEV/SNP rules, reverting a committed SFS updated SEV firmware image requires a system reboot.

When In-Band SFS is used and SNP is enabled, all pages used for the update package must be `HV-FIXED` pages (using the x86 `RMPUPDATE` command). The command should be issued by the driver/software sending the package to the SFS interface.

3.2.4 OS Notifications

For Out-Of-Band updates, AMD can use an ACPI General Event Notification that host SW can consume to indicate that an update has occurred.

***Note**: This is still a work in progress, and this section will be updated as a clear direction is determined.*

3.2.5 Security and Attestation Note

For attestation requirements, the OS or system/platform management software should request attestation data/evidence via appropriate interfaces & protocols (viz., side-band interface/SPDM or in-band/DPE. SFS provides required measurements to the appropriate attestation data managers for reporting via the above mechanisms.

Some attestation schemes prevent the update of certain firmware components of AMD-provided firmware. AMD will not attempt to modify that firmware.

3.2.6 Resiliency

Each update package is responsible for ensuring that the proper versions of the firmware updated and/or influenced by a patch exist on the system prior to beginning the update process. This ensures the firmware in the system does not get out-of-sync if there are any inter-firmware dependencies.

If any portion of a package cannot fully apply its updates, then application of the package should not occur. Otherwise, the only choice may be to force a reset, which is an undesirable outcome. System testing by AMD and the customer’s site will verify that a reset is not necessary.

Chapter 4 Boot Time Application Overview

Update packages are non-persistent across system reboots (warm or cold), so it is important to define a mechanism to apply patches. Customers have the option to apply update patches as part of OS boot, immediately after system boot, or at any time during system operation.

For example, the Linux OS places firmware updates in a location known to the OS, then loads the firmware from that location during system boot. SFS update patches have version numbers and a boot service (or driver install) and can apply patches sequentially by placing the update package files in a location known to the file system.

See *Section 5.2.1* for information on package naming.

Chapter 5 Sending SFS Commands

There are two ways to send commands to the SOC. *Section 5.1* describes issuing commands to the ASP via an IB method using a host driver. *Section 5.2* describes sending commands via an OOB method using a BMC or node controller. These methods are mutually exclusive and controlled by BIOS compile time tokens.

5.1 Issuing SFS Commands Out-Of-Band

For this use case, once an Update Package is prepared (which could be as delivered by AMD or as co-signed by a customer) it will be communicated by the customer management system (likely through the BMC) using MCTP. AMD OpenBMC code will support a command to target the SOC with this Update Package. If customers are using other BMC/Node Controller solutions, it is expected that they will deliver the Update Package via MCTP to the SOC.

MCTP is a public specification that be reviewed here: *MCTP Spec*.

MPIO presents a discoverable endpoint for MCTP. It will also handle the complete SFS communications with the rest of the SOC and ensure that status is returned via the TEE Extended command buffer and, for SFS_GET_FW_VERSIONS, the output for the command(s).

AMD utilizes the MCTP Type 0x7F (OEM Specific) with the specific commands of 0x01 representing the SFS_GET_FW_VERSIONS command which causes SFS to return two 4k pages that match the specific output documented in *Section 5.2.1* and *5.4.1* and 0x02 representing the SFS_UPDATE command which takes an SFS update package and returns a single 4k page which represents the return of the TEE Extended command buffer as described in *Section 5.2.1*.

Figure 2 shows the flow for the OOB solution:

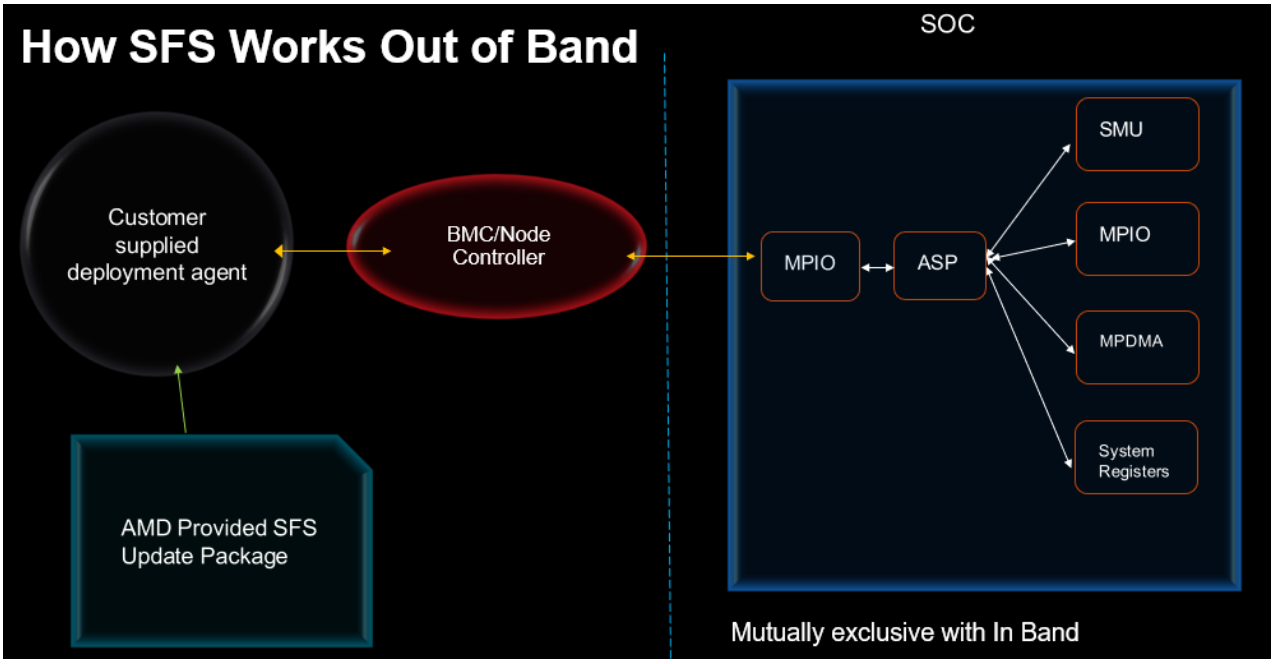

Figure 2. Out of Band SFS Diagram

And the flow is shown below:

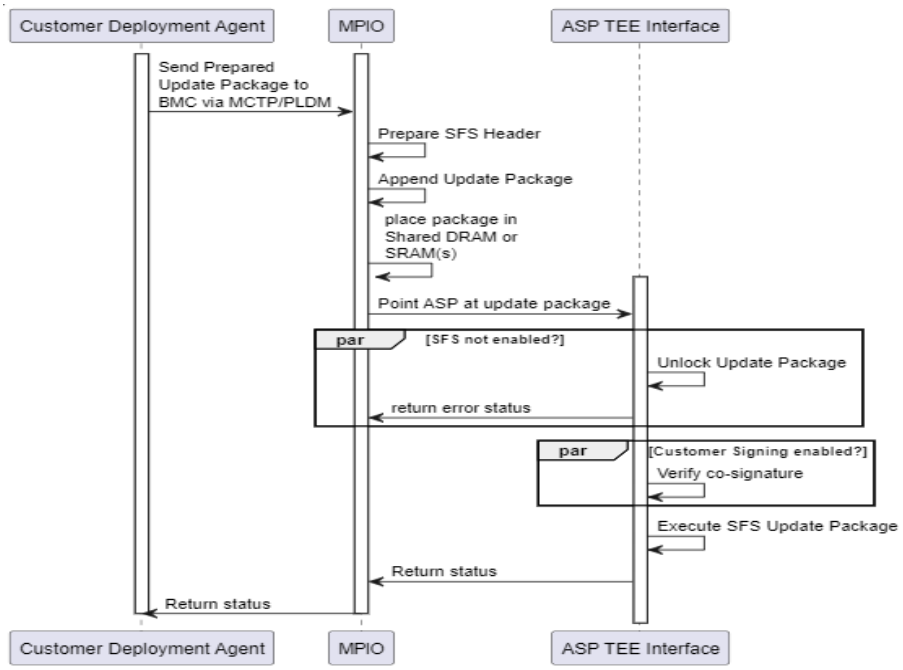

Figure 3. Out of Band SFS Flow

What is returned is the TEE Extended Command Buffer, as shown below in *Section 5.2.1*.

5.1.1 Notifications of SFS Update Completion to Host Software

To enable host software notifications when Seamless Firmware Servicing (SFS) operates in Out-of-Band (OOB) mode, the following steps were implemented:

Overview:

- BIOS reports an SCI signal and runtime memory buffer address to ASP using the BIOS ↔ ASP mailbox interface.
- The AMD ACPI ASPT table provides the necessary information for host software to manage notifications.
- This mechanism is valid only when OOB is the SFS method.
- GPE _L method to handle SCI.
- _L method notifies the host SW about the SFS update.

5.1.1.1 Step 1 – Enabling Communications from ASP to Host SW

At boot time, platform BIOS reserves a memory area for SFS to store status and extended status (two 32-bit words). The reserved address is set in the ASPT table during build time.

5.1.1.2 Step 2 – Informing SFS Where to Write Status/Extended Status

Platform BIOS communicates the reserved memory address to ASP via a BIOS command, enabling SFS to update its status.

5.1.1.3 Step 3 – Informing SFS Which Bit as an SCI Register Is Used to Trigger a Notification

Platform BIOS informs SFS which SCI register bit should be used to trigger a notification to the host software. _L method in GPE ACPI namespace would service the SCI.

5.1.1.4 Step 4 – Host SW Gets and Acts Upon the SCI

Upon receiving the SCI signal, _L method notifies the host software, and the host software retrieves the status from the reserved memory area and acts accordingly—or ignores the notification if not required.

5.1.1.5 Data Structures

```
typedef struct SCI_TRIGGER_INFO
{
    uint64_t Address;// Physical address where SCI MBox will be located
    uint32_t AllocatedSize;
    uint32_t SciBit; //Bit position in the register assigned for ASP
SCI
    uint32_t Reserved;
} SCI_TRIGGER_INFO;
```

BIOS ASP Mailbox command

```
BIOS_CMD_SET_SCI_INFO                  = 0x7D,  /*!< BIOS -> PSP: Setup
SCI Mailbox information */

SCI General Purpose Event
Sci bit 28 is assigned.
```

ACPI Device Object

```
DefinitionBlock (
    "SFSV2.aml",               // Output file
    "SSDT",                        // Signature
    0x02,                            // SSDT Revision
    "AMDINC",                   // OEM ID
    "SFS-UPV2",                // OEM Table ID
    0x1                              // OEM Revision
    )
{
  Scope (_GPE) {
    Method(_L1F, 0) {
      Notify(\_SB.SFUP, 0x80)
    }
  }

  Scope (\_SB) {
    Device (SFUP) {                 // SFS Update ACPI Device
      Name (_HID, "AMDI0098")
      Name (_UID, "SFS2")

    }
  }
}
```

5.2 Issuing SFS Commands In-Band

The In-Band method of sending an update package to the SOC is similar, with more details available in sections below. Refer to *Figure 4* to see how the IB solution works.

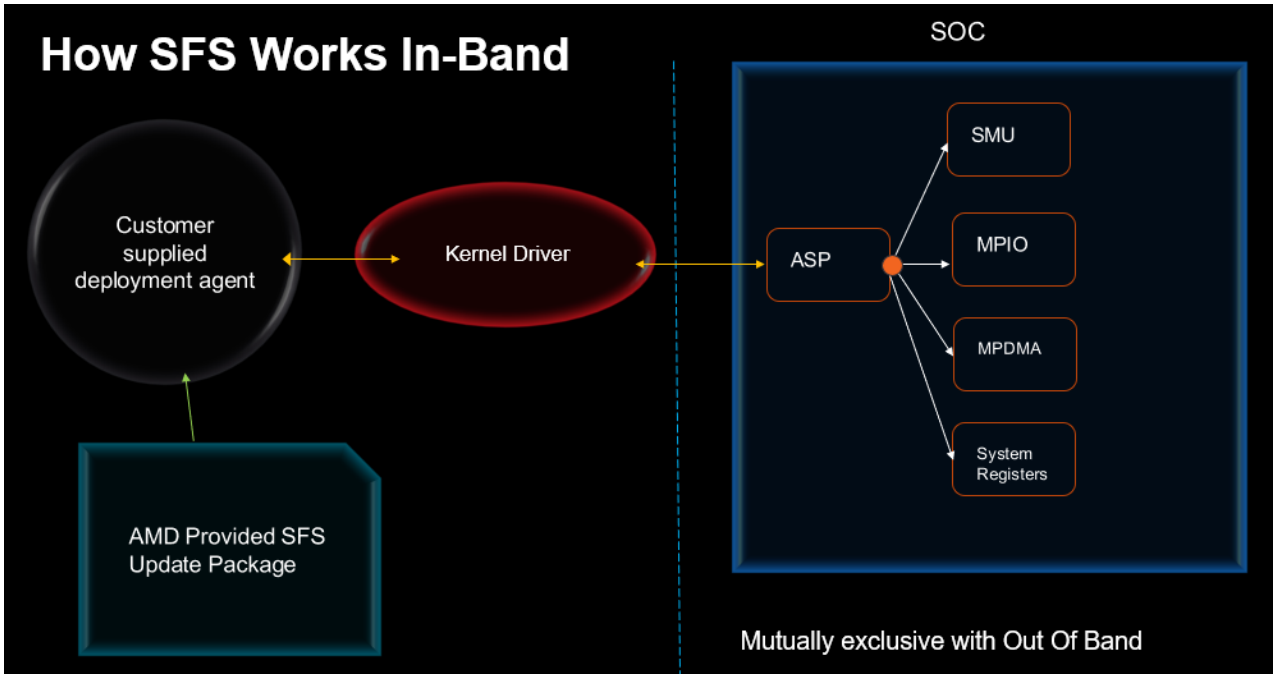

Figure 4. In-Band SFS Update Flow Graphic

Issuing SFS commands through the kernel driver requires the following:

1. Memory alignment
    a. The page used for the TEE_EXT_CMD_BUFFER must be at the start of a 2MB aligned page that is 2MB in size.
    ***NOTE**: This 2MB page must be flushed to memory before issuing the SFS command to the TEE interface (or marked as non-cacheable).*
    b. For the GET_FW_VERSIONS command, the output page must follow the TEE_EXT_CMD_BUFFER address + 4k.
    c. For the UPDATE_FW command, the update package must start at the address of the TEE_EXT_CMD_BUFFER + 4k. If the customer has co-signed the update package, the customer co-signing header must start at TEE_EXT_CMD_BUFFER + 4k.
2. Customer-provided SFS Deployment Agent
    a. Requirements
        i. The TEE_EXT_CMD_BUFFER must be 2MB aligned.
    b. Process
        i. Create a TEE_EXT_CMD_BUFFER (see *Table 4*) in DRAM.
        ii. Appends an SFS co-signing header in DRAM (see below for definition) if customer co-signing is enabled.
        iii. Appends the entire AMD provided update package.

iv. Appends the co-signing signature (if used) at the end of the AMD provided update package.
***NOTE**: the co-signature must cover the Co-signature header and AMD Update package provided.*

v. Communicates to the primary ASP firmware the request for SFS by putting an SFS command in the TEE mailbox interface and passing the physical address of the SFS TEE_EXT_CMD_BUFFER in memory into additional mailboxes (see *Section 5.2.4*).

The process of applying an SFS update package follows the flow indicated below (*Figure 5*).

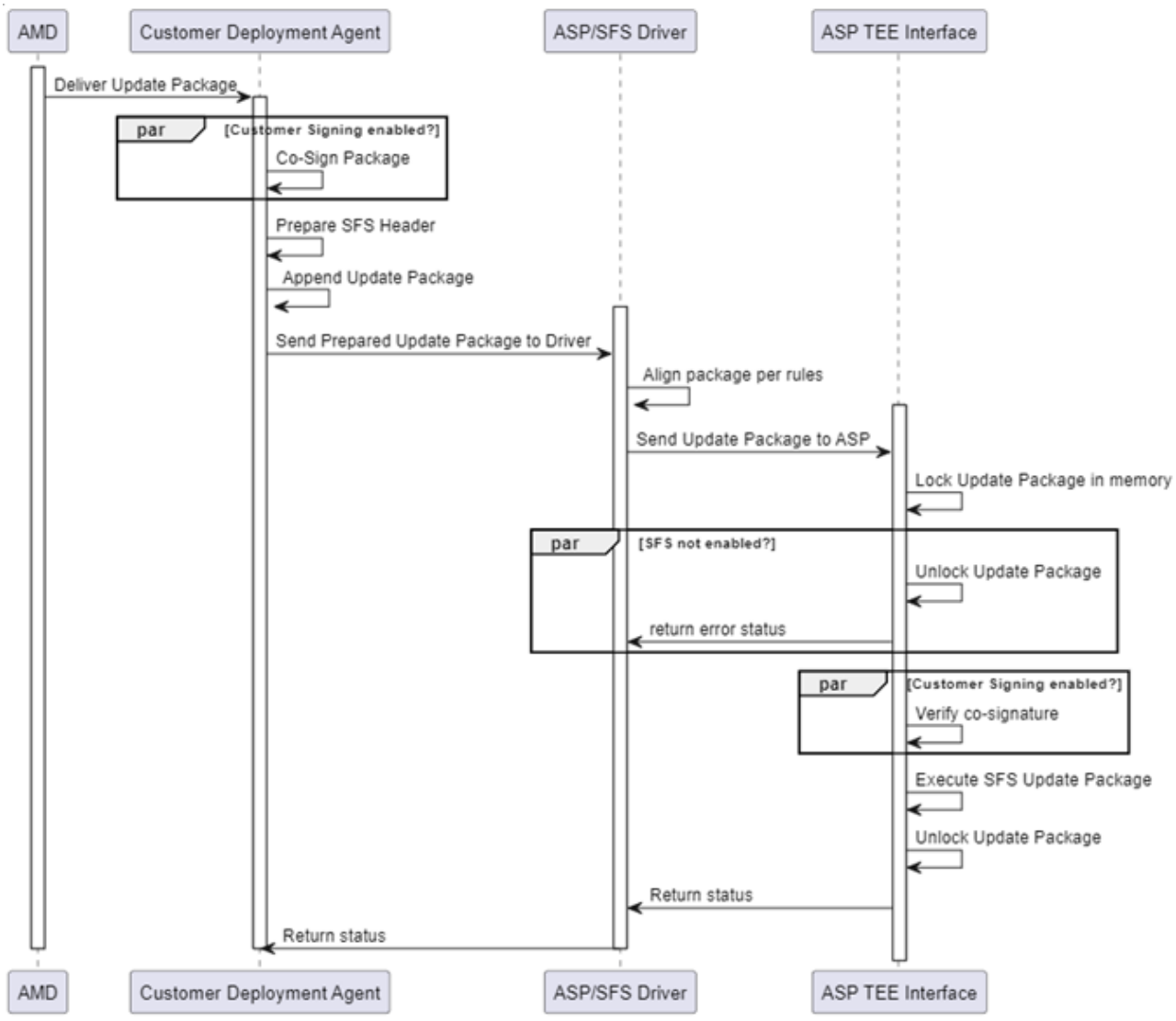

Figure 5. In-Band SFS Update Flow

For a detailed verbal description of applying an SFS update package, see below (*Figure 6*).

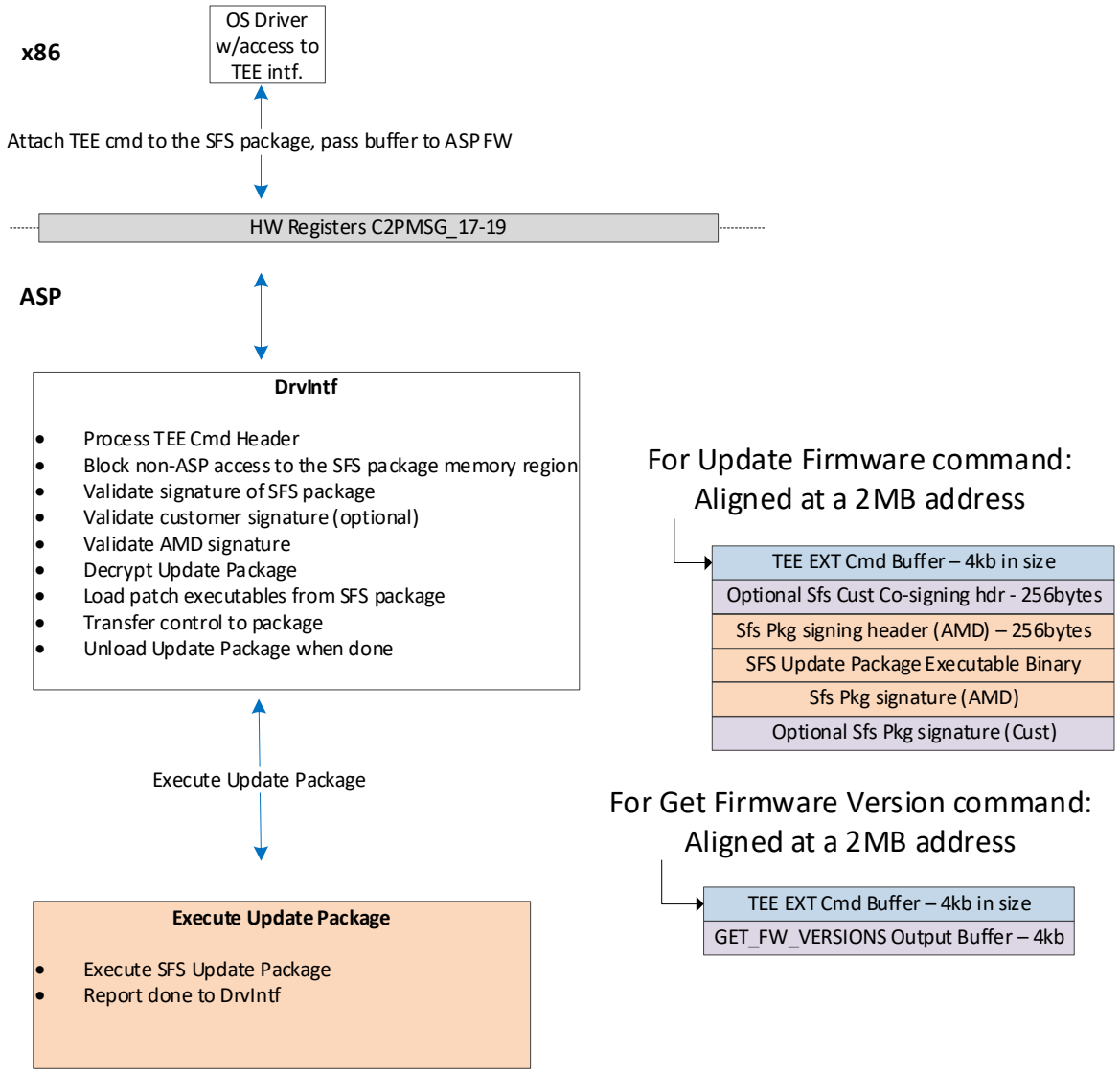

Figure 6. Patch Application Process

5.2.1 TEE Extended Command Buffer (TEE_EXT_CMD_BUFFER)

The structure consists of a `TEE_EXT_CMD` padded to a 4k size; represented by the following example:

```
typedef struct TEE_EXT_CMD_BUFFER
{
    TEE_EXT_CMD Command;
    // For alignment and future expansion
    TEE_UINT8   Reserved[ TEE_EXT_CMD_MAX_SIZE - sizeof(TEE_EXT_CMD) ];
} TEE_EXT_CMD_BUFFER;
```

5.2.2 TEE Extended Command Structure (TEE_EXT_CMD)

This structure is as follows. For SFS, use the following structure for the TEE_EXT_SUB_COMMAND.

```
typedef struct TEE_EXT_CMD
{
    TEE_EXT_CMD_HEADER   Header;
    TEE_EXT_SUB_COMMAND  ExtSubCmd;
} TEE_EXT_CMD;
```

5.2.3 TEE Extended Sub Command Header (TEE_EXT_SUB_COMMAND)

Byte Offset	Bits	Dir	Name	Description
00h	31:0	In	TotalSize	Total Size of EXT_CMD_BUFFER (including this header) in bytes rounded up to the next 64k.
04h	31:0	In	SubCmdId	See TEE SFS Sub-Command ID values below (*Table 5*)
08h	31:0	Out	Status	Command Execution Status
0ch	31:0	Out	EXT_STATUS	Extended Status (Please report these values to AMD if Status != SFS_SUCCESS)
10h-fffh	31:0	-	Reserved	

Table 5. TEE Sub-Command Header

TEE also supports additional SFS-specific commands.

Command ID Value	Command ID Name	Description
1h	TEE_SUB_CMD_SFS_GET_FW_VERSIONS	
2h	TEE_SUB_CMD_SFS_UPDATE	
3h	Reserved	

Table 6. TEE SFS Sub-Command ID Values

SFS reports statuses using a full 32-bit word values shown in *Table 12*.

5.2.4 Sending the SFS Command to the ASP

Simplified algorithm for sending this command (*Table 8*). Access to the ASP TEE registers is via the ASP PCI driver.

1. Software needs to interrogate Bit 3 (SFS_ENABLED) of the ASP Feature Register (MP0_C2PMSG63) to determine if SFS is enabled at all.
2. Client polls on Ready flag of the TEE Command/Status Register until set to 1.
3. Write the physical address of the SFS Command Header into the TEE CmdRspBufAddr_Lo and CmdRspBufAddr_Hi registers.
4. Write the TEE_IF_EXT_CMD_ID into the Command/Status Register[Command ID] field and write zero to all other fields and update the Command/Status Register.
5. Loop on Ready flag of Command/Status Register while 0 (command in progress).
6. At this point, the Status field of the Reload Firmware Header is valid and combined with the Status in the Command/Status Register, reflects the Reload operations' status.

Register Name	Bits	Name	Description
Command/Status Register	31	Ready	Set by the target to indicate the mailbox interface state 0 – Not ready to handle commands (or handling previous command) 1 – Ready to handle next command
	30:20	-	reserved
	19:16	Command ID	0x0E – TEE Extended Command
	15:0	Status	Set by the ASP to indicate the execution status of the last command
CmdRspBufAddr_Lo	31:0	Addr_Lo	Lower 32-bits of physical address of Command/Response Buffer (TEE EXT CMD Header)
CmdRspBufAddr_Hi	31:0	Addr_Hi	Upper 32-bits of physical address of Command/Response Buffer (TEE EXT CMD Header)

Table 7. ASP Register Definitions

Relative to the base address assigned by BIOS to the PSP device's memory mapped I/O space, Command/Status is register 17 (offset 68), CmdRspBufAddr_Lo is register 18 (offset 72), and CmdRspBufAddr_Hi is register 19 (offset 76).

5.2.5 ASP Process of SFS

1. ASP checks that the DRAM address can be protected from x86 access.
2. ASP verifies the customer signature (if enabled and provided to ASP).
3. ASP verifies the AMD signature.
4. ASP loads and runs the executable patch.
5. ASP updates the current patch level information.
6. ASP returns the status to the software agent.

5.3 Sending a Customer Co-Signed Update Package

This consists of the optional customer co-signature header, followed by the mandatory AMD signature, and the SFS update package itself. The required signatures follow - AMD (and optionally the co-signature). Note that the co-signature covers the AMD Update Package as well as 256 byte co-signing header.

Byte Offset	Name	Description
00h – 0Fh	Reserved1	Must be zero
10h – 13h	Cookie	Required. Must be set to 5453_5543h
14h – 17h	SizeFWSigned	Size of the package in bytes. This would be the entire AMD update package.
18h - 33h	Reserved2	Must be zero
34h – 37h	Algorithm	Supported Algorithm IDs 2 → RSA-PSS-SHA384 with 4096-bit key size 3 → HSS of 2 level LMS trees (height 5 and 15), w=8
38h – FFh	Reserved3	Must be zero

Table 8. Optional SFS Co-signing Header

5.4 SFS Sub-Commands and Status

SFS supports the following sub-commands.

5.4.1 TEE_SUB_CMD_SFS_GET_FW_VERSIONS

The ASP interface can provide the current level of base firmware for the ASP (and other microprocessors) as well as the current patch level(s).

This command consists of two pages sent to the ASP interface. The first page is the TEE_EXT_CMD_BUFFER which must be exactly one page (4096 bytes). The output data from the command (also 4096 bytes) uses the second page.

To check the firmware versions, the GET_SFS_VERSION_INFO command passes into the SFS handler in the ASP as a subcommand in the SFS Command Header. The address values passed in would point to a contiguous two-page area. The output buffer needs to initialize to C7h in every byte. For SNP systems, the page needs to be an HV-FIXED page. On successful command completion, that buffer contains the data as specified below (All reserved fields are zero).

Items not supported have a 0000_0000h value in the item's field.

Offset	Bits	Name	Description
00h	31:0	SFS_API_VERSION	The version of the API that this command supports
04h	31:0	CURRENT_PATCH_LVL	The current SFS patch level as an integer (SFSPL)
08h	31:0	SYS_PATCH_LVL	The current System Patch Level as an integer (SYSPL)
0Ch	31:0	NUM_SUPPORTED_FWS	Number of FW Versions captured out of MAX (MAX = 300)
10h-E1Fh	31:0	FW_VERSION_INFO	See below
E20h-FCFh	31:0	Reserved	432 bytes, MBZ
FD0h-FFFh	31:0	CO_SIGN_TOKEN_HASH	A SHA-384 hash of the co-signing token sent to the ASP (48 bytes or 12 words)

Table 9. GET_SFS_VERSION_INFO Entry

Note: *Table 9 is subject to change.*

Each FW_VERSION_INFO entry is a triplet consisting of:

Offset	Bits	Name	Description
00h	31:0	VERSION_TYPE	The Enum of the version type (see *Table 11*)
04h	31:0	VERSION	The version of the indicated VERSION_TYPE
08h	31:0	SEC_PATCH_LVL	The current Security Patch Level (SPL) as an integer

Table 10. FW_VERSION_INFO Entry

Firmware IDs for VERSION_TYPE:

Note: *Not all these items are currently SFS updatable.*

FW_NAME	FWID	Description
DRV_SYS_VERSION	000Ah	ASP System Driver Version
DRV_SOC_VERSION	014Ah	ASP SOC Driver Version
DRV_HAD_VERSION	014Bh	ASP High Availability Driver (Debug) Version
DRV_BOOT_VERSION	0150h	ASP Boot Driver Version
DRV_INTF_VERSION	014Ch	ASP Interface Driver Version
DRV_RAS_VERSION	0152h	ASP RAS Driver Version
DRV_SEV_VERSION	0151h	ASP SEV Driver Version
DRV_SPDM_VERSION	01C7h	ASP SPDM Driver Version
DRV_IPKEYMGR_VERSION	01EDh	ASP IP Key Manager Version
DRV_SFS_VERSION	01D9h	SFS Driver Version
AGESA_BL_VERSION	000Fh	AGESA Boot Loader Version
ASP_VERSION	0009h	ASP Secure OS Version
PMFW_VERSION	01D2h	Power Management Firmware Version
SDXI_VERSION	1048h	SDXI Firmware Version

FW_NAME	FWID	Description
TMPM_VERSION	1049h	Tiered Memory Page Migration Firmware Version
SMN_SRAM_VERSION	1001h	This is a made-up FWID for this type of update
MP5_VERSION	104Ch	MP5 Firmware Version
MPIO_VERSION	101Bh	MPIO Firmware Version
SFS_UPDATE_PACKAGE	1D8h	Update Package Measurement
SFS_DRIVER	1D9h	The SFS Driver Version
SFS_SUB_DRIVER	1Deh	The SFS Subdriver, if used
ART_FMC	1CCh	The ART First Mutable Code
ART_RUNTIME	1CDh	The ART Runtime Code
CALIPTRA_FMC	1E5h	The Caliptra First Mutable Code
CALIPTRA_RUNTIME	1E6h	The Caliptra Runtime Code
uCode	104Eh	uCode FWID
MPRAS.PFEH_APPLET	1068h	PFEH Applet for MPRAS
MPRAS.PCIE_APPLET	1069h	PCIE Applet for MPRAS
MPRAS.CXL_APPLET	106Ah	CXL Applet for MPRAS

Table 11. Firmware Enums for VERSION_TYPE

5.4.2 TEE_SUB_CMD_SFS_UPDATE

This sub-command tells the ASP to load, verify, and execute the SFS package. The package does not apply any updates until all parts of the update package complete validation as appropriate.

This command consists of a buffer the size of the TEE_EXT_CMD_BUFFER (1 page) followed by the SFS Update Package sent to the ASP interface.

5.4.3 SFS Status Codes

Status Code	Name	Description
00h	SFS_SUCCESS	Success – patch applied
01h	Reserved	
02h	SFS_INVALID_TOTAL_SIZE	Invalid TotalSize
03h	Reserved	
04h	SFS_INVALID_PKG_SIZE	Invalid Image size
05h	SFS_DISABLED	Patching not allowed
06h	SFS_INVALID_CUSTOMER_SIGNATURE	Invalid Customer Signature
07h	SFS_INVALID_AMD_SIGNATURE	Invalid AMD signature
08h	SFS_INTERNAL_ERROR	Please report the extended status to AMD.
09h	SFS_CUSTOMER_SIGNING_ NOT_ALLOWED	Customer signed but not allowed
0ah	SFS_INVALID_BASE_PATCH_LEVEL	Invalid base patch level – Base FW Version mismatch
0bh	SFS_INVALID_CURRENT_PATCH_LEVEL	Invalid current SFS patch level – current patch level mismatch
0ch	SFS_INVALID_NEW_PATCH_LEVEL	Invalid new SFS patch level – less than current patch level
0dh	SFS_INVALID_SUBCOMMAND	Invalid SFS subcommand
0eh	SFS_PROTECTION_FAIL	Payload cannot be protected - possibly not aligned on 2MB boundary.
0fh	SFS_BUSY	Busy – SFS cannot update with this package
10h	SFS_FW_VERSION_MISMATCH	The uploaded FW is less than the existing FW.
11h	SFS_SYSTEM_VERSION_MISMATCH	The current SYS patch level and new SYS patch level are not one apart.

Status Code	Name	Description
12h	SFS_SEV_STILL_INITIALIZED	SEV and SEV/ES clients and SEV SHUTDOWN need to have happened to update SEV.
13h	SFS_SUCCESS_PARTIAL	Indicates that SFS Updates were successful, but notification did not work. Please report the extended status to AMD.
14h	SFS_UCODE_PATCHING_DISABLED	Microcode updates are disabled on this platform.
15h	SFS_SEV_NOT_COMMITTED	SEV cannot be updated a second time if the first update has not been committed, which is a customer responsibility.
16h	SFS_OOB_UPDATES _NOT_ENABLED	IB Updates are selected by configuration, so OOB is not enabled
17h	SFS_IB_UPDATES_NOT_ENABLED	OOB Updates are selected by configuration, so IB is not enabled
18h	SFS_UCODE_UPDATES_NOT_COMPLETED	Not all threads/cores had their uCode updated successfully.

Table 12. SFS Status Codes

Chapter 6 SFS Package Release

6.1 Patch Naming

SFS packages are binary objects that a customer maintains and aligns to AGESA releases.

Package names follow this example:

`fam_<family>-<AGESA-Release>-SFS-<n>.pkg` (e.g., `fam_19h-1006-SFS-1.pkg`)

where:

- `<AGESA-Release>` is AMD's PI release number (e.g., 1002 or 1006).
- `<n>` is a monotonically increasing counting number (i.e., integer) starting at 1.

6.2 SFS Update Package Metadata Header

Located after the AMD Signing Header (0x100 byes after the beginning of the Update Package) is a metadata header that is provided for consumption by users of SFS, such as the customer infrastructure. It contains various data used to describe the contents of the update package. It consists of the following:

Size	Name	Description
4b	HDR_STR	The String SFSM
4b	SIZE	The size of this metadata starting at HDR_STR
4b	MD_STRUCT_VER	The version of this metadata structure
4b	TOTAL_SIZE	The total size of the update package from AMD signing header to the end of the AMD signature (does not include any co-signing header/signature)
4b	SFS_API_VER	The version of the SFS API this package works with
4b	SFSPL	The SFS Patch level. This is a monotonically increasing number (from 0) that counts how many update packages have been applied
4b	REQ_SFSPL	The required SFSPL that this update package requires in order to apply the updates

Size	Name	Description
4b	SYSPL	The SYStem Patch Level that this update package will update to (can be zero, meaning there is no system patch applied)
4b	REQ_SYSPL	The required SYSPL that this update package requires in order to apply the updates
16b	CoRIM_ID	This is the 16 byte CoRIM magic string
4b	NUM_PAYLOADS	The number of payloads this update package contains. For each payload, the next three words are repeated
4b	FWID	The Firmware Identifier of a payload in this update package. These IDs are those used in Get Firmware IDs
4b	REQ_FW_VER	The required Firmware Version for this payload in order for the update to proceed
4b	EXP_FW_VER	The expected Firmware Version for this payload after a successful update

Table 13. SFS Update Package Metadata Header

And can be implemented as:

```
typdef struct {
  uint8_t  HDR_STR[4] = "SFSM"
  uint32_t METADATA_SIZE;
  uint32_t METADATA_VER;
  uint32_t UPDATE_PACKAGE_SIZE;
  uint32_t SFS_API_VER;
  uint32_t SFSPL;
  uint32_t REQ_SFSPL;
  uint32_t SYSPL;
  uint32_t REQ_SYSPL;
  uint8_t[16] CoRIM_ID;
  uint32_t NUM_PAYLOADS;
  struct {
    uint32_t FW_ID;
    uint32_t REQ_FW_VER;
    uint32_t NEW_FW_VER;
    } [];
  }
```

6.3 Release Mechanism

Platform provider and large-scale data fleet operators can work through their AMD customer representatives for SFS update packages.

Chapter 7 Operational Considerations

Update packages have a specific set of firmware and system patch level versions to which they apply; therefore, any update package released must match the specific combination of firmware running on a given system to apply the package.

The update process runs in parallel with x86 and DMA traffic. The goal of package design is that updates have a minimal impact on system performance, at most requiring a brief quiescence of certain activities. Optimally, updates require no quiescing from the OS perspective; therefore, any updates, module, and micro-processor firmware replacements will look at worst like a small slowdown in response – well within the latency windows of the AMD data fabric and customer expectations.

There are very few SFS operations that would require quiescing OS operations or portions thereof. One example of a more impactful update (but still not requiring a system restart) is that SEV or SEV/ES virtual machines need to stop and then restart for an SEV firmware update. SEV/SNP virtual machines are capable of continuing operation across an SFS update to the SEV firmware. Another example is SDXI. If SDXI is running many virtual functions with very large transfer sizes, quiescing that traffic that is in flight and updating the firmware may take more than 100ms.

Error Handling

SFS can detect and handle many types of errors during updates. Some types of errors are outside the ability of SFS to handle gracefully. In those cases, AMD's Machine Check Architecture will log the conditions leading to the error, and information about the error itself.

Uncorrectable DRAM Errors

In the event of a system memory error that prevents DRAM access during an SFS operation, the ASP will trigger a cold-reset recovery.

Firmware Update Errors

In the event the ASP encounters a serious error updating its own internal firmware, a warm-reset recovery may be triggered. If an update error leaves the system in an inconsistent state, the ASP will report this condition, allowing the platform control plane to perform gracefully restart the system.

7.1 Error Handling

SFS is capable of detecting and handling many types of errors during updates. There exists some types of errors that are outside the capability of SFS to handle gracefully. In those cases, AMD's

Machine Check Architecture will provide logging of the conditions leading to the error, and possibly information about the error itself.

7.1.1 DRAM Unavailable

In the event of a system memory error that prevents DRAM access during an SFS operation, the ASP - which handles most SFS processing, will know that DRAM is not accessible and is capable of triggering cold-reset recovery. This happens for ASP-internal as well as microprocessors in the control plane path of the SOC.

7.1.2 Firmware Update Errors

In the event the ASP is updating its own internal firmware, it can cooperate with the other internal microprocessors in the SOC and trigger a warm-reset recovery in the event that the SFS code cannot manage the error itself. In the event the firmware state is inconsistent, ASP can report that allowing the platform control plane to perform recovery (save things, as possible, or notify and reset as needed.

7.2 Micro Code (uCode) Updates

AMD's implementation of delivering and handling uCode updates (available in Family 1Ah, Models 50h-5Fh) works as follows:

There is an option that allows/prevents uCode updating. SFS will report UCODE_PATCHING_DISABLED if the option is set to prevent updates.

All of these APCB's are build-time and not changeable at runtime.

1. SFS must be enabled on the platform
2. uCode updates must be enabled
3. Any uCode update payload will be the only payload in a given update package
4. For In-Band and Out-Of-Band delivery of uCode updates, SFS will:
    a. Validate the optional customer co-signature (if present and enabled)
    b. Validate AMD's update package signature
    c. Validate AMD's uCode signature
    d. Update the uCode on all thread/CCD's (AMD does not support updating some but not all CCDs/Threads)
    e. Determine success (all updates complete) or failure
    f. SFS will signal the host SW for OOB (which can optionally consume the signal)
    g. Updates the version information if successful
    h. Return an appropriate success/failure code