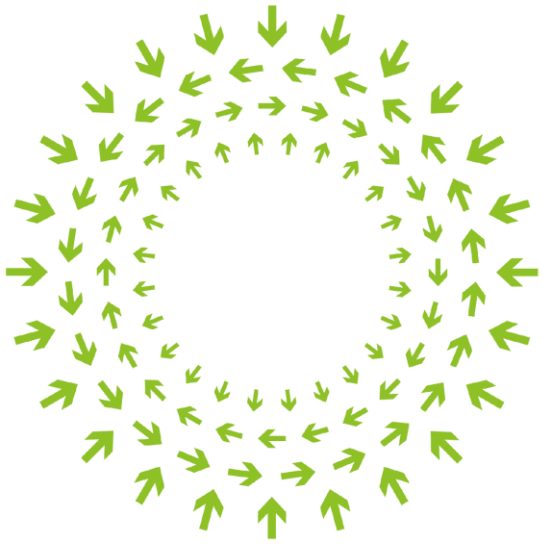

OPEN Compute Project

HYPERSCALE CPU IMPACTLESS FIRMWARE UPDATES REQUIREMENTS

Specification version 0.7

Effective 9/12/2025

Authors:

[Giri Mudusuru, Alex Akesson, Rama Bhimanadhuni, Rajat Madhusudan **Microsoft**]

[Jicksen Joy, Sean Malley **Google**]

[Josef Zeevi, Justin King **AMD**]

[Mathias Brossard, Jose Marinho, Gareth Stockwell, Daniil Egranov **Arm**]

[Anil Agrawal, Balaji Vembu **Meta**]

[John Leung, Allison Goodman, Nilanjan Palit, Jonathan Bentley **Intel**]

Contact: hyperscale-cpu@ocproject.net

Table of Contents

Table of Contents 2
Table of Figures 3
Revision History 4
License 5
About Open Compute Project Foundation 7
Executive Summary 8
1 Introduction 9
1.1 Definitions 10
1.2 Problem Statement 11
1.3 Goals 12
1.4 Scope 14
1.5 Not in Scope 14
2 Impactless Firmware Update Requirements 15
2.1 Concepts 15
2.2 Impactless Update Requirements 15
3 Impactless Firmware Update Model 18
3.1 General Flow 18
3.2 Impactless Update Status Codes 22
3.3 Development Model 23
3.4 Notifications Mechanisms 24
3.5 In-Band (IB) Deployment 25
3.6 Out-Of-Band (OOB) deployment 26
3.7 Package Format 27
4 Microcode-Specific Details 29

4.1 Microcode Update Categories 29
4.2 Microcode-specific Impactless Update Requirements 29
5 Runtime BIOS & Management Mode (MM) Firmware Impactless Updates 31
6 Compliance tools 32
7 References 33
8 OCP Tenets 34
9 Appendices 35
9.1 Appendix A - Glossary of Terms 35

Table of Figures

Figure 1: Impactless Firmware Update Development Model 23
Figure 2: Impactless Update Notifications 24
Figure 3: In Band Impactless Update Flow 25
Figure 4: Out of Band Impactless Update Flow 26
Figure 5: Package Format 27

Revision History

Revision	Date	Revision History
0.5	5/1/25	• Original
0.7	9/12/25	• Introduced Status Codes • Expanded architecture support to cover a broader range of platforms • Clarified and made explicit the IB (In-Band) update mechanisms • Enhanced security controls for improved protection • Improved lifecycle traceability for better management and oversight • Added GitHub links to Compliance Tools and refined framework details • Streamlined adoption and improved interoperability within hyperscale environments across vendors and CSPs.

License

Open Web Foundation (OWF) CLA

Contributions to this Specification are made under the terms and conditions set forth in **Modified Open Web Foundation Agreement 0.9 (OWFa 0.9)**. (As of October 16, 2024) ("Contribution License") by:

AMD, Arm, Google, Intel, Meta, Microsoft

Usage of this Specification is governed by the terms and conditions set forth in **Modified OWFa 0.9 Final Specification Agreement (FSA)** (As of October 16, 2024) **("Specification License").**

You can review the applicable Specification License(s) referenced above by the contributors to this Specification on the OCP website at https://www.opencompute.org/contributions/templates-agreements.

For actual executed copies of either agreement, please contact OCP directly.

Notes:

The above license does not apply to the Appendix or Appendices. The information in the Appendix or Appendices is for reference only and non-normative in nature.

NOTWITHSTANDING THE FOREGOING LICENSES, THIS SPECIFICATION IS PROVIDED BY OCP "AS IS" AND OCP EXPRESSLY DISCLAIMS ANY WARRANTIES (EXPRESS, IMPLIED, OR OTHERWISE), INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, OR TITLE, RELATED TO THE SPECIFICATION. NOTICE IS HEREBY GIVEN, THAT OTHER RIGHTS NOT GRANTED AS SET FORTH ABOVE, INCLUDING WITHOUT LIMITATION, RIGHTS OF THIRD PARTIES WHO DID NOT EXECUTE THE ABOVE LICENSES, MAY BE IMPLICATED BY THE IMPLEMENTATION OF OR COMPLIANCE WITH THIS SPECIFICATION. OCP IS NOT RESPONSIBLE FOR IDENTIFYING RIGHTS FOR WHICH A LICENSE MAY BE REQUIRED IN ORDER TO IMPLEMENT THIS SPECIFICATION. THE ENTIRE RISK AS TO IMPLEMENTING OR OTHERWISE USING THE SPECIFICATION IS ASSUMED BY YOU. IN NO EVENT WILL OCP BE LIABLE TO YOU FOR ANY MONETARY DAMAGES WITH RESPECT TO ANY CLAIMS RELATED TO, OR ARISING OUT OF YOUR USE OF THIS SPECIFICATION, INCLUDING BUT NOT LIMITED TO ANY LIABILITY FOR LOST PROFITS OR ANY

CONSEQUENTIAL, INCIDENTAL, INDIRECT, SPECIAL OR PUNITIVE DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND EVEN IF OCP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

About Open Compute Project Foundation

The Open Compute Project Foundation is a 501(c)(6) organization which was founded in 2011 by Facebook, Intel, and Rackspace. Our mission is to apply the benefits of open source to hardware and rapidly increase the pace of innovation in, near and around the data center and beyond. The Open Compute Project (OCP) is a collaborative community focused on redesigning hardware technology to efficiently support the growing demands on compute infrastructure. For more information about OCP, please visit us at http://www.opencompute.org

Executive Summary

In modern data centers, maintaining system uptime and reliability is crucial. Traditional firmware updates often present challenges such as downtime, operational disruptions, and resource-intensive processes. By reducing the need for downtime and minimizing operational disruptions, impactless updates aim to improve system reliability and performance. This specification outlines the methodologies and technologies used to achieve seamless firmware updates, ensuring hyperscalers can maintain continuous operation without compromising security or functionality.

This specification standardizes In-band (IB) and Out of Band (OOB) runtime non-persistent FW Impactless Updates capabilities for CPUs. With the explosion of hyperscale Data Centers supporting AI, HPC, Compute, and General Purpose workloads it is essential to have a standard for impactless updates to maximize operational efficiency and interoperability. The primary goals of this specification are to (a) increase interoperability by standardizing impactless updates interfaces and mechanisms; (b) increase the number of components that can be updated impactlessly.

This document specifies requirements for impactless updates of non-host SW components. The goal is to leverage the Open Compute Project to disseminate these requirements to CPU suppliers and hyperscalers, providing mutual benefits and advancing the industry.

1 Introduction

In the dynamic and demanding environment of modern data centers, maintaining system uptime and reliability is paramount. Traditional firmware updates as part of good firmware hygiene, while necessary, often pose significant challenges including downtime, operational disruptions, and resource-intensive processes. These updates typically necessitate system reboots, which can interrupt critical services and lead to financial losses. By reducing the need for downtime and minimizing operational disruptions, impactless updates aim to enhance system reliability and performance. This specification outlines the methodologies and technologies employed to achieve impactless firmware updates, ensuring that data centers can maintain continuous operation without sacrificing security or functionality.

Impactless firmware updates aim to address the challenges associated with traditional updates by allowing fixes to be deployed without downtime or complex coordination while optimizing data center resource allocation. By understanding and implementing these specifications, organizations can enhance their operational efficiency and resilience.

This specification standardizes runtime non-persistent Impactless In-band (IB) and Out of Band (OOB) updates for CPU SoCs in a Hyperscale environment. With the demanding uptime needs of modern data centers, growth of multi-supplier CPU fleets, and the increase in frequency of updates, ensuring a standard approach to impactless updates will provide enhanced operational efficiency, security, and general interoperability.

For the remainder of this specification, the terms “CPU” and “device” will be used interchangeably. In all cases, the term refers to the product shipped by the device supplier, which may include multiple components residing on an OCP Universal Baseboard (UBB) as well as the firmware and software stacks (e.g., device drivers) used to operate the device.

Impactless updates preserve compatibility with the most recent impactful baseline and enable delivery of critical security fixes and/or feature enhancements with minimal or no service interruption. Adoption is not universal or uniform: each Cloud Service Provider (CSP) may determine whether, which, how, and when to deploy impactless updates based on its security posture, risk tolerance, and operational practices. Silicon vendors likewise retain discretion over the availability, scope, prioritization, and timing of such updates,

guided by customer impact, security severity, and engineering readiness. Impactless updates are complementary to impactful updates and do not eliminate the need for periodic impactful baselines; implementations should support forward progress from the last impactful baseline regardless of whether intervening impactless updates were applied, subject to documented prerequisites and dependencies.

1.1 Definitions

Impactless refers to an update operation that does not disrupt the proper function of "host software" (aka, the user's operational view of the platform) because it does not reboot the platform. And, while not a strict requirement, impactless updates should typically apply within approximately 100 milliseconds to meet this definition. This timing window applies only to the update execution itself and excludes activities such as communicating the update to the platform or returning status, which shall be completed without impact to the host software. Impactless updates are ephemeral and only last until the next reboot unless the non-volatile firmware storage is reloaded or updated.

Silicon Vendor Firmware (SVF) is firmware provided by the vendor of the physical CPU SoC. This firmware is (not limited to but may include) critical CPU functionality such as security/root-of-trust operations, power and thermal management, and debug/recovery operation when the CPU data plane has failed. SVF does not execute on the primary, user-accessible CPU cores. Therefore, BIOS and/or UEFI implementations are not SVF.

Microcode (uCode) is firmware that resides in the CPU and controls its operations. It manages the low-level functions of the CPU by translating higher-level machine instructions into detailed operations that the CPU executes. Given its critical nature, microcode updates are particularly important in maintaining security and performance.

Deployment Agent (DA) is software that runs, typically, on a System Management Network and may have localized components on a given platform in order to perform In-Band updates. For Out-Of-Band updates, the DA would typically know how to talk to and command a BMC or other node controller to execute commands, in this case, (for example) Impactless Updates or other SVF maintenance commands.

Impactless Update Functionality (IUF) is a method to support gaining version information about FW running on a given platform and the ability to update the running FW in an impactless manner.

1.2 Problem Statement

A primary issue with traditional host firmware updates lies in the fact that most of the code pertains to the boot time operations, with the remainder associated with runtime or Management Mode (MM) functions that can be updated without causing disruptions. There often are several other dedicated firmware running in the host SoC which need to be updated as well. This imbalance means that updates often necessitate system reboots, resulting in downtime and operational challenges, particularly in environments where maintaining continuous uptime is crucial.

1.2.1 Pain Point 1: Downtime & Freeing Customer Loads

Traditional impactful firmware updates often require one or more system reboots, and in some cases, multiple reboots may be needed to complete a single update sequence. Each reboot extends the maintenance window and compounds downtime, disrupting customer workloads and critical services, which can lead to financial losses and operational inefficiencies. In hyperscale environments, where uptime is paramount, even short interruptions can have significant repercussions.

To mitigate this, CSPs may maintain a pool of reserve machines that can be updated first, allowing workloads to shift until the entire fleet is updated. However, this is a policy decision and does not eliminate downtime entirely. In multi-tenant scenarios, updates can be further delayed if a tenant cannot tolerate any interruption to critical workloads, making scheduling even more complex.

1.2.2 Pain Point 2: Performance Impact

Microcode and silicon-vendor FW updates can be resource-intensive, requiring core utilization if done via Management Mode (MM) or other host-driven methods. This often results in the reallocation of resources from critical tasks, compromising performance and ability to maintain data center efficiency for short durations of time.

1.2.3 Pain Point 3: Coordination Complexities & Time constraints

Coordinating firmware updates across a multi-supplier CPU fleet can be complex and time-consuming. Different suppliers may have varying update schedules and processes, leading to coordination challenges. Additionally, time constraints can further complicate the update process, making it difficult to ensure timely and efficient updates.

1.2.4 Pain Point 4: Delayed application of critical security fixes

Since impactful firmware updates require downtime and/or maintenance windows, organizations often postpone implementation until the next scheduled update cycle. This delay can be particularly risky when critical fixes, especially those addressing severe security vulnerabilities, are needed urgently. The inability to apply such updates immediately exposes businesses to ongoing threats, increasing the window of vulnerability for attacks, data breaches, and compliance violations. In the high-stakes environments of CSPs, carrying the risk of known, unpatched security issues can have dire consequences. The necessity to balance minimizing downtime with responding rapidly to emerging threats creates a persistent tension, leaving organizations unable to act decisively in the face of urgent risks.

1.2.5 Pain Point 5: Architecture specific differences

While the pain points and use cases above apply equally to x86, ARM, and other architectures, fundamental differences introduce additional challenges to achieving uniform impactless update interfaces. These include variations in security protocols, firmware protections, and other architectural features.

SoC architectures are inherently diverse. A key pain point is the absence of a defined, uniform interface and a common set of capabilities to support impactless updates across these varied designs. This lack of standardization complicates efforts to build scalable and consistent update solutions.

1.3 Goals

The overarching goal of implementing impactless firmware updates is to eliminate the disruptions and risks associated with traditional updates, thereby enhancing overall system reliability and performance. Impactless

firmware updates seek to address this problem by focusing on the code that can be updated seamlessly, thereby enhancing system reliability, security and minimizing interruptions.

With the above limitations, broadly, impactless updates can be classified into the categories below.

1) Non-Host SoC Controller Firmware
   Firmware that executes within the SoC but outside host-accessible CPU cores, such as embedded controllers and security or management engines. This firmware is delivered and signed by the silicon vendor. Examples include HSP/ASP/ME, PMU/SMU/PMC, DDR training engines, and PHY or I/O controllers.

2) Vendor-Supplied Host Core Firmware
   Firmware or patches that execute on host-accessible CPU cores and are delivered and signed by the silicon vendor. Examples include x86 microcode and Arm CPU patch or errata mechanisms where supported, as well as vendor-guided firmware-first mitigations. These updates are frequently runtime-loadable and therefore often impactless; however, some changes only take effect after a warm or cold reset or may require operating system or hypervisor feature re-evaluation.

3) Integrator-Owned Privileged Host Runtime Firmware
   Firmware executing on host-accessible cores in privileged or management contexts that is visible to and typically editable by the platform integrator. Examples include x86 System Management Mode (SMM) components and UEFI runtime drivers, as well as Arm EL3 secure-monitor or S-EL1 runtime services. Updates in this category are generally impactful because they require persistence to platform flash and reloading on reset; while hot-patching frameworks exist, they are vendor-specific and subject to strict safety and attestation requirements.

1.3.1 Benefits

- **Continuous Availability:** By applying updates impactlessly out-of-band during runtime, systems can remain operational without interruptions to the Host SW (<100ms as a goal, well below host SW driver timeout limits), ensuring constant availability of services.

- **Minimized Disruptions:** Impactless updates ensure that critical operations can continue without the need for scheduled downtimes, maintaining productivity and service quality**.**
- **Enhanced Security:** Prompt application of critical firmware updates during runtime ensures that vulnerabilities are addressed immediately, bolstering system security.
- **Improved Reliability:** Reducing the risk of update-related failures leads to a more stable and reliable cloud environment, ensuring smooth operations.
- **Efficient Resource Utilization:** With the reduction of reboot requirements, data center resources can be redirected towards more strategic and value-adding activities.

1.4 Scope

This specification aims to address the pain points listed in Section 1.2 Problem Statement by proposing standardization for impactless firmware updates of Silicon Vendor Firmware and Microcode.

1.5 Not in Scope

Components that are not the main host SoC running the primary operating system for the chassis, are not in scope for this specification (e.g. BMC, GPU, IPU, DPU, and other coprocessors).

Firmware changes that require impactful activation or persistence are out of scope for this specification. These may be covered in separate Open Compute Project efforts, for example Open Platform Firmware (OPF).

Furthermore, runtime host firmware/ BIOS including MM is not in scope for this initial specification but may be included in future revisions.

This specification defines impactless firmware updates for hyperscaler-supported platforms. Its current scope is limited to critical security patches and fixes for suboptimal firmware behavior, ensuring updates occur without disrupting the running operating system. Scenarios that require OS involvement—such as OS indications or handshakes which could be delivered IB or OOB—are out of scope for this version and will be addressed in future revisions. The group maintains flexibility to support diverse use cases and will document any gaps or deviations from existing standards, with the objective of guiding future protocol enhancements.

2 Impactless Firmware Update Requirements

2.1 Concepts

Non-host firmware can be thought of as being updated in one of three ways.

1. The first way is to replace the entirety of firmware running on a given microprocessor. Think of this as storing essential state, then replacing the entirety of running code in a microprocessor's SRAM and then restoring the state.
2. A second method would be a derivative of the first. This would replace firmware on a driver-by-driver (or function-by-function) basis.
3. A third method would be to update the executing firmware "in-place" by replacing specific bytes of the code or data (think "peek" and "poke" operations) during runtime. This is not without increased risk over the previous methods but is a possibility.

2.2 Impactless Update Requirements

1. The applied firmware is non-persistent (does not survive SoC reboot).
    1.1. Firmware may be made persistent by storing in a persistent memory. In that case, the update may become persistent. Persistent updates are likely to have a different form-factor than impactless updates (e.g. SPI NOR Flash, eSPI Flash devices).
2. Impactless updates shall participate with system attestation.
    2.1. FW ID, Silicon Configuration Level, Impactless Update Level, AND measurement of the update being applied must be reported to system-attestation.
    2.2. Journey Reporting provides the ability to trace the history of updates.
3. The IUF shall prevent SVN rollback. Furthermore, it can optionally prevent Impactless Update Level, System Update Level and FW Version from being lower than what is currently running. This optionality is at the discretion of silicon vendors to support and CSPs to implement.
4. The Deployment Agent (DA) will provide the following:
    4.1. DA may maintain a Patch Level for each updatable firmware component.
    4.2. DA must enforce a minimum required version threshold before allowing an update to proceed. For example, to install version 10, the system must already be running version 6 or higher.

4.3. The Impactless Update function (IUF) must be able to return to the user a list of current Silicon vendor FW version information of the running system. It shall include for each component current FW version, Impactless Update Level, and optionally for non-FW updates - such as registers in a common data fabric - a System Update Level.

4.4. An increasing firmware version system level or impactless update level may revert a previous update in order to achieve desired functionality. Independent of this capability, a vendor can request that impactless update version must increase monotonically.

4.5. The SoC's Root of Trust (RoT) will ensure it can operate on the update package in such a way that the update package cannot be modified while being verified or applied. This may be accomplished by removing access from non-trusted entities (e.g. Host SW) to the memory buffer in which the update package resides or moving the update package to private memory to which only the RoT has access.

4.6. The IUF shall verify all versioning and verification requirements prior to attempting to apply any update. The IUF shall report any failures to the DA. This includes customer co-signing, if desired.

4.7. Impactless Firmware Update functionality is optional and must be opted-in to by a given platform firmware. Such configuration must be completed and locked prior to handing off to the host OS.

4.8. Customer co-signing of an update package is also optional and, if desired, must be provisioned in a platform-specific manner. For example, prior to host OS initialization.

4.9. The IUF must be able to accept a customer co-signing key token at early BIOS boot (pre-MM lockdown) for use in verifying update packages. It is recommended that this code exist within the confines of a silicon vendor's "Secure Boot" or within any extension of that into platform BIOS.

4.10. The IUF must be able to provide a hash of the customer co-signing token during early BIOS boot for use in ensuring the correct key is loaded for this platform.

4.11. The IUF may provide FW version information prior to boot.

4.12. Mechanisms to provide customer co-signing tokens include, but are not limited to:

4.12.1. Hard-coding such a token in the platform BIOS

4.12.2. Placing the token in a non-volatile storage, such as SPIROM, in a known location. This is not recommended unless the platform BIOS has a mechanism to

ensure no Time of Creation Time of Use (TOCTOU) errors occur using Platform Secure Boot mechanisms (SVF supplied and UEFI Secure Boot compatible)

4.12.3. Establishing a protocol with an external BMC or node controller that allows the platform to identify a given SoC/CPU and request a key token from a customer-specific key store

3 Impactless Firmware Update Model

This workflow is scoped to SVF and uCode payload updates. Additional payloads will be discussed in future revisions.

3.1 General Flow

The process starts with the system management network (Rack Manager, or other technology) that passes the update to the SoC (typically via its BMC or node controller) for the SoC to consume, verify and apply the update.

The following steps are a suggested implementation of impactless updates.

(1) Impactless updates will be delivered either In-Band (IB) or Out-of-Band (OOB), but not both. Consumers will have the flexibility to configure the delivery method based on their needs.
   a. Implement a boot-configuration option that can be consumed by the silicon to control whether updates are exclusively IB or exclusively OOB.
   b. Once selected, this option is not changeable until the next reboot.
   c. Willingness to support both IB and OOB simultaneously within a boot cycle is optional and a Silicon Vendor choice.

(2) Implement a notification to inform the Operating System and Control Plane that the update has occurred (OOB/IB).
   a. This notification can be ignored by the Operating System or Control Plane.

(3) This section provides an example of an OEM-specific implementation for impactless OOB firmware updates using MCTP/PLDM communication between BMC and SoC. Another example, based on the DSP0267 (2023-12-13, Version 1.3.0) PLDM-defined flow, will be considered for integration in a future release.
   a. It is recommended that the OEM-Specific PLDM Type be used as this operation does not fit the currently defined PLDM method of FW update and each silicon vendor may have their own unique implementation. Once (and if) this operation fits within PLDM method, it will be recommended to transition to the standard PLDM Type (PLDM DSP0267).
   b. This is the format of the Impactless Update (IU) command/status header. The update package may be a 4K page at a 2M aligned address, or a variable amount of memory depending on the size of the IU Update package. Note: Each vendor may have specific

requirements for update package memory alignment and location. Refer to the respective SoC vendor documentation for details.

Byte Offset	Bits	Dir	Name	Description
00h	31:0	In	TotalSize	Total Size of EXT_CMD_BUFFER (including this header) in bytes
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 the Silicon Vendor if Status != Success
10h	31:0	Out	CAPABILITIES	- Bit 0 set to 1 indicates SFS is enabled. Otherwise SFS is disabled and no other bit is meaningful. - Bit 1 set to 1 means Customer Co-Signing is required. - Bit 2 set to 1 means OOB is selected, otherwise IB is selected.
14h – fffh	31:0	-	Reserved	

c. **Command 1: Get Firmware version information:** This command retrieves the current firmware version, ensuring that the system is aware of the existing firmware state before proceeding with the update.

   i. The issuance of the command is simply a PLDM command with OEM-Specific Type and command ID of 1.

   ii. **Suggested implementation:** The return data consists of command status and extended status (as a 4KB block) plus the FW Version information as follows (also a 4KB block). This is shown below, with additional detail below that.

IU Cmd/Status Buffer – 4kb in size
Returned Version information – 4kb

Offset	Bits	Name	Description
00h	31:0	API_VERSION	The version of the API that this command supports

04h	31:0	IMPACTLESS_UC	The current Impactless Update Count as an integer (IUC) as a monotonically increasing value from update to update
08h	31:0	STUC	The SOC Tuning Update Count for updates that don't change firmware version numbering
0Ch	31:0	NUM_SUPPORTED_FWS	Number of FW Versions captured out of MAX (MAX = 255)
10h-E1Fh	31:0	FW_VERSION_INFO	See below
E20h-FCFh	31:0	Reserved	432 bytes, must-be-zero (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)

Each FW_VERSION_INFO entry is a triplet consisting of:

Offset	Bits	Name	Description
00h	31:0	VERSION_TYPE	The enum of the FW type (per silicon vendor)
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

d. **Command 2:** Perform an update. This command initiates the firmware update process, applying the new firmware to the system.

IU Cmd/Status Buffer – 4kb in size
Optional IU Customer Cosigning Header – 256 bytes
SVF Signing Header – 256 bytes
SVF Update Package (variable size)
SVF Signature
Optional IU Customer Signature

e. **Command 3:** Request Impactless Update Capabilites. This command gets the following information:
    i. Bit 0 – set to 1 if Impactless Updates are enabled. Set to 0 if Impactless Updates are disabled. If disabled, no other capability bit is meaningful.
    ii. Bit 1 – set to 1 if Customer Co-Signing is required. Set to 0 if Customer Co-Signing is not required.
    iii. Bit 2 – set to 1 means OOB IU is enabled. Set to 0 means IB IU is enabled.

f. Returned Status Block (Note: this is also the command/status header block)

Byte Offset	Bits	Dir	Name	Description
00h	31:0	In	TotalSize	Total Size of EXT_CMD_BUFFER (including this header) in bytes
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 the Silicon Vendor of Status != Success)
10h	31:0	Out	CAPABILITIES	- Bit 0 set to 1 indicates SFS is enabled. Otherwise SFS is disabled and no other bit is meaningful. - Bit 1 set to 1 means Customer Co-Signing is required. - Bit 2 set to 1 means OOB is selected, otherwise IB is selected.
14h – 1fffh	31:0	-	Reserved	

3.2 Impactless Update Status Codes

Status codes 0x00-0x7f are errors that are generic and publicly available. Status codes 0x80-0xff are reserved for Vendor-Specific error codes and will vary depending on the SOC vendor.

Status Code	Name	Description
00h	IU_SUCCESS	Success – patch applied
01h	IU_FEATURE_NOT_PRESENT	An attempt was made to update something that is not enabled
02h	IU_INVALID_TOTAL_SIZE	Invalid TotalSize
03h	Reserved	
04h	IU_INVALID_PKG_SIZE	Invalid Image size
05h	IU_DISABLED	Patching not allowed
06h	IU_INVALID_CUSTOMER_SIGNATURE	Invalid Customer Signature
07h	IU_INVALID_VENDOR_SIGNATURE	Invalid Vendor Signature
08h	IU_INTERNAL_ERROR	Please report the extended status to the SOC Vendor
09h	IU_CUSTOMER_SIGNING_ NOT_ALLOWED	Customer signed but not allowed
0ah	IU_INVALID_BASE_PATCH_LEVEL	Invalid base patch level – Base FW Version mismatch
0bh	IU_INVALID_CURRENT_PATCH_LEVEL	Invalid current IU patch level – current patch level mismatch
0ch	IU_INVALID_NEW_PATCH_LEVEL	Invalid new IU patch level – less than current patch level
0dh	IU_INVALID_SUBCOMMAND	Invalid IU subcommand
0eh	IU_PROTECTION_FAIL	Payload cannot be protected - possibly not aligned on 2MB boundary.
0fh	IU_BUSY	Busy – IU cannot update with this package
10h	IU_FW_VERSION_MISMATCH	The uploaded FW is less than the existing FW.
11h	IU_SYSTEM_VERSION_MISMATCH	The current SYS patch level and new SYS patch level are not one apart.

3.3 Development Model

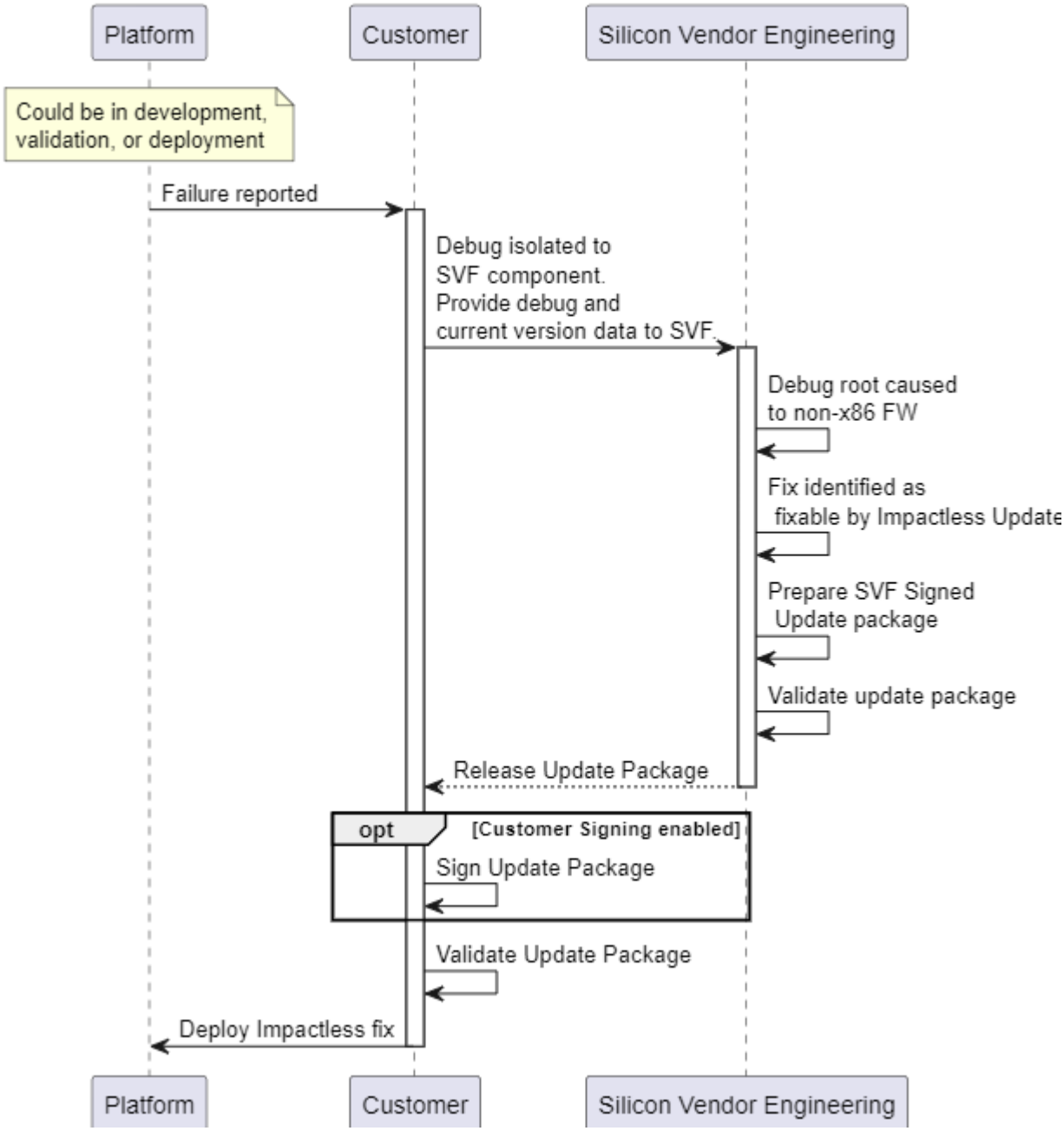

Figure 1: Impactless Firmware Update Development Model

3.4 Notifications Mechanisms

Impactless updates shall include a mechanism to notify the Control Plane that an update has been completed and if further actions need to be taken. This covers both microcode and vendor-specific firmware. To provide context-dependent information, if an update is initiated in-band (IB), the OS is notified organically by the return of status from the command to perform the update. Similarly, for out-of-band (OOB) updates, the OS is notified through the respective OOB mechanisms. The intention is to ensure that notifications are possible between IB and OOB mechanisms, allowing for streamlined communication and updates across different channels.

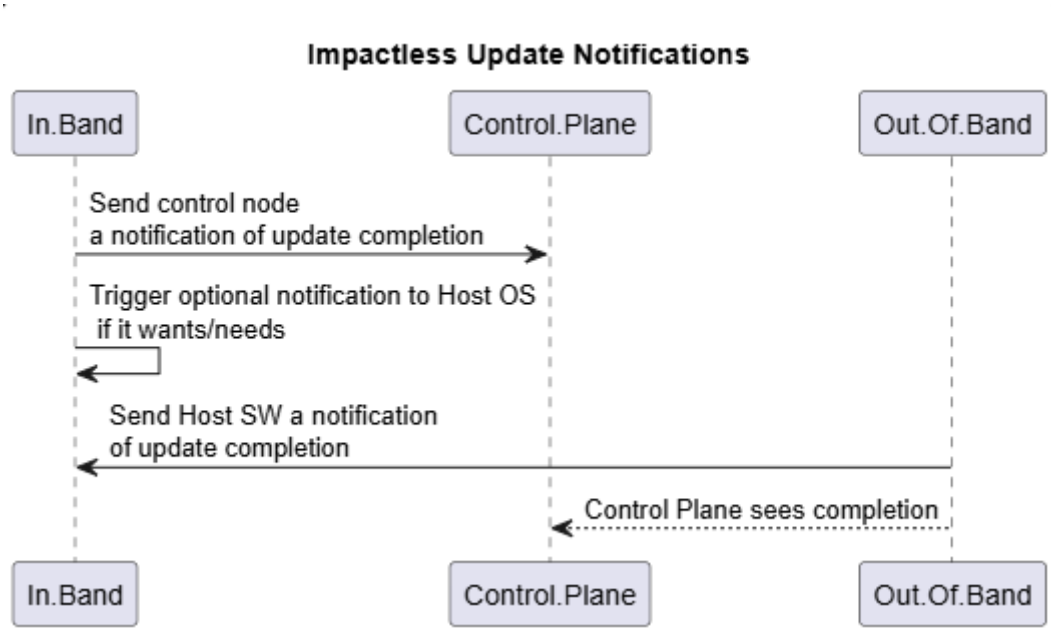

Figure 2: Impactless Update Notifications

3.5 In-Band (IB) Deployment

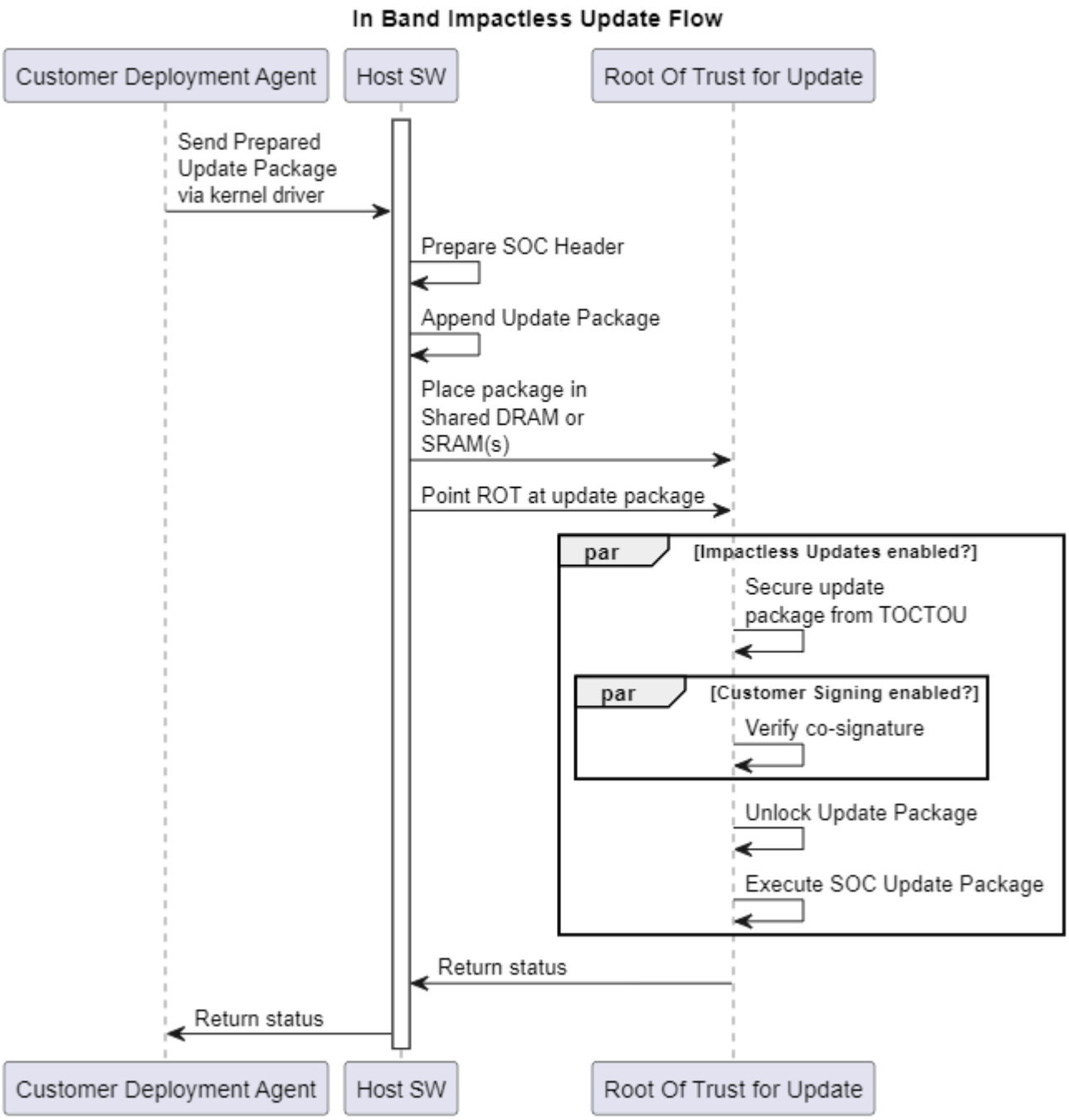

Figure 3: In Band Impactless Update Flow

3.6 Out-Of-Band (OOB) deployment

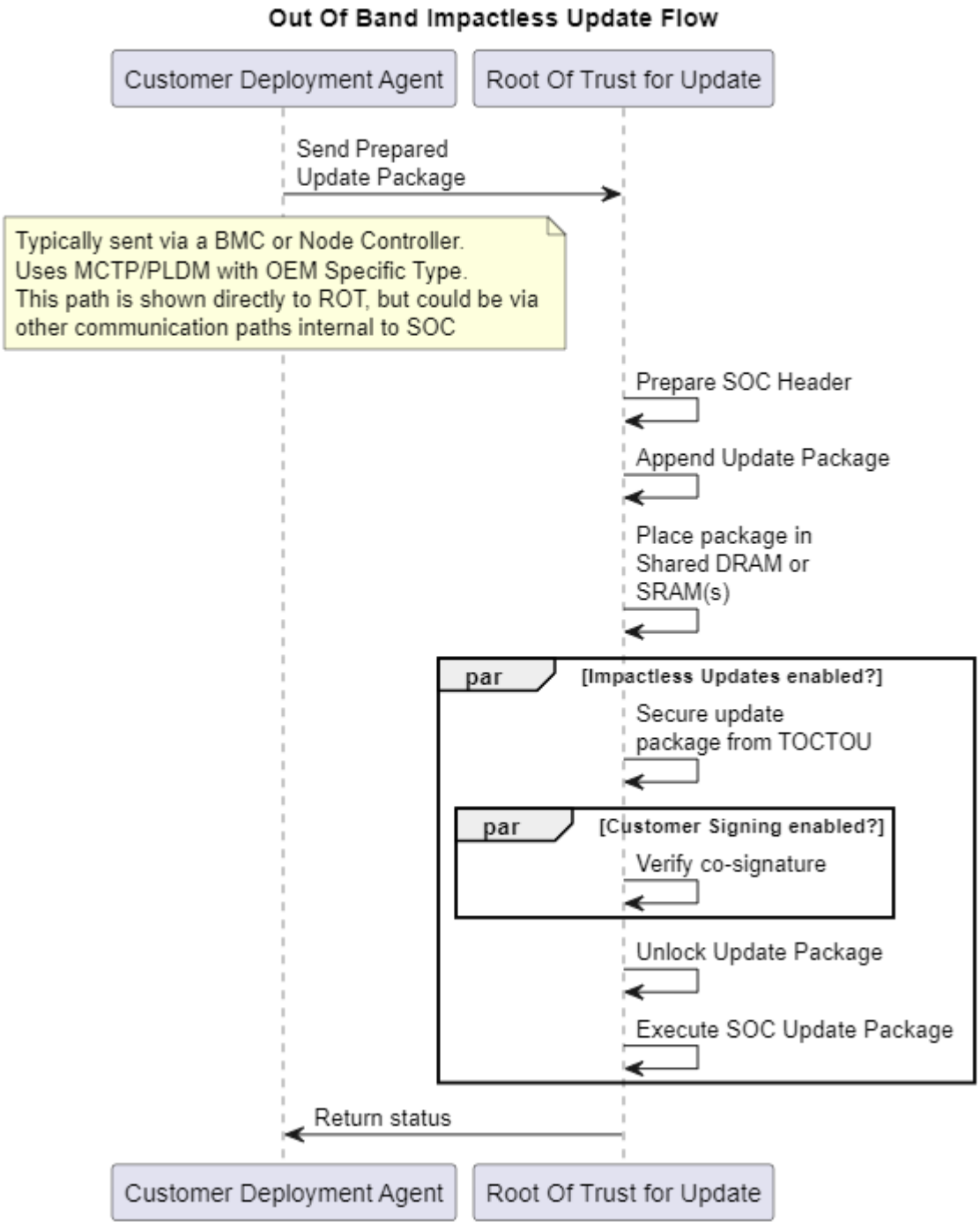

Figure 4: Out of Band Impactless Update Flow

3.7 Package Format

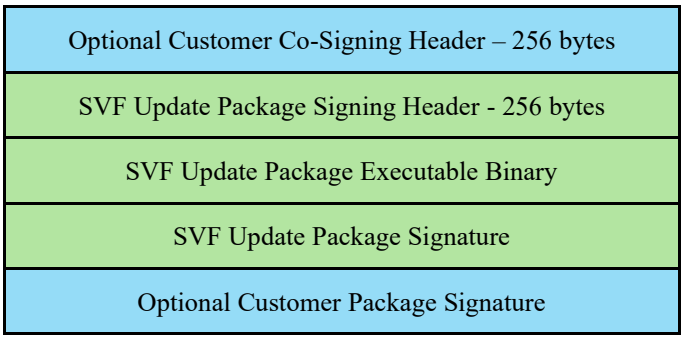

Figure 5: Package Format

3.7.1 Optional Customer Co-signing

There shall exist a method for optional customer co-signing of an update package (or indicating a customer does not wish to utilize co-signing). This allows impactless update users to co-sign the SVF supplied update package providing a layer of extra protection by utilizing an RSA keypair to further protect the update package. It also allows users to enable their test lab with one signature and their data center deployment with another and make sure no test update package is inadvertently pushed into deployed systems. Further, users can distinguish between platform types, if they have multiple, by using different customer signing on update packages intended for the varying platforms.

The customer public key token, used by the SOC, shall be able to be sent to the SOC in the platform boot phase when the machine is securely operating (for example AMD Platform Secure Boot).

3.7.2 RSA_TOKEN_FORMAT

Customers define the OemCoSignKey using the RSA_TOKEN_FORMAT.

To co-sign an update package received from an SVF, the customer generates an RSA key pair (either 2048-bit or 4096-bit – 4096 bit recommended) and signs the package with private key using RSA-PSS signature scheme (RSASSA_PKCS1_PSS_MGF1_SHA256 for a 2048-bit key (total size: 576 bytes), or 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

3.7.3 Examining the RSA Token

If customer co-signing is enabled, each SVF must provide a command for BIOS to get the SHA-384 hash of the provided customer co-signing token so the platform BIOS (pre-MM lockdown) can determine what key has been sent to the SOC.

Future versions of the spec will cover unsuccessful updates, including the messaging of failures and the recovery process. Given the anti-rollback set-up, a new firmware version would be needed (would be able to include previous firmware content to bring the system back to pre-update state).

3.7.4 Cryptographic Agility

This section will be further refined for v1.0 to ensure cryptographic agility (i.e. ECDSA and ML-DSA).

4 Microcode-Specific Details

This section covers microcode-specific requirements applying to the Impactless firmware update model. Requirements and flows align with Section 3 Impactless Firmware Update Model.

4.1 Microcode Update Categories

Categories for microcode impactless updates:

1. In-Band phase: Operating System driven.
2. Out-of-band phase: Firmware driven.

4.2 Microcode-specific Impactless Update Requirements

4.2.1 In-Band: Loading uCode from Operating system to avoid reset

This section describes the existing process used to load or reload microcode (uCode) in an in-band manner, leveraging a software patch load orchestrator to coordinate the update across all logical processors.

1. **Delivery of uCode Blob:**
   A microcode (uCode) blob is delivered to a software delivery agent operating within a running OS or Hypervisor.
2. **Staging:**
   The microcode blob is staged via a staging interface. The staging phase ensures that the FW updates will happen within the pre-established 100 ms time requirement.
3. **Quiescing Logical Processors:**
   A software patch load orchestrator - typically implemented as a software delivery agent running as a single management thread - quiesces all logical processors in the system, bringing them to a minimal operating state to prepare for the update.
4. **Triggering the Update:**
   The orchestrator performs the necessary MSR writes to initiate the uCode update across all logical processors within the platform.
5. **Hardware Verification and Update:**
   Hardware verifies and applies the uCode update to each core.

6. **Restoring Normal Operation:**
   The orchestrator returns all cores to their normal operating mode, resuming standard system operations.
7. **Operational Efficiency:**
   This operation is typically very fast but does require the OS (via the orchestrator) to coordinate that all cores are properly quiesced before the update.

4.2.2 Out Of Band uCode updates

The same requirements discussed in Section 3 apply for microcode. This section calls out additional requirements specific to uCode.

1. It is conceptually possible to have a silicon vendor's internal processor quiesce all cores and issue internal instruction(s) to cause the cores to update their microcode and then release the cores. Driven by HW/FW, this would be independent of OS state or activity.
2. While the system may experience a small performance impact, it would be shorter than the existing OS-driven method.
3. As envisioned, benefits of OOB uCode update are expected to include no inter-process coordination, minimal performance impact, Simplified Redundancy checks, and Compatibility across different OS types
4. For OOB uCode updates, a method to (optionally) notify the host SW that an update has occurred is defined as an ACPI General Purpose Exception. No OS would have to consume it, but it would be available to those that want/need to know when an update is applied. Indeed, this would also be true for any Impactless update.
5. As referred to in section 3, the process starts from the system management network (Rack Manager, or other technology) that passes the update to the SoC (typically via its BMC or node controller) for the SoC to consume, verify and apply the update.
6. Out of band updates provide end users running on Bare-Metal clients, the ability to delegate uCode updates to the CSP.

5 Runtime BIOS & Management Mode (MM) Firmware Impactless Updates

This section will be covered in future spec revisions.

1. Management Mode (MM)
    a. MM handlers to update impactlessly.
    b. Mechanisms to reload the MM driver.
2. Runtime BIOS (Boot Time FW)
    a. Address Boot time initialization issues, such as incorrect PCIe completion timeout values.

6 Compliance tools

The OCP Test & Validation (OCPTV) initiative is a collaboration between datacenter hyperscalers with the goal of standardizing aspects of the hardware validation/diagnosis space, along with providing necessary tooling [4] to enable both diagnostic developers and executors to leverage these interfaces. In support of this, CPACT (Cloud-Processor-Accessibility-Compliance-Tool) [5] is being developed, to allow hyperscalers and suppliers to check for CPU compliance with the specifications. The tool will be a combination of compliance scenarios, categorized by domains, and the underlying framework. Impactless Update requirements will be a domain by itself. The compliance scenarios for each domain will be dictated by the respective specifications.

The compliance tool will support a range of access methods to the target, inband and out-of-band. It will also provide a standardized way to build the results for all the scenarios run on the target. All logs collected will be available for detailed analysis. This tool bridges the hyperscaler and the suppliers, assisting in the onboarding of supplier diagnostic capabilities with confidence. Once the specifications are public, the tool will become available publicly on GitHub. The tool will leverage OCPTV's philosophy and underlying *pytest* based framework to create each of the different compliance scenarios.

7 References

1. OCP GPU & Accelerator Management
2. FW Update Specification.
3. OCP Test & Validation Project
4. OCP Cloud Processor Accessibility Compliance Tool (CPACT) GitHub
5. OCP Test & Validation GitHub
6. OCP Hyperscale CPU RAS and Debug Requirements V0.5
7. OCP Hyperscale CPU Impactless Firmware Updates Requirements V0.5
8. OCP GPU & Accelerator Management Interfaces Version 1.0
9. Specification: Firmware Update Requirements for GPU Revision 1.0
10. Unified Extensible Firmware Interface Specification, version 2.9,

8 OCP Tenets

Openness

This specification was developed collaboratively among industry partners and competitors, and is open to all OCP members.

Efficiency

The goal is streamlined CPU integration into hyperscaler solutions, with compliance tools ensuring high-quality products.

Impact

As one of the first initiatives to standardize CPU Impactless Firmware Updates, it will significantly improve quality and TTM for CPU Systems by hyperscalers.

Scale

Applicable to large-scale CPU deployments in hyperscale Data Centers.

Sustainability

Improved ability to perform updates will reduce the component replacement rates and node idle states.

9 Appendices

9.1 Appendix A - Glossary of Terms

Term	Definition
BMC	Baseboard Management Controller
CPU	Central Processing Unit.
CSP	Cloud Service Provider - The term Cloud Service Provider is intended to be a broad term for any organization that runs a large fleet of machines and provides services using that fleet. The use cases in this document are broad and are not strictly limited to organizations that provide cloud services. This proposal also supports use cases that include server operators that are not cloud providers.
FW ID	Vendor-specific identifier that distinguishes each firmware component within the SoC (does not refer to firmware versions).
Hardware component	A component designed by a supplier. Examples of hardware components include CPU chips/SoCs, DRAMs and PCIe devices.
Hyperscalers	Cloud service providers of the world.
Impactless Update Level	Identifies the sequential ID of an update package.
IU	Impactless Update
Node	A platform designed by a supplier integrating multiple components that run an operating system. These are sometimes also called compute nodes.
Platform firmware	Firmware running on mission-mode cores on the processor. On x86 CPUs, platform firmware is sometimes called BIOS or UEFI firmware.
OS	Operating System - that runs on the processor. The operating system may run in a virtual machine on a hypervisor or directly on the machine (bare metal).

Reset	A reset is an operation that reinitializes hardware or firmware to a known state. For this specification, a reset is considered base-level if it reloads foundational firmware (e.g., platform or boot firmware). Base-level resets include cold resets (full power cycle) and warm resets (reinitialization without full power removal). Functional or peripheral resets that do not reload base-level firmware are out of scope unless stated otherwise.
RM	Rack Manager
RoT	Root of Trust
Silicon Configuration Level	An identifier used to indicate a change in the state or configuration of the SoC without updating a specific firmware version, typically reflecting changes such as register updates or run-once functionality.
SLA	Service Level Agreement
SLO	Service Level Objective
SVN	Security Version Number
TOCTOU	Time of Creation Time of Use
TTM	Time to Market.
UEFI	Unified Extensible Firmware Interface, defined in www.uefi.org.