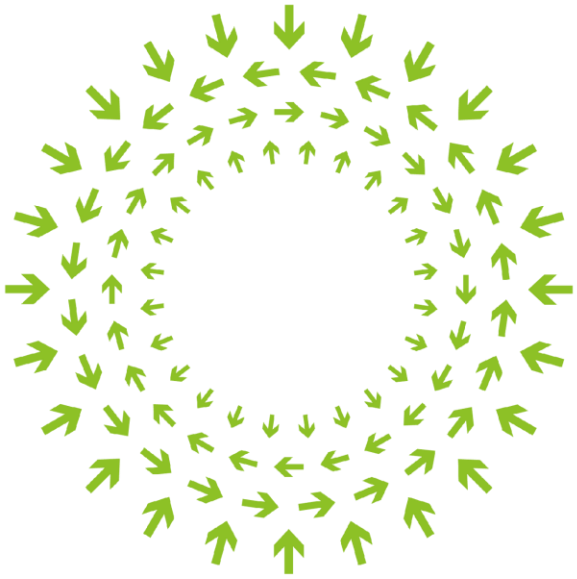


OPEN

Compute Project

HYPERSCALE CPU IMPACTLESS FIRMWARE UPDATES REQUIREMENTS

Specification version 0.5

Effective 5/1/2025

Authors:

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

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

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

Contact: aakesson@microsoft.com

Table of Contents

Table of Contents 2

Table of Figures 3

Executive Summary 4

License 5

About Open Compute Foundation 6

1 Introduction 7

1.1 Definitions 8

1.2 Problem Statement 8

1.3 Goals 10

1.4 Scope 11

1.5 Not in Scope 11

2 Impactless Firmware Update Requirements 12

2.1 Concepts 12

2.2 Impactless Update Requirements 12

3 Impactless Firmware Update Model 15

3.1 General Flow 15

3.2 Development Model 19

3.3 Notifications Mechanisms 20

3.4 In-Band (IB) Deployment 21

3.5 Out-Of-Band (OOB) deployment 22

3.6 Package Format 23

4 Microcode-Specific Details 25

4.1 Microcode Update Categories 25

4.2 Microcode-specific Impactless Update Requirements 25

5 Runtime BIOS & Management Mode (MM) Firmware Impactless Updates....26
6 Compliance tools....27
7 References....27
8 OCP Tenets....28
9 Appendices....29
9.1 Appendix A - Glossary of Terms....29

Table of Figures

Figure 1: Impactless Firmware Update Development Model....19
Figure 2: Impactless Update Notifications....20
Figure 3: In Band Impactless Update Flow....21
Figure 4: Out of Band Impactless Update Flow....22
Figure 5: Package Format....23

Executive Summary

This specification standardizes 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 all 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.

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.

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:

Microsoft, AMD, Google

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 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

1 Introduction

This specification standardizes runtime non-persistent Impactless 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.

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.

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 are designed to maintain the most recent impactful update. They provide the ability to update security and latest feature updates in the most recent impactful update. However, not all impactless updates will be implemented universally. Each Cloud Service Provider (CSP) has the discretion to decide based on their specific security requirements and operational needs. Similarly, silicon vendors have the flexibility to share updates at their discretion and capability, driven by factors such as customer impact,

security severity, and other considerations. Impactless updates are not meant to replace and avoid ever having to do impactful updates.

1.1 Definitions

Impactless – the use of the term "impactless" is defined as not requiring a system reset to update running firmware and applies to the view the host software has of the update operation. While not a hard definition, this is typically intended to be performed within 100ms. That window includes just the actual time surrounding the update of the firmware. Time to communicate an update to the platform is an example of what is not included as well as the time to return status.

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 the entire SVF supplied capability to support impactless updates. This would include, at a minimum, 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 system reboots, leading to downtime. This downtime can interrupt customer workloads and critical services, causing financial losses and operational inefficiencies. In hyperscale environments, where uptime is crucial, even minimal downtime can have significant repercussions. Data center providers can minimize this by establishing policies for having some number of additional machines held "in reserve" that can be updated and have workloads shifted to them until an entire fleet can be updated. This is, of course, a data center policy decision.

Furthermore, in certain cases, systems can be shared between customers or tenants with different requirements. In this case, the system might not be able to be updated as a customer might not want their work interrupted in case of a critical workload.

1.2.2 Pain Point 2: Performance Impact

In-band uCode 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: Architecture specific differences

While the pain points and use cases above apply equally to both x86, ARM or other architectures, there are specific fundamental architecture differences that add additional challenges to achieving unified impactless update mechanisms. These differences include variations in security protocols, firmware protections, and other architectural features.

For instance, ARM architecture utilizes the Hardware Security Processor (HSP) for authentication and verification, which can offload these tasks from the host. On the other hand, x86 architecture, specifically AMD, has a module on the System on Chip (SoC) that can perform similar offloading functions.

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) Silicon Vendor Firmware (SVF)
2) Microcode (uCode)
3) Host-related Management Mode (MM)

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 elimination 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

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.

2 Impactless Firmware Update Requirements

The following technical requirements are mainly focused on x86 architecture. Future revisions will expand the requirements to be inclusive of ARM architecture.

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, eSPI).
2. Impactless updates shall participate with system attestation.
    2.1. FW ID, System Patch Level, Impactless Update Level, AND measurement of the update being applied must be reported to system-attestation.
3. The Deployment Agent (DA) will provide the following:
    3.1. DA may maintain a Patch Level for each updatable firmware component
    3.2. 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 (for non-FW updates - such as registers in a common data fabric) a System Update Level

3.3. The IUF must prevent Impactless Update Level, System Update Level and FW Version from being lower than what is currently running (anti-rollback).

3.4. 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, Impactless update version must increase monotonically.

3.5. 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.

3.6. The IUF shall verify that a valid SVF header and cryptographic signature are present within the update package. In this case, if the header or signature is not valid or non-existent, the IUF will abort the update process and return an error to the DA.

3.7. IUF shall verify, if provided and provisioned for, that a valid customer platform header and cryptographic signature are present, wrapping the update provided by the Silicon Vendor. If the customer's header or cryptographic signature are not valid, the IUF will abort the update process and return an error to the DA.

3.8. Impactless Firmware Updates are optional and must be opted-in to by a given platform's BIOS.

3.9. Customer co-signing of an update package is also optional and, if desired, must be configured by platform BIOS (opt-in and providing of a public key token to be used for verifying that a given update comes from the platform owner.)

3.10. 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.

3.11. 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.

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

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

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

3.13.2. 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)

3.13.3. 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 BIOS token that can be consumed by the platform BIOS/Firmware to control whether updates are exclusively IB or exclusively OOB.

(2) Install an ACPI GPE mechanism to inform the Operating System that the update has occurred (OOB/IB).

(3) Notification for every update for the IB mechanism in order for Control Plane to be aware that an update has occurred.

(4) Implement MCTP/PLDM communication between BMC and SoC to communicate the impactless OOB update.
   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.
   b. This is the format of the IU command/status header. This is a 4k page at a 2M Aligned address or a variable amount of memory depending on the size of the IU Update package. See below 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 AMD if Status != SFS_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	CURRENT_PATCH_LVL	The current Impactless Update patch level as an integer (IUPL)
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 = 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 AMD if Status != SFS_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 Development Model

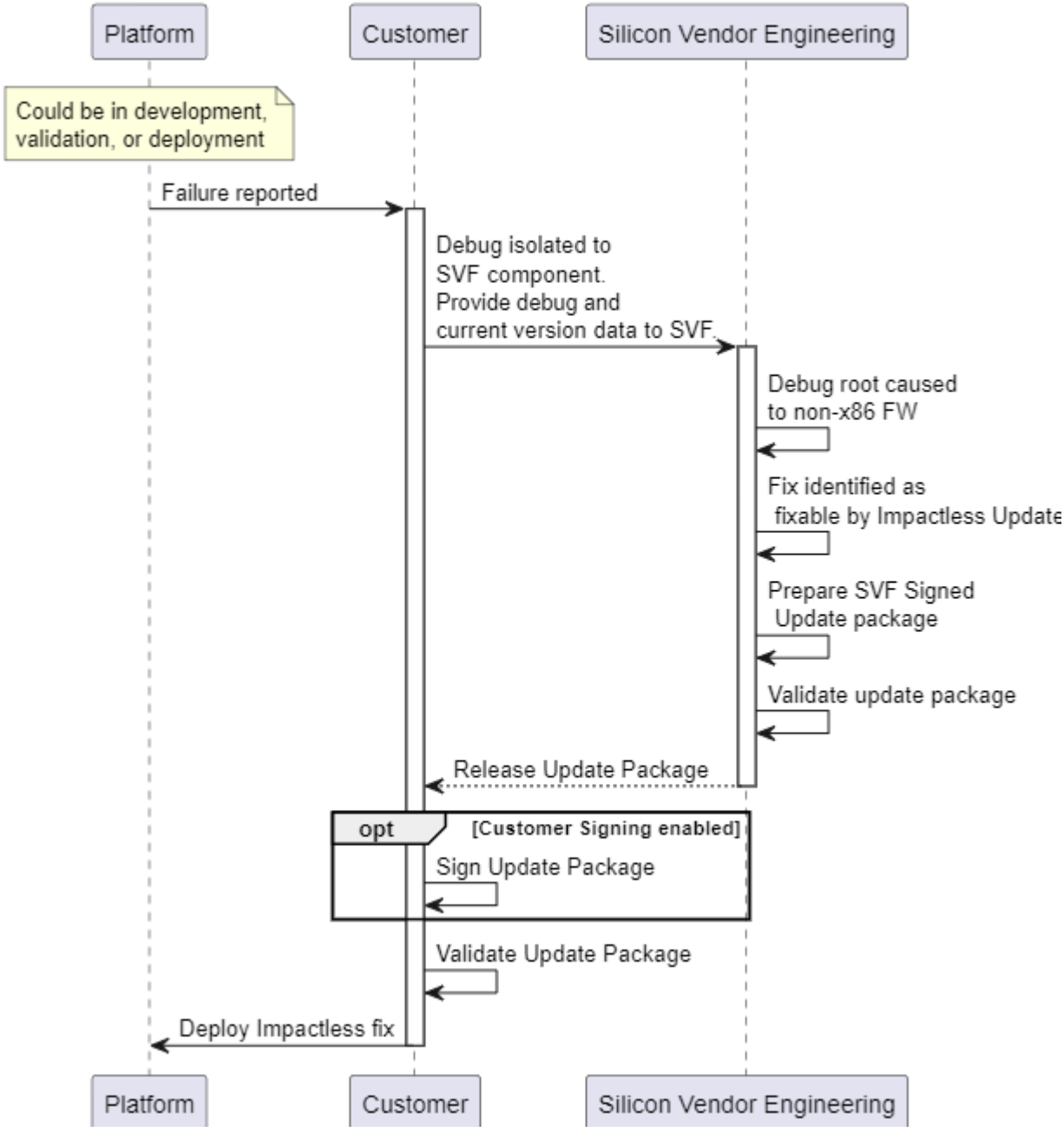


Figure 1: Impactless Firmware Update Development Model

3.3 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.

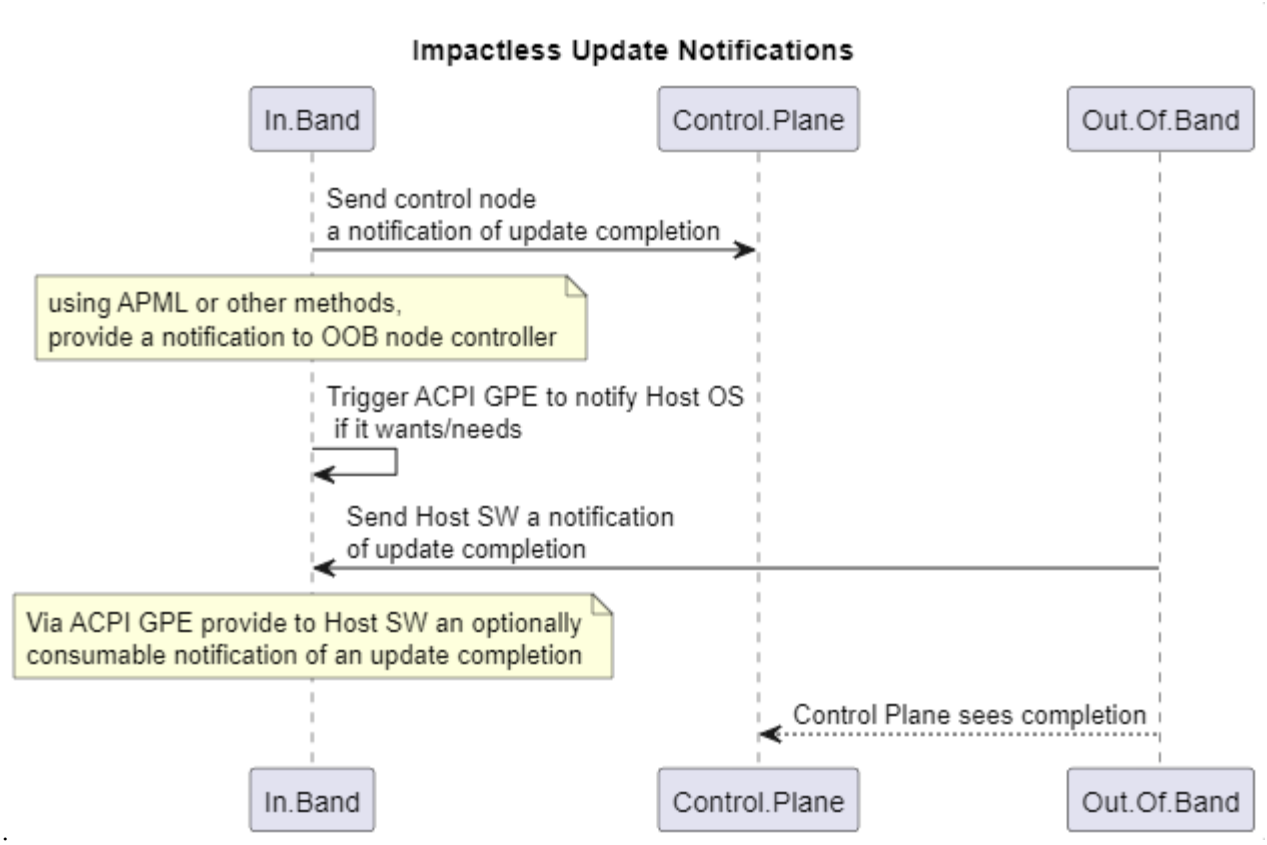


Figure 2: Impactless Update Notifications

3.4 In-Band (IB) Deployment

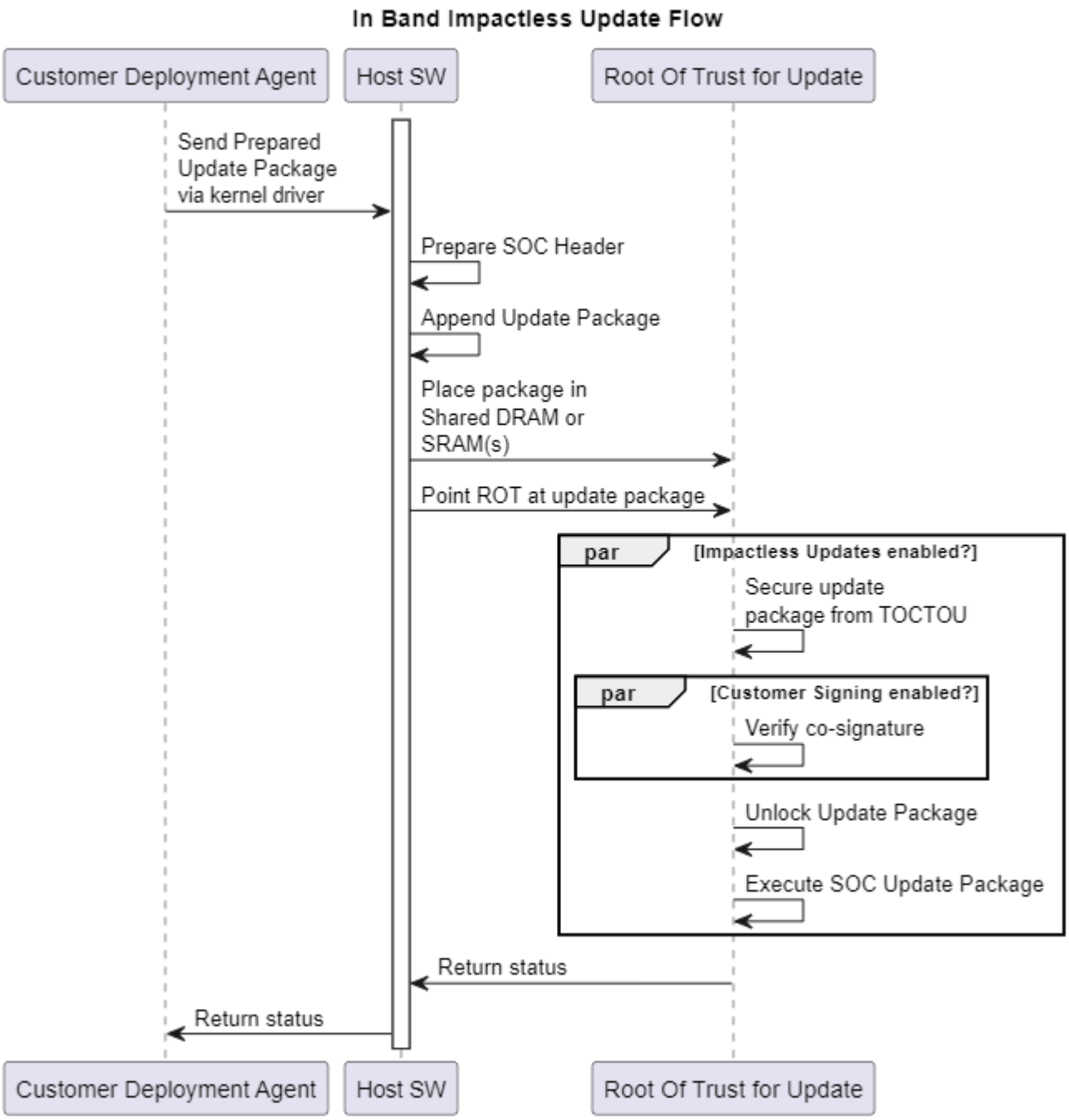


Figure 3: In Band Impactless Update Flow

3.5 Out-Of-Band (OOB) deployment

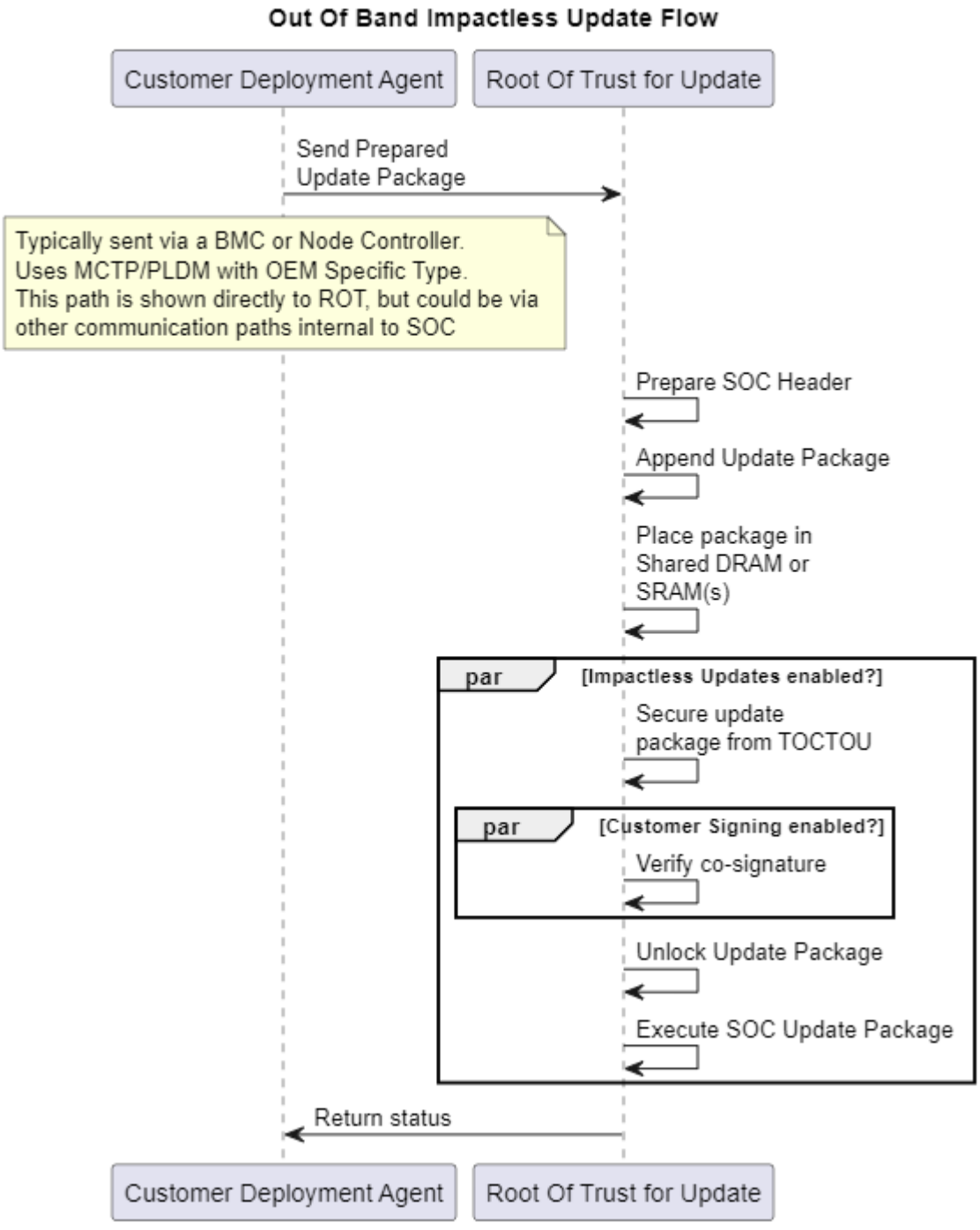


Figure 4: Out of Band Impactless Update Flow

3.6 Package Format

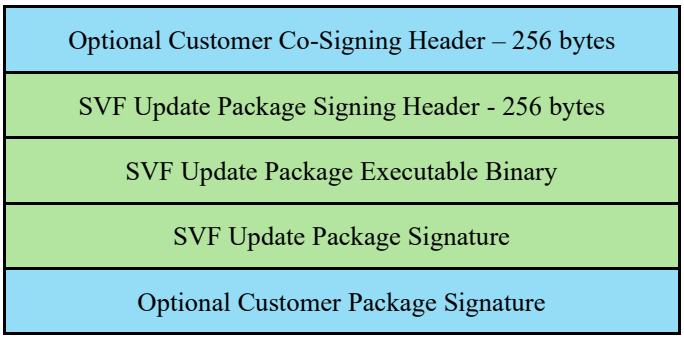


Figure 5: Package Format

3.6.1 Optional Customer Cosigning

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).

3.6.2 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 (can be any sequence of up to 16 bytes)
14h	Reserved	Must be zero
24h	Reserved	Must be zero
28h	Reserved	Must be zero
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

3.6.3 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).

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. Pre-Platform BIOS phase: Microcode update performed before cores up and running.
2. Pre-OS-Boot phase: Microcode updates applied by platform BIOS.
3. In Band phase: Operating System driven.
4. 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 is the existing process used to load/reload uCode.

1. A uCode blob is delivered to a software delivery agent in a running OS or Hyper-Visor (HV)
2. Every Core is quiesced to some minimal operating state
3. Every core writes to an MSR register to point to the uCode update blob
4. Hardware verifies and updates each core's uCode
5. The cores are returned to "normal" operating mode.
6. This operation is very quick but does require the OS to coordinate that all the cores are quiesced.

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 GPU & Accelerator Management Interfaces Version 1.0
5. Specification: Firmware Update Requirements for GPU Revision 1.0
6. Cloud Processor Accessibility Compliance Tool - CPACT
7. 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.
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.
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).
RM	Rack Manager
RoT	Root of Trust
SLA	Service Level Agreement

SLO	Service Level Objective
TOCTOU	Time of Creation Time of Use
TTM	Time to Market.
UEFI	Unified Extensible Firmware Interface, defined in www.uefi.org.