Secure Firmware Recovery

EDITOR: Eric Spada, Broadcom

CONTRIBUTORS:
Wojtek Powiertowski, Facebook, Inc.
Ben Stoltz, Google
Bryan Kelly, Microsoft
Vladimir Dreizin, Broadcom
Edmund Szeto, Broadcom
Yigal Edery, NVIDIA
Varun Sampath, NVIDIA
Danny Ybarra, Western Digital

Revision History

Revision	Date	Guiding Contributor(s)	Description
0.9	04-13-21	Eric Spada, Broadcom	Draft Release
1.0-rc	06-14-22	Eric Spada, Broadcom	Review Release
1.0	09-14-22	Eric Spada, Broadcom	Final Release

Purpose

This document creates guidelines on how to recover a failed or compromised device. The recovery operation provides a mechanism for a recovery agent (RA), in coordination with a PA-ROT (Platform Active RoT), to recover a device's firmware and/or security critical parameters of an AC-ROT (Active Component RoT). The recovery process MUST bring the device to a known security state.

Audience

The audience for this document includes, but is not limited to, system and system component designers, security information and event management (SIEM) system developers, and cloud service providers.

Syntax and conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The roles “attester”, “verifier”, and “reference integrity measurements” are defined in the OCP Attestation Doc

Introduction

Guiding principles for this document are based on NIST SP 800-193 and the three pillars supporting Platform Resiliency:

- Protection – Secure boot/Attestation/Threat Model
- Detection – Attestation Doc
- Recovery – This Doc

This document focuses on the Recovery principle, which is a mechanism for restoring Platform Firmware code and critical data to a state of integrity in the event that firmware code or critical data have been corrupted, the device is unresponsive, or when forced to recover through an authorized mechanism. While focusing on Recovery, the document will also discuss aspects of Detection and Protection where needed.

The agents involved in the recovery process

- Device or AC-ROT: Device which is being recovered

- PA-ROT: System component responsible for determining the health of a device and initiating recovery via the recovery agent.
- Recovery Agent (RA): System component responsible for orchestrating the recovery process.

Relationship to Other OCP Security Documents

The threat model is described in the OCP threat model. When attestation is referred to in this document, it assumes compliance with the OCP Attestation document. The device is expected to conform to the OCP secure boot document. This includes all images including recovery. A common glossary of terms for OCP security is referenced here.

Device Recovery and SP800-193 Alignment

The device shall follow the guidelines defined in the NIST SP800-193 specification, for its firmware protection, tamper detection and recovery capabilities.

A device may or may not be able to persist critical data through a recovery process. A Device that is unable to maintain critical data MUST go through a process to reestablish this data or be returned to manufacture.

The Recovery process is a critical process for the overall security of the platform since the PA-RoT. This document focuses on device recovery, specifically of the AC-ROT which is a symbiont device to the PA-RoT.

Device Recovery's goal is return a Device to its normal operation state running the correct firmware as verified by Device Attestation.

Detection, remediation, corruption and initiating recovery action

The primary mechanism of detecting the boot state of symbiont devices by PA-ROT shall leverage reporting and attestation capabilities of devices primarily based on the OCP Attestation specification.

If a device does not pass attestation, then remediation must occur. If the device is functional and has trusted images (healthy firmware and can respond to attestation), the firmware should be updated via standard means. An example of firmware update procedures are described in DMTF PLDM (Platform Level Data Model) Firmware update specification. If the device is sufficiently out of date, a PA-ROT can choose to recover the device and to return it to a consistent state.

This document further proposes a protocol to create a standard recovery protocol across managed symbiont devices. In case of critical failure resulting in the symbiont device not being able to communicate over a high-level protocol, the device and PA-RoT shall fallback to recovery via SMBus.

Types of Failures

Firmware recovery may be required for various reasons, the critical one being a corrupted firmware image leading to boot failure. The attestation capabilities defined in the OCP Attestation document provide the mechanisms to help identify issues with later stages of device boot and configuration such as non-compliance of either firmware version or configuration, but do not directly help reporting critical boot failures. Two type of remediation are defined below:

- Update: Device is in a functional state, but has old firmware or configuration. This is updated via standard means and the security state is validated via attestation.
- Recovery: The device may or may not be in a functional state. The device can enter this state via internal error handling mechanisms or the PA-ROT can force recovery if permitted. The mechanism for Recovery is the primary goal of this document.

Response to firmware/configuration failures

Depending upon the failure one of the following actions can be performed:

- The device fails attestation. Standard firmware update mechanisms can be used to bring the device into compliance.
- The device detects a failure and voluntarily enters recovery. This recovery interface can be used to download or select a recovery image.
- The device is unresponsive (e.g. does not respond to MCTP messages). Forced recovery can be used in this case to recover the device. If force recovery is not enabled the device MAY need to be returned to manufacture.

Upon detection of firmware code or critical data corruption, PA-RoT SHALL initiate the recovery process, which MAY be gated by authorization from the system administrator.

Administrative Forced Recovery

The RA instructs the device to enter recovery mode. This can be achieved using a recovery command or physical presence indication (e.g. GPIO). The feature can be administratively disabled but SHOULD be enabled by default. The control enabling this is outside of the scope of this document. This can be used to bring a device back into compliance with minimal reliance on state of the device. This command forces the device into a recovery state regardless of the initial device state. Note: Forced recovery implicity trusts the RA/PA-RoT. This can cause a denial-of-service attack affecting the device availability by using this interface to repeatedly resetting/recover the device.

Recovery requirements across platform components

A recovery process might require recovering several devices or components to ensure a base security state. By performing recovery across all critical components on the platform to a known good set of firmware code, the PA-RoT ensures a recovery to a consistent and known state.

In order to provide long term security objectives, the PA-RoT should provide a means of updating recovery images, since relying on a golden image (static recovery installed at manufacturing) for recovery can lead to roll back to possibly vulnerable firmware. An example

method of updating a recovery image would be through A/B versioning, where one version is latest/active, while the second acts as recovery. In this scheme an update is performed to the recovery version, upon completing the update the latest/active and recovery roles switch between A/B versions. In such A/B versioning schemes, This ensures that the recovery image is not older than the N-1 version. If both A and B copies are corrupted, then a dedicated recovery image (C-image) is used. This image should have the minimal footprint to enable attestation and firmware update to bring the device into compliance. The C image can also be pushed into the device using the indirect memory interface described in this document. Note the C image is signed with the same secure boot keys as the production images and is subject to the same anti-roll back rules.

Summary of images and roles

- A-image: A copy of operational image.
- B-image: B copy of operational image.
- C-image: Recovery image used to install/update A or B copy
- Critical data: data which is critical to the security of the device (e.g. provisioned identity)

Recovery Authorization

The platform policies may require an authorization of the recovery process. This can be implemented through requiring physical presence indication, GPIO connected to the device (AC-ROT/PA-ROT) or through forced recovery interface. Once the PA-ROT has received authorization, the recovery of the platform devices (AC-ROT) can proceed. The PA-ROT can force a recovery through this interface, if enabled, or can use a physical presence/GPIO to enter recovery. The forced recovery interface MAY be disabled via a device specific means, but SHOULD be enabled by default. The PA-RoT SHOULD be capable of performing recovery on demand, at the behest of an authorized entity.

Reasons for Device Recovery

The device shall follow the guidelines defined in the NIST SP800-193 specification, for its firmware protection, tamper-detection and recovery capabilities. A device may or may not be able to persist critical data through a recovery process. Devices unable to maintain critical data MUST go through a process to reestablish this data.

The following sections define the requirements for recovery of a device to an approved state (approved by platform owner).

Device recovery flows for a component (AC-ROT) SHALL be initiated by the PA-RoT (RA) on the following conditions:

1. Tampering of device firmware and/or sensitive security parameters is detected by the platform root-of-trust.
2. Device is in an unknown state/fails to respond.
3. Forced recovery

4. Device detects corruption and enters recovery
5. PA-RoT determines via dynamic means (e.g. SPDM Challenge) the device is out of compliance.

Tampering of device firmware and/or sensitive security parameters SHALL be detected by OCP Attestation.
A PA-RoT MAY use platform specific mechanisms to isolate the device until it can be brought into compliance. An example would be holding the PCIe reset of a device until it is properly recovered.

Device Recovery Use Cases

There are several targeted use cases for recovery. The following section will outline 3 specific use cases. From a NIST SP800-193 perspective, critical data includes provisioned identity as well as security critical parameters. Depending on the type of failure, recovery may not include this critical data.

Device Software Update

In this case, the device has been secure-booted properly (including anti-roll back check) and has a functional attestation agent compliant with the OCP Attestation specification. Upon successful challenge and comparison of measurements against the platform manifest, the device firmware or configuration can be updated using unspecified update techniques. Examples of these update techniques are DMTF PLDM firmware update or Cerberus firmware update. In this case, the recovery process is not required or used.

Device Recovery with Critical Data

The device has entered or was forced into recovery by the RA. If supported, the C image can be enabled from device flash to recover the device. The C image MUST be a dedicated firmware used to bring the device back into compliance. If the recovery image is not present or valid, this recovery image can be pushed into the device using the recovery flow as described in this document. This use case assumes security information (critical data) and identity provisioning (i.e. provisioned certificate stacks) are still available in the device.

Device Recovery without Critical Data

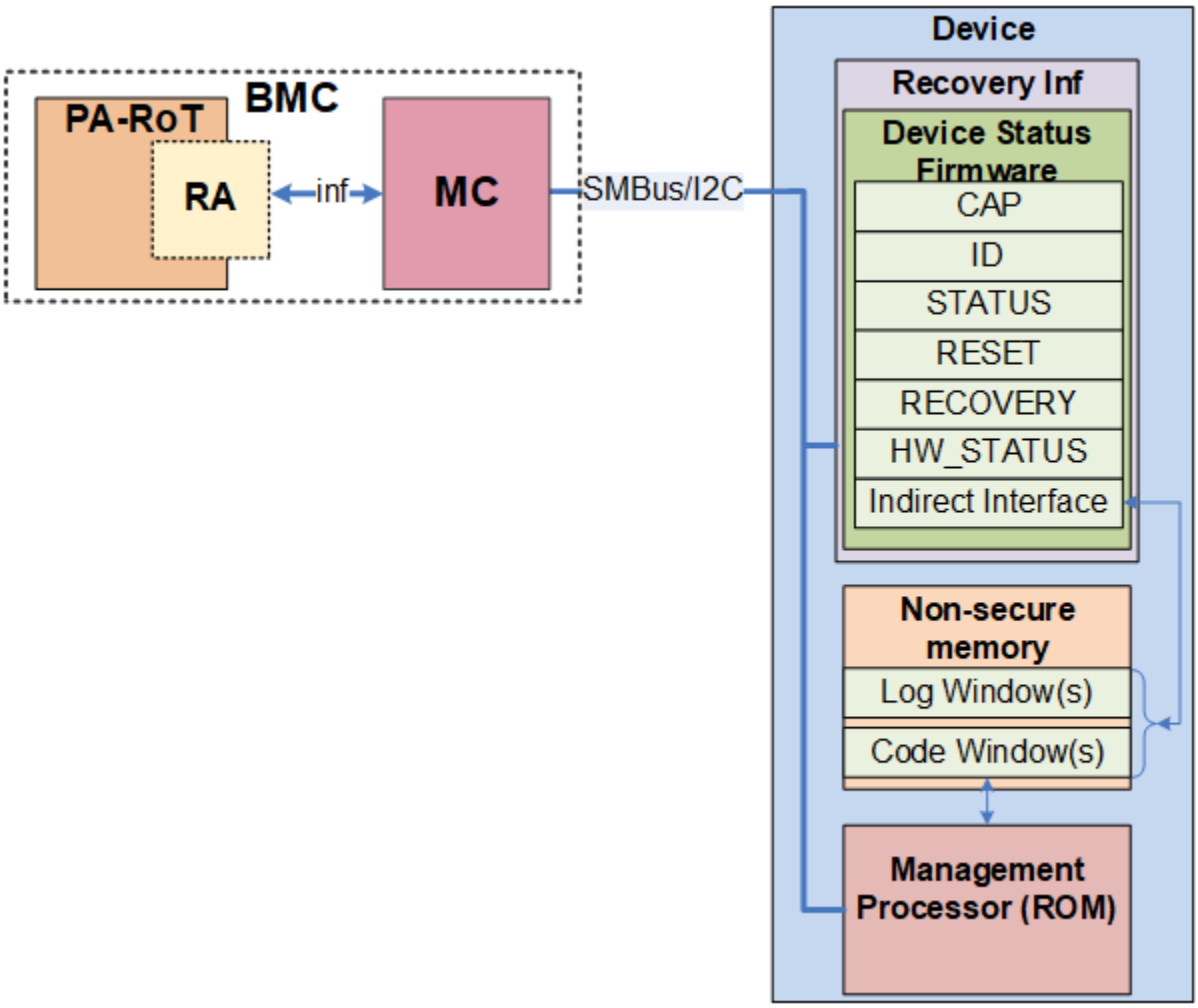
The device has entered or was forced into recovery by the RA and the critical data is corrupted or erased (NOTE: this can not be used for ownership transfer). Critical data is defined in SP800-193 as “mutable data which persists across power cycles and must be in a valid state for the booting of the platform to securely and correctly proceed”. In our context, this can include key manifests, provisioned certificates used for attestation, etc.

The same recovery procedure described above with intact critical data applies. However, once the recovery image is booted the critical data needs to be reprovisioned into the device. There are several means to reestablishing this identity including using a manufacturer provisioned

certificate stack, if available. Depending on the characteristics of the device, it may not be possible to restore the device's critical security data without returning the device to the manufacturer or provisioning facility.

Recovery Components and Roles

The following picture show various components of the recovery process



RA & PA-RoT Recovery Role

The RA & PA-RoT are responsible for the following tasks:

- Maintains the list of approved operational FW images for the devices
- Maintains the list of approved recovery FW images for the devices
- Protects the operational FW images from denial-of-service attacks
- Pushes operational FW images into the device (include recovery images)
    - For devices which store this recovery image locally the PA-RoT and AC-RoT MUST provide a mechanism for updating this firmware.

- Downloads recovery FW images into the device
- Invokes device recovery from either persistently stored recovery FW image or downloaded recovery FW image

AC-RoT Recovery Role

The AC-ROT SHALL perform the following tasks:

- Authenticates FW images during the Secure Boot process
- Authenticates ALL FW images via a cryptographic signature before usage
- Authenticates ALL FW images using key material that is cryptographically bound to immutable keys
- Authenticates downloaded recovery FW image that is transient stored in RAM/DDR before usage
- Authenticates All FW images persistently stored inside the device during Device SecureBoot process
- Enforces Rollback policies using FW version, Key Revocation, anti-rollback counter
- Reports Secure Boot failure error flags. (MAY rely on the secure boot process to validate the image as long as it reports this failure via the Recovery Reason code (missing or corrupted boot firmware image (BFFIMC).)
- Supports Device Attestation identity authentication
- Includes ALL FW images (including persistently stored operational and recovery images) in the reported Device Attestation results
- Manage device's Recovery State

Immutable hardware SHALL serve as the device root-of-trust for recovery purposes. This can be implemented in devices as hard-coded logic and/or immutable ROM code. Depending on the corruption, the device's cryptographic identity may not be available until recovery has been completed.

The device SHALL provide a path for the RA to push a recovery image or select a recovery image from a local source.

Post-recovery, standard OOB attestation should be performed to verify device compliance. In case of recovery, it is possible that the security critical data or provisioned information (i.e. owner's certificate) may have been lost, in such case attestation may fail. In such scenarios the device may require re-provisioning of this information or return to manufacture.

On initiation of the device recovery flow by the RA, the device shall provide a mechanism to replace the current mutable operational firmware image with an approved version that shall be maintained in the PA-RoT. The device SHALL determine validity of the approved version by performing digital signature verification of the firmware according to the OCP Secure Boot

specification. This implies that the recovery image is signed by the same RoT as the operational firmware and uses the same anti-roll back counters.

Recovery Image

The PA-RoT is responsible for maintaining the list of approved firmware for devices and protecting the recovery images from denial-of-service attacks. The RA is responsible for pushing or activating this image onto the device. In the case of recovery image, the PA-RoT SHOULD maintain a list of recovery images for the device. For devices which store this recovery image locally it MUST provide a mechanism for updating this firmware.

The recovery image should be considered an operational image and must follow the OCP secure boot document. Specifically, this image will be signed with the firmware signing key and is subject to the anti-rollback counters. The device SHOULD verify the cryptographic signature of the recovery image as a pre-step before performing recovery. A compliant device MAY rely on the secure boot process to validate the image as long as it reports this failure via the Recovery Reason code.
The following table is a summary of the firmware components.

TERMINOLOGY	DEFINITION
A-Image	A copy of operational image with associated Critical Data[x]. This image is persistently stored in Device's flash memory.
B_Image	B copy of operational image with associated Critical Data[y]. This image is persistently stored in Device's flash memory.
C_image:PersistentRecoveryImage	This FW image installs/updates the A/B images This image is persistently stored in the device's Flash memory. This image is selected via recovery image selection
C_image:TransientRecoveryImage	This FW object installs/updates the A/B images This image is transiently stored in the device's transient memory. This image is selected via recovery image selection
FW_Update:NormalImage	This FW download package updates A/B and MAY contain C_image:PersistentRecoveryImage.
FW_Update:TransientRecoveryImage	This FW image is transformed into C_image:TransientRecoveryImage. This image is transiently stored in RAM.

All FW images shall support the following qualities:
· shall support cryptographic authentication (FW signatures)
· shall be signed with key material that is cryptographically bound to the device's immutable ROOT Key (secure boot)
· shall check for Rollback protection

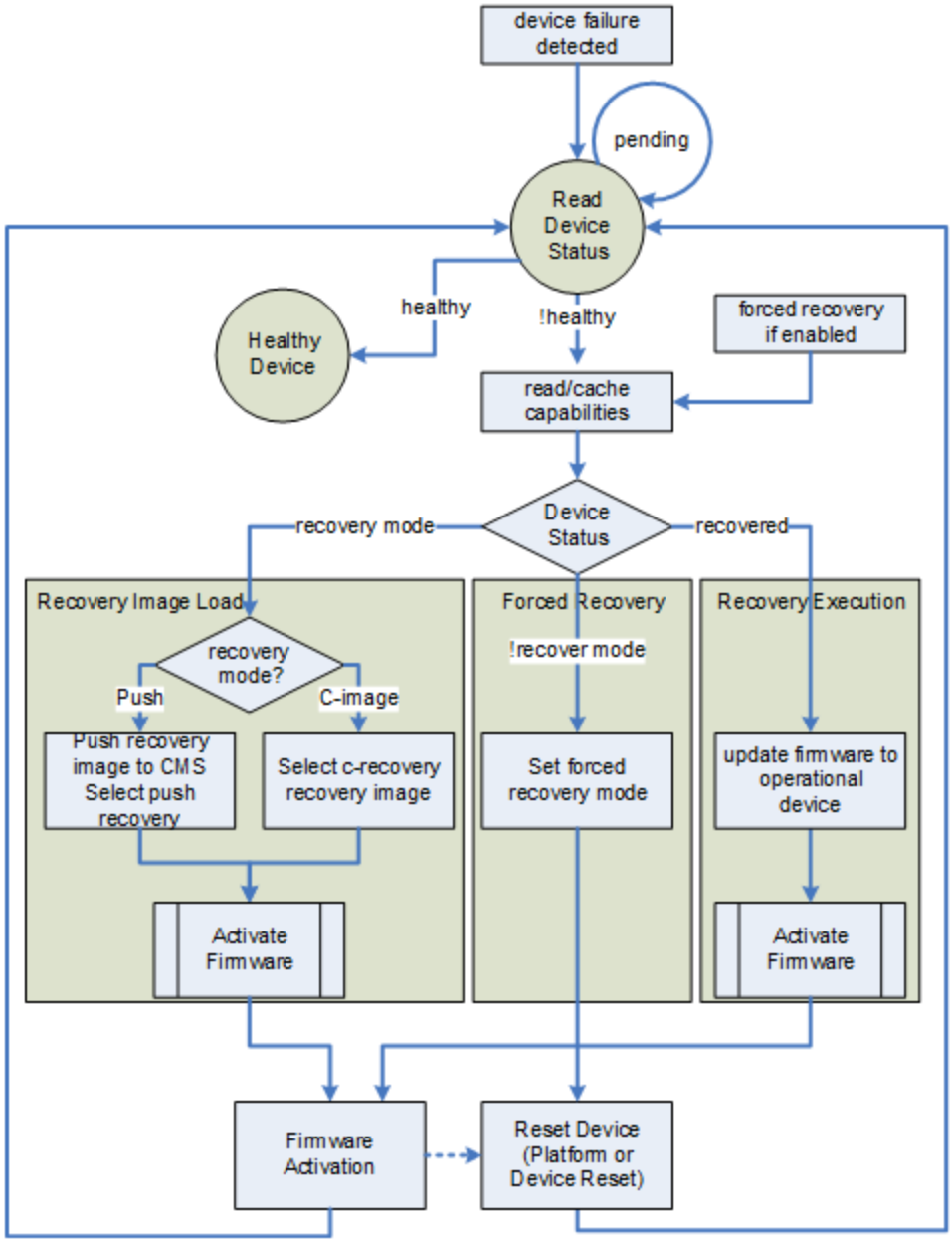
Recovery Process

When needed, the PA-RoT shall orchestrate the recovery flow for devices by using the recovery agent (RA). A recovery agent (RA) is a defined component which is responsible for coordinating the recovery process. The RA MAY be part of the PA-RoT or a separate component. The RA will use the PA-ROT as the source for all images and configuration. A multi-state recovery process may be required in order to bring the device into full compliance.

The health of a device is maintained by the platform and is outside of the scope of this specification. This determination is typically done by using information from the PA-RoT. If a device is declared unhealthy, normal software update procedures (e.g., DMTF PLDM) should be used to bring the device into compliance. That is, the recovery process is not a replacement for a normal software update and should be viewed as a last resort before the device is declared un-recoverable.

The RA can query the device status using this protocol via reading the device status register. The device status MUST indicate status for the software component which is used for attestation. Other software components (e.g. other compute domains within a device) SHOULD NOT be reported in this status register. The status register is only valid when it is not zero (e.g. not pending). This allows time for a boot process on the device to properly reflect the status. This status is informational only and the state of the device SHOULD be cryptographically attested by using the attestation procedure in the OCP attestation documents.

The following picture depicts the overall process to recover a device. The recovery process is entered after a device is declared unhealthy by the platform or the device. A device MAY be administratively commanded to recover via forced recovery, if enabled. Forced recovery MAY be disabled via a device specific means. A device MUST advertise forced recovery via the capabilities described in this spec if a device is capable and enabled. The device status is read to determine the next steps of the recovery process.



A device passes through a number of states in the recovery process. The following definition of the defined states:

- Healthy - device is running an operational image. This state is designed only to reflect the status of the management entity firmware. Specifically, In devices which contain the attestation agent and firmware update process. The PA-RoT is responsible for determining if the device is healthy.
- Not healthy - the device is not healthy. The recovery reason code contains additional information. Depending upon the recovery reason code, additional information can be optionally communicated.
- Recovery state - a device in the recovery state is ready to accept either a pushed image or a command to use the recovery image stored on the device (C image).

- Recovery pending - a device which has completed the image push or selected the recovery image.
- Recovery successful- The recovery image is currently running.

Recover Interface Functions

Device Reset

Multiple device resets may be required to fully recover a device. A device MAY support device reset via this interface (RESET) or rely on a platform reset mechanism. A device MAY support two different types of resets. A device reset via the RESET registers will reset the device and MAY cause a bus enumeration. A device MAY support a management reset where only a subset or management portion of the device is reset. A device which supports management reset:

- MUST NOT cause a bus re-enumeration of the device.
- MUST reset all security components of the device. This includes any processor subsystem responsible for attestation of the device.

Forced Recovery

A device can be commanded to enter recovery mode. This is achieved by writing forced recovery to the RESET register. At the next reset, the device will enter recovery mode. A device MAY disable forced recovery via device specific means. A device which has been commanded to enter recovery but forced recovery is disabled MUST report “Error entering Recovery mode” in the RECOVERY_STATUS register.

Recovery Image Push

A device MAY have entered the recovery state based on local error conditions. For example, a signature failure on the first mutable image loaded by the ROM. The device MUST reflect the recovery state in the device status register. A device in this state is ready to receive or select the recovery image.

A device in recovery is ready to accept the recovery image if supported. This image is pushed via the memory window or a local recovery image can be selected. This selection is written to the RECOVERY_CTRL register. If the push protocol is used it MUST be written using the indirect memory protocol to a memory region (CMS) specified for code. The recovery window CMS MUST be selected in the RECOVERY_CTRL command. The RECOVERY_CTRL command MUST be completed before image activation.

Recovery Image Selection

A device which supports c-image as the recovery image MUST use the RECOVERY_CTRL to select this mode. This command MUST be completed before image activation.

Recovery Image Activation

Once the image is fully written or local c-image is selected, an activation command MUST be performed to activate the recovery image. The activation process MUST ensure the device restarts into the device immutable trust anchor as described in OCP secure boot document. An image waiting for activation of the recovery image MUST report “Recovery Pending" in the DEVICE_STATUS. The result of activation will start running the recovery image. A device MAY uses a management reset to implement the activation function.

NOTE:
It is possible to combine image selection and image activation. In this case, the recovery image and activation are sent in the same command RECOVERY_CTRL (i.e. writing 0xF|imagemode|CMS). This will cause the device to immediately start executing the recovery image.

Recovery Image Authentication and Operation

After the image is activated, it executes as an operational image and MUST pass all security checks defined in the OCP secure boot document. When the recovery image is running it MUST report “Running Recovery Image” in the DEVICE_STATUS registers. The recovery image is responsible for attestation and bringing the firmware or configuration data into compliance. Once the recovery process is complete, the device SHOULD use device reset to activate an operational image.

Normal/Healthy Operation

A device in this state is fully functional and running operational firmware. The device MUST report “Device Healthy” in the DEVICE_STATUS registers when running operational firmware.

Recovery Interface

The recovery interface abstractly is described by block read and write commands. These commands can be implemented using many protocols. The following sections describe different mechanisms.
The following command groups are defined in the following table

Table 1 - Recovery Command Summary

Command	Req	Scope	Notes
PROT_CAP	Y	A	Device Capabilities Information
DEVICE_ID	Y	A	Device identity information
DEVICE_STATUS	Y	A	Device status information

DEVICE_RESET	N	A	Device reset and control
RECOVERY_CTRL	Y	A	Recovery control and image activation
RECOVERY_STATUS	N	A	Recovery status information
HW_STATUS	N	R	Hardware status including temperature
INDIRECT_CTRL	N	R	Indirect memory window control
INDIRECT_STATUS	N	R	Indirect memory window status
INDIRECT_DATA	N	R	Indirect memory window for pushing recovery image

The req column indicates if the command is required. The scope column indicates when the command must be active (e.g., RA can expect a response).

- A - indicates the command should be available anytime the device SMBus interface is available.
- R - indicates the recovery interface must be active. This is indicated by a non-zero device_status.

Capability/Discovery

The capabilities of the device are discovered via reading PROT_CAP.

Indirect Memory Interface

A common indirect access mechanism is defined to facilitate reading and writing memory spaces within the device. Component Memory Spaces (CMS) are mapped directly to the resources within the device in a device specific way. The device resource can be memory, registers, flash or other device resource. This interface allows for a common interface to exchange code, logs or other vendor defined data with the device. If the indirect memory interface is supported, it MUST support at least one memory region. The code, critical and vendor-defined CMS types are defined and described in the following sections.

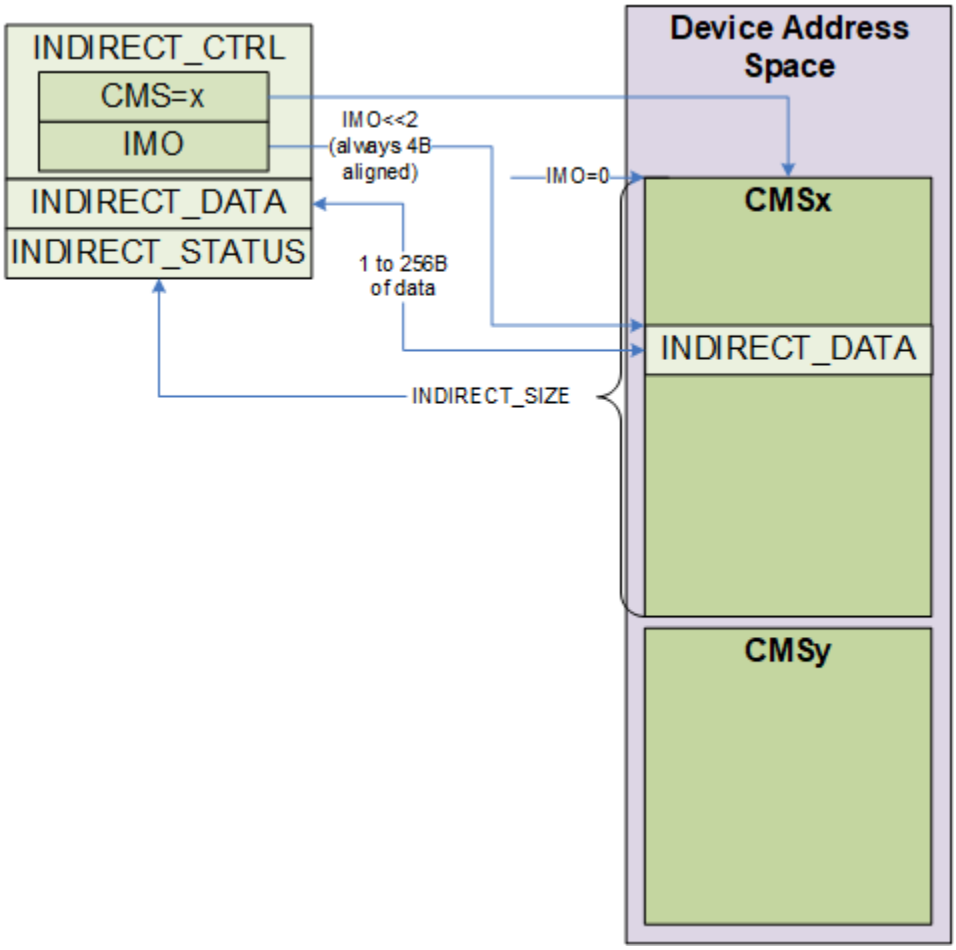
The size (INDIRECT_SIZE) and type (INDIRECT_TYPE) of memory space is queried by writing the memory region to the INDIRECT_CTRL register and reading the INDIRECT_STATUS. A CMS can use either polling or direct access. This is reported as the high-order bit of the CMS type.

Addressing within a Component Memory Spaces

Addressing within a CMS maps the INDIRECT_DATA window into the CMS. An indirect memory offset (IMO) is maintained within the current CMS and is always 4-byte aligned. The base address of the CMS within the device is vendor defined. Before writing to a CMS, the type and

size of a CMS SHOULD be determined by the INDIRECT_CTRL command. The first byte of an address region is defined as byte 0 written to the INDIRECT_DATA registers when the indirect memory offset (IMO) is zero. The IMO is incremented by the number of bytes written to the INDIRECT_DATA registers module 4 bytes. Changing the CMS or IMO in the INDIRECT_CTRL MUST reset the IMS counter to zero. For non-polling regions, the device is expected to be able to accept continuous requests. If a region requires polling, the ACK status is reported in the INDIRECT_STATUS register when the device can accept the next transaction. The RA MUST poll the INDIRECT_STATUS register before the next INDIRECT_DATA transaction. The ACK indication is cleared on read.

The following figure shows the relationship between various indirect commands, the CMS and device resources.



Error conditions

- Address Wrapping: (e.g the IMO extends beyond the reported size). This MUST wrap to the beginning of the buffer AND report an overflow in the INDIRECT_STATUS.
- Writing to a read only CMS: This MUST not write to the internal address space and report an access error in the INDIRECT_STATUS.

- Writing or Reading to a polling enabled region when not ready. This transaction MUST be ignored and MUST NOT increment the IMO. A polling error in the INDIRECT_STATUS must be reported.
- Unaligned access to CMS. IMO address will be truncated (e.g. lower bit set to zero)

Code CMS

A code region is designed to deposit code (the recovery image) to facilitate the recovery process. A device which supports code push MUST support at least one memory region and be mapped to Memory region 0 (CMS=0). Multiple code spaces can be used to support multiple domains within the device, but these are used in a vendor specific way.
Once all bytes of the recovery image are written to the device, then the image can be activated by writing the recovery CMS to the RECOVERY_CTRL register.
For these spaces, care MUST be taken to prevent time-of-check to time-of-use attacks. One way to accomplish this is to map the code region to a non-secure region and close the region after it has been activated.

Critical Logging CMS

This CMS used for logging is defined as read only. Support of critical logging is optional. Write access to the region MUST not make changes to the CMS and report an error in the INDIRECT_STATUS register. The critical logs are not signed and no security guarantees are provided. In addition, the logs may not persist a device reset.
The log is viewed as a circular buffer. Entries are added to the log in a sequential fashion. The entry identifier must be unique and monotonic. Based on parsing of the debug log and entry identifiers, the event sequence can be inferred. The general structure uses a magic number, length and entry_id. The entry contains a debug log entry which contains a format and opaque payload.

```
LOG_MAGIC_NUMBER 0xE5E5
struct debug_log_entry {
   struct logging_entry_header header;         /**< Standard logging header. */
   struct debug_log_entry_info entry;          /**< Information for the log entry. */
};
struct logging_entry_header {
   uint16_t log_magic;                         /**< Start of entry marker. */
   uint16_t length;                            /**< Total Length of the entry. */
   uint32_t entry_id;                          /**< Unique entry identifier. */
};
struct debug_log_entry_info {
   uint16_t format;              /**< Format of the log entry (msg_body) */
   uint8_t msg_body[];           /**< body of log message. */
};
```

Vendor Defined CMS

Two types of vendor defined regions are defined, one which is read only (vendor defined logs) and one which is read/write.

Recovery SMBus Interface

This section describes a recovery protocol based on SMBus/I2C block read and write commands. The recovery protocol requires an SMBus compliant interface between RA and the managed card used to transport the recovery protocol. The recovery protocol is designed to be simple and MUST be embedded into ROM or dedicated hardware.

As per SMBus 3.1 spec, each Block Write/Read contains a byte for the Command Code, followed by a byte for Byte Count and up to 255 bytes of data. Command data sizes vary depending on the commands; refer to table below for specifics.
The following are the requirements for the recovery interface for SMBus/I2C:

- MUST be compliant with ”System Management Bus (SMBus) Specification version 3.1, 19 Mar 2018”.
- MUST support physical layer per spec specified in [SMB 31]
- MUST support Class 100 kHz operation
- SHOULD support Class 400 kHz and Class 1 MHz operation
- MUST support data link layer per spec in [SMB 31]
- MUST support network layer per spec in [SMB 31]
- MUST support block read and write with and without PEC protocols.
- SHOULD support and use Address Resolution Protocol (ARP) for dynamic target address assignment.
- SMBus interface and recovery agent SHOULD be designed to have maximum uptime and have minimal external dependencies (e.g., flash).
- MUST respond to recovery commands sent by RA once target address is assigned or to default address
- MUST support target functionality
- MUST not support master functionality when in recovery mode
- MUST support a fixed I2C address, if ARP is not used. If a shared topology is used then the default SHOULD be 0xD4 (7-bit) for a separate topology it SHOULD be 0xD2.
- SHOULD use PEC checksum per SMB 3.1

The recovery protocol does not require MCTP or any variant of protocol that runs on top of MCTP. The recovery protocol does not depend on bidirectional communication initiation. The recovery protocol MAY exist on the same interface as the one used for MCTP, provided there is a way to deliver non-MCTP commands to the recovery interface. The recovery interface can also be a separate standalone SMBus interface. The following two diagrams depict a few different topologies. NOTE: The preferred topology is a separate address for the recovery interface. I3C will only support this topology since the command byte is not specified in the I3C specification.

SMB Topology

There are two topologies supported. In the first topology, the recovery interface and the MCTP EP share the same SMBus address. Note in this topology the default I2C address SHOULD be 0xD4.

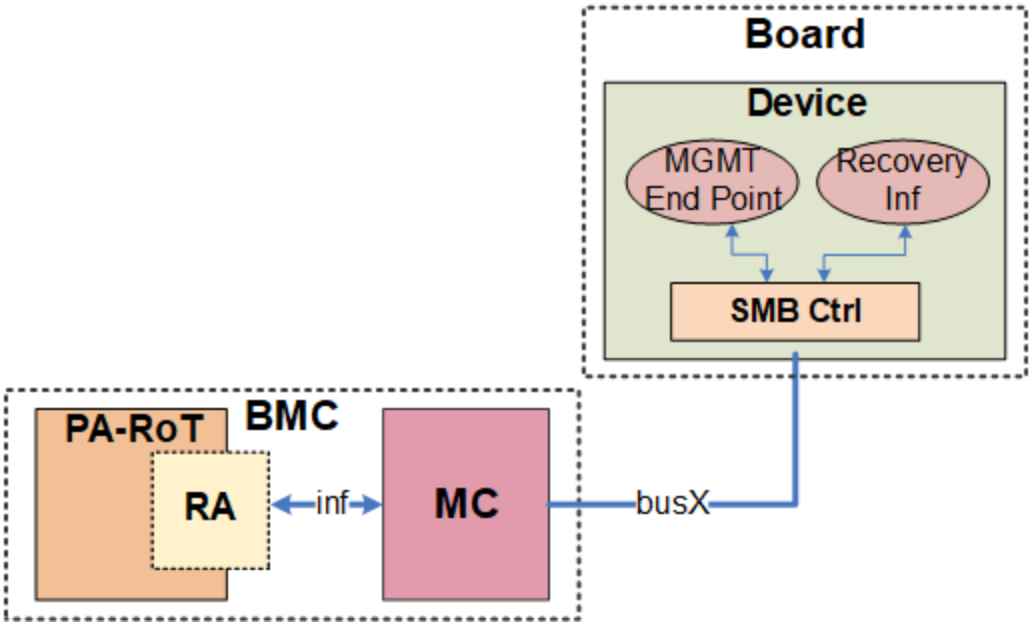


Diagram of shared SMBus interface

The second topology has separate components for the recovery interface and MCTP End Point (EP). Note the connection between the controllers can be external (e.g. separate pins) or internal. Note2 in the topology the default I2C address SHOULD be 0xD2.

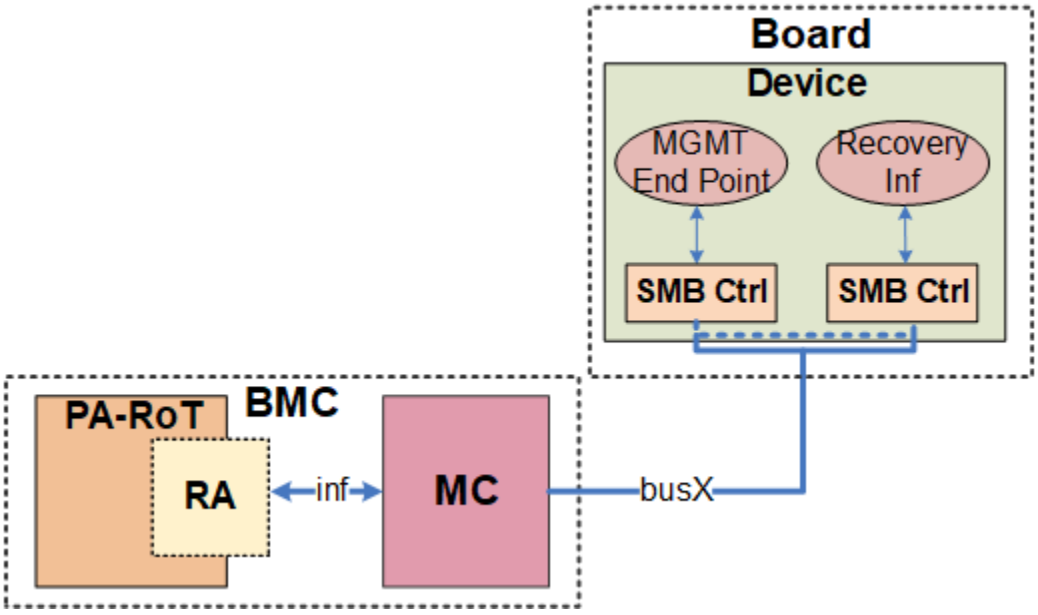


Diagram of separate SMBus interface for recovery

SMBus Device Addressing and Commands

The SMBus protocol uses a 7-bit device addressing and will be used for recovery. In addition, an 8-bit command byte is defined in the block read/write commands. A compliant device MUST support the required commands using SMB block read and write interface.

Several other protocols/standards which use block read and write commands for various functions. The following is a list of considered standards.

- DMTF MCTP over SMBus transport (v 1.0.0)
    - This specification reserves command 16 (0x0F)
- NVMe-MI OOB basic management interface (v1.2)
    - This specification reserves commands 0, 8 and 32
- OCP NVMe Cloud SSD Specification (v1.0)
    - Section 10.2 command 0, 8, 32, 50, 90, 96, 154, 242, 248

This specification does not overlap or conflict with command allocation from these standards. Therefore, a compliance device can implement the recovery interface and be compatible with these standards on the same interface. Care must be taken to ensure future compatibility with these standards.

Interface Sharing/Isolation

SMBus is a multi-master protocol without fair arbitration. Device firmware could cause a denial-of-service to the recovery interface by mastering transactions that win arbitration in perpetuity. This would prevent the RA from issuing a device reset or forced recovery commands. To mitigate this a device SHOULD disable SMBus mastering out of power-on. Bus Mastering can be enabled via device specific means OR by using the interface master enable.

For devices which support interface isolation, it MUST report interface isolation in the PROT_CAP command. For capable devices, the power up condition MUST be disabled mastering in the isolation field of the DEVICE_RESET register. The RA MUST enable mastering, in the DEVICE_RESET register, when it determines the device is healthy. Note: MCTP notification will fail if mastering is not enabled.

Recovery Interface Commands

This section describes the command defined for the recovery interface. It is described as a generic block read and write protocol with a command byte. Not all commands are required to be implemented and it is up to the RA to determine which ones are available via the PROT_CAP command. All commands and fields are specified in little-endian format.

Error Handling/Unsupported Features

There are several errors which can occur in the protocol. The PROTOCOL_ERROR field in the device status register is used to indicate these errors. Reading the device status register will clear the PROTOCOL_ERROR field. Note: the PROTOCOL_ERROR is a clear on read field.

- An unsupported command to a device MUST set an unsupported error condition in the DEVICE_STATUS. This includes all optional commands.
- A device which receives (write) commands with unsupported parameters (e.g. local c-image selection when the device it is not supported) MUST generate an ‘unsupported parameter’ error in the DEVICE_STATUS registers.
- A device which receives (write) with an incorrect number of bytes MUST generate an ‘length write error’ error in the DEVICE_STATUS registers.
- A device which receives (write) with an invalid checksum (e.g. PEC) MUST set the CRC error in the DEVICE_STATUS.
- Writing to a read only command (e.g. PROT_CAP) MUST generate an ‘unsupported command’ error in the DEVICE_STATUS.

Command Summary

Recover Capabilities Command				
Command	**r/w**	**bytes**	**Description**	**Req**
PROT_CAP cmd=34	ro	15	**Recovery protocol magic string** Byte 0-7: Magic string “OCP RECV” in ASCII code - “4f 43 50 20 52 45 43 56” **Recovery protocol version** Byte 8: Major version number = 0x1 Byte 9: Minor version number = 0x0 **Recovery protocol capabilities** Byte 10-11: **Agent capabilities** BIT 0: Identification (DEVICE_ID structure) BIT 1: Forced Recovery (From RESET) BIT 2: Mgmt reset (From RESET) BIT 3: Device Reset (From RESET) BIT 4: Device status (DEVICE_STATUS) BIT 5: Recovery memory access (INDIRECT_CTRL) BIT 6: Local C-image support BIT 7: Push C-image support BIT 8: Interface isolation BIT 9: Hardware status Bit 10: Vendors command BIt 11-15: Reserved Byte 12 (0-255): The total number of component memory space (CMS) regions a device supports. This number includes any logging, code and vendor defined regions	Y

			Describes the maximum amount of time an operation can take. A device SHOULD not take more than 100 ms to respond to an operation. Byte13: **Maximum Responses Time** 0-255: Maximum response time in 2^x microseconds (us). Byte14: **Heartbeat Period** 0-255:: Heartbeat period:n 2^x microseconds (us) - 0 indicates not supported	

Mandatory capabilities are:

- DEVICE_ID
- DEVICE_STATUS
- Local C-image OR Push C-Image
- INDIRECT_CTRL if Push C-Image

The device identifier is used to identify the type of device. This DEVICE_ID command is designed to retrieve data to construct the device identifier record per the DMTF Firmware update standard.

Command	**r/w**	**bytes**	**Description**	**Req**
ID			Device Identification	
DEVICE_ID cmd=35	ro	24-255	Number of bytes available for each of the following IDs: Byte 0: **Initial Descriptor Type**:- Based on table 8 from [DMTF PLDM FM] 0x00: PCI Vendor 0x1: IANA 0x2: UUID 0x3: PnP Vendor 0x4: ACPI Vendor 0x5: IANA Enterprise Type 0x6-0xE: Reserved 0xFF: NVMe-MI Byte 1: **Vendor Id String Length** 0-0xFF: length of Vendor String . 0 indicates not supported **For PCI Type:** Byte 2-3: PCI Vendor ID Byte 4-5: PCI DeviceID Byte 6-7: PCI Subsystem Vendor ID Byte: 8-9: PCI Subsystem ID Byte: 10: PCI Revision ID Byte: 11-23: 0x0 (PAD) **For UUID type:** Byte: 2-17: UUID assigned to the device Byte: 18-23: 0x0 (PAD) **For IANA type:** Byte: 2-5: IANA Enterprise ID Byte: 6-17: ACPI Product Identifier Byte: 18-23: 0x0 (PAD) **For PnP type:**	Y

			Byte: 2-4: PnP Vendor Identifier Byte: 5-8: PnP Product Identifier Byte: 9-23: 0x0 (PAD) **For ACPI type:** Byte: 2-5: ACPI Vendor Identifier Byte: 6-8: Vendor Product Identifier Byte: 9-23: 0x0 (PAD) **For NVME-mi** Byte: 2-3 Vendor ID Byte: 4-23: Device Serial Number Vendor Specific String: **Vendor Specific String** Byte:24-254 - ASCII encoded string	

Command	**r/w**	**bytes**	**Description**	**Req**
STATUS			Device Status - Accumulated device status	
DEVICE_STATUS cmd=36	ro	7-255	Byte 0: **Device status** 0x0: Status Pending (Recover Reason Code not populated) 0x1: Device healthy (Recover Reason Code not populated) 0x2: Device Error (“soft” error or other error state) - (Recover Reason Code not populated) 0x3: Recovery mode - ready to accept recovery image - (Recover Reason Code populated) 0x4: Recovery Pending (waiting for device/platform reset) - (Recover Reason Code populated) 0x5: Running Recovery Image (Recover Reason Code not populated) 0x6-0xD: Reserved 0xE: Boot Failure (Recover Reason Code populated) 0xF: Fatal Error (Recover Reason Code not populated) 0x10-FF:Reserved Byte 1: **Protocol Error (Clear on Read)** 0x0: No Protocol Error 0x1: Unsupported/Write Command - command is not support or a write to a RO command 0x2: Unsupported Parameter 0x3: Length write error (length of write command is incorrect) 0x4: CRC Error (if supported) 0x5-0xFF: Reserved Byte 2-3: **Recovery Reason Codes** - See table 3 Byte 4-5: **Heartbeat** 0-4095 - Incrementing number (counter wraps) Byte 6: **Vendor Status Length** 0-248: Length in bytes of just VENDOR_STATUS. Zero indicates no vendor status and zero additional bytes. Byte:7-254: **Vendor Status (if vendor length is non-zero)** Vendor defined status message	Y

There are various conditions where the reason code is populated. The goal is to describe why the device failed to boot or entered recovery mode. The following table describes the source of the recovery reason code in different device status/states.

Table 2 - Recovery Reason Code Population

Device Status/State	Valid	Source of Recovery Reason Code	Notes
Status Pending	N	None	Device is booting or has not yet populated the reason code
Device healthy	N	None	
Device Error	N	None	
Recovery Mode	Y	Previous boot	Device has entered recovery based on forced recovery, or error in the previous boot.
Recovery Pending	Y	Previous boot	Device has entered recovery based on forced recovery, or error in the previous boot.
Boot Failure	Y	Current Boot	Device current boot is halted. The reason is defined in the recovery reason code.
Fatal Error	N	None	

Recovery reason codes are defined in the following table. The RCV column indicates if the device can be recovered based on error code (Y-yes, N-no, M-possible).

Table 3 - Recovery Reason Codes		
Code	**Description**	**RCV**
0x0	No Boot Failure detected (BFNF)	N
0x1	Generic hardware error (BFGHWE)	N
0x2	Generic hardware soft error (BFGSE) - soft error may be recoverable	M
0x3	Self-test failure (BFSTF) (e.g., RSA self test failure, FIPs self test failure,, etc.)	M
0x4	Corrupted/missing critical data (BFCD)	M
0x5	Missing/corrupt key manifest (BFKMMC)	Y
0x6	Authentication Failure on key manifest (BFKMAF)	Y
0x7	Anti-rollback failure on key manifest (BFKIAR)	Y
0x8	Missing/corrupt boot loader (first mutable code) firmware image (BFFIMC)	Y
0x9	Authentication failure on boot loader (1st mutable code) firmware image (BFFIAF)	Y
0xA	Anti-rollback failure boot loader (1st mutable code) firmware image (BFFIAR)	Y
0xB	Missing/corrupt main/management firmware image (BFMFMC)	Y
0xC	Authentication Failure main/management firmware image (BFMFAF)	Y
0xD	Anti-rollback Failure main/management firmware image (BFMFAR)	Y
0xE	Missing/corrupt recovery firmware (BFRFMC)	Y
0xF	Authentication Failure recovery firmware (BFRFAF)	Y
0x10	Anti-rollback Failure on recovery firmware (BFRFAR)	Y
0x11	Forced Recovery (FR)	Y
0x12 – 7F	Reserved	NA
0x80 - FF	Vendor Unique Boot Failure Codes	NA
0x0100-0xFFFF	reserved	NA

Command	**r/w**	**bytes**	**Description**	**Req**
DEVICE_RESET			Reset Control - combinations Reset, Recovery and Activate are possible	
RESET cmd=37	rw	3	Reset control - For devices which support reset, this register will reset the device or management entity. Byte 0: **Device Reset Control (Write 1 Clear e.g., after action device starts 0x0)** 0x0: No reset 0x1: Reset Device (PCIe PRESET or equivalent. This is likely bus disruptive) 0x2: Reset Management. This reset will reset the management subsystem. If supported, this reset MUST not be bus disruption (cause re-enumeration) 0x3-FF: Reserved Mode Byte 1: **Forced Recovery** 0x0 - No forced recovery 01-E - Reserved	N

			0xF - Enter recovery mode on next platform reset 0x10-FF: Reserved Byte 2: **Interface Control** 0x0: Disable Interface mastering 0x1: Enable Interface mastering	

Example device reset commands (starting with byte 0):

- Write: 0x02, 0x0F, 0x00 - device will management reset AND enter forces recovery AND disables interface mastering
- Write: 0x00, 0x0F, 0x00 - device will enter recovery on next platform reset AND disables interface mastering

Interface control is used to enable target initiated transactions (e.g., Bus mastering in SMBus). The device must power-on to mastering disabled and the bus configuration solely managed by the RTRec. Otherwise the DEVICE_RESET command is subject to denial-of-service attacks by device components outside of the RTRec (e.g., management controller).

Command	**r/w**	**bytes**	**Description**	**Req**
RECOVERY				
RECOVERY_CTRL cmd=38	rw	3	Recovery configuration/ctrl Selects the memory region address used for recovery. This region must be a code region. Byte 0: **Component Memory Space (CMS)** 0-255: Selects a component memory space where the recovery image is. 0 is the default Byte 1: **Recovery Image Selection** 0x0: No operation 0x1: Use Recovery Image from memory window (CMS) 0x2: Use Recovery Image stored on device (C image) 0x3-FF: reserved Byte 2: **Activate Recovery Image (Write 1 Clear)** 0x0 - do not activate recovery image - after activation device will report this code. 01-E - Reserved 0xF - Activate recovery image 0x10-FF-reserved	Y
RECOVERY_STATUS cmd=39	ro	2	Recovery status: Recovery Debug status of device Byte 0: **Device recovery status** Bit [7:0]: 0x0: Not in recovery mode 0x1: Awaiting recovery image 0x2: Booting recovery image 0x3: Recovery successful 0xc: Recovery failed 0xd: Recovery image authentication error 0xe: Error entering Recovery mode (might be administratively disabled) 0xf: Invalid component address space 0x10-FF: Reserved	Y

			Byte 1: **Vendor specific status** Bit [7:0]: Vendor Defined	

Command	**r/w**	**bytes**	**Description**	**Req**
HW_STATUS				
HW_STATUS cmd=40	ro	4-255	Byte 0: **HW Status (bit mask active high)** bit 0: Device temperature is critical bit 1: Hardware Soft Error (may need reset to clear) bit 2: Hardware Fatal Error bit 3-7:Reserved Byte 1: **Vendor HW Status (bit mask active high)** bit 0-7: Vendor Specific Byte 2: **Composite temperature (CTemp) -** Current temperature of device in degrees Celsius: Compatible with NVMe-MI command code 0 offset 3. 0x00-0x7e: 0 to 126 C 0x7f: 127 C or higher 0x80: no temperature data, or data is older than 5 seconds 0x81: temperature sensor failure 0x82-0x83: reserved 0xc4: -60 C or lower 0xc5-0xff: -1 to -59 C (in two's complement) Byte 3: **Vendor Specific Hardware length (bytes)** 0-251: Length in bytes of HW_VENDOR structure. **Vendor Specific Hardware Status** Byte 4-254	N

Command	**r/w**	**bytes**	**Description**	**Req**
INDIRECT			This is the interface to memory regions within the device	
INDIRECT_CTRL cmd=41	rw	6	Indirect memory access configuration. This register selects a region within the device. Read/write access is through address spaces. Each space represents a location in memory which is described by the config register. The INDIRECT_OFFSET can be used to access certain offsets within an address space. Byte 0: **Component Memory Space (CMS)** 0-255 - Address region within a device.	N

			Indirect memory configuration: Byte 1: Reserved Byte 2:5 **Indirect memory offset (IMO)** Writes and reads via INDIRECT_DATA will auto-increment this offset by the number of bytes written/read. The offset must be 4-byte aligned (e.g, the lower 2-bit are always zero). Writes and reads that are not 4-byte multiples will increment to the next 4-byte offset. **Note**: IMO can be read to determine the number of bytes read or written since last initialization (write to this register).	
INDIRECT_STATUS cmd=42	ro	6	Byte 0: **STATUS** (bit mask) - (**clear on read**) bit 0: Overflow CMS wrapped bit 1: Read Only Error - write to a RO (log area) bit 2: ACK from device in a polling address space Note: The RA MAY poll with a timeout. The device MUST respond within **Maximum Responses Time reported** in the PROT_ID command. If the timeout expires, then the RA must read RECOVERY_STATUS bit 3-7: Reserved Byte 1: **Type of region**: 0bP000: Code space for recovery. (read/write) 0xP001: Log uses the defined debug format (read only) 0bP101: Vendor Defined Region (read/write) 0xP110: Vendor Defined Region (read only) 0xX111: Unsupported Region (address space out of range) If P is set, polling is required for this region. That is, the ACK must be set before the next indirect operation Byte 2-5: **INDIRECT SIZE** - size of memory window specified (by CMS in the INDIRECT_CTRL component and type) in 4B units	N
INDIRECT_DATA cmd=43	rw	1-255	Indirect memory access to address space configured in INDIRECT_CTRL at offset specified in INDIRECT_OFFSET. Note: The length of the transfer does not need to be 4-byte aligned, but the IMO will be auto-incremented to the next 4-byte offset, so data size should be 4-byte aligned for contiguous access.	N

Command	**r/w**	**bytes**	**Description**	**Req**
VENDOR			Vendor defined command	
VENDOR cmd=44	rw	1-255	Vendor Defined	N

Protocol Conformance Checklist/Statement

The following table is used to describe a device compliance to the recovery protocol.

Table 4 - Recovery Protocol Compliance Statement		
Feature	**Description**	**Compliance**
PROT_CAP mandatory	Mandatory capabilities:	DEVICE_ID, DEVICE_STATUS, Local or Push C-image, INDIRECT if push C-image
PROT_CAP optional	Optional capabilities:	forced recovery, mgmt_reset, hardware status, recovery_memory access, heartbeat
PROT_CAP	Response time	Xx ms
PROT_CAP	Protocol version	{0x1, 0x0}
PROT_CAP	Heartbeat support	Yes or No
PROT_CAP	Heartbeat period values	Xx seconds
PROT_CAP	Recovery image type	Push, local, both
PROT_CAP	CMS Support	Yes or No
PROT_CAP	Device Bus Isolation	Yes or No
DEVICE_ID	Supported Device ID format	PCI Vendor, IANA, UUID, PnP Vendor, ACPI Vendor, or NVMe-MI
DEVICE_ID	Vendor specific string	Value and length
DEVICE_STATUS	Recovery Reasons codes	Enumerate supported reasons codes from table 3
DEVICE_STATUS	Vendor Status	Supported? Length and description
DEVICE_RESET	Management reset support	Yes or No
DEVICE_RESET	Device reset support	Yes or No
DEVICE_RESET	Forced recovery support	Yes or No
RECOVERY_CTRL	Recovery Status support	Yes or No
HW_STATUS	HW status support	Yes or No
HW_STATUS	Vendor HW Status	Yes or No, description if yes
HW_STATUS	Composite temperature support	Yes or No
INDIRECT	Indirect Access command support	Yes or No
INDIRECT	Number and size of CMS code spaces	0-255, 0 – 4GB

INDIRECT	Number of CMS log spaces	0-255, 0-4G
INDIRECT	Number of Vendor CMS code spaces	0-255, 0-4G
INDIRECT	Number of Vendor CMS log spaces	0-255, 0-4G
INDIRECT	Is polling required for any CMS	Yes or No
VENDOR	Is vendor defined command supported	Yes or No, description if yes
SMBUS	Speed classes supported	100K, 400K, 1M
SMBUS	PEC support	Yes or No
SMBUS	ARP support	Yes or No
SMBUS	Fixed address support	Yes or No, is this device configurable?

The following is a list of tests which SHOULD be performed to demonstrate compliance to this protocol. The list is not exhaustive, but is designed to give implementer guidance from the authors as to the important or non-obvious parts of the protocol.

Table 5 - Recovery Protocol Error/Test

Test	Description	Compliance
Unsupported commands	Read and write using unsupported commands	Verify protocol error and COR behavior
Read Only	Write to read only commands	Verify protocol error and COR behavior
Write Error	Write a command with an incorrect number of bytes	Verify protocol error and COR behavior
Write PEC error	If supported write using an incorrect PEC	Verity correct SMBus behavior and Verify protocol error
Device Status	Read device status while device is not ready	The device must report pending in the device status register.

Glossary and Abbreviations

See Glossary and Abbreviations

Relevant standards, guidelines, and documents

[1] NIST Special Publication 800-155 (DRAFT), BIOS Integrity Measurement Guidelines

[2] NIST Special Publication 800-193, Platform Firmware Resiliency Guidelines

[3] Open Compute Project, Project Cerberus Firmware Update Specification

[4] SMBus 3.1 Specification

[5] OCP Attestation of System Components v1.0 Requirements and Recommendations

[6] NVM-Express-Management-Interface-1.2a

[7] DMTF PLDM for Firmware Update DSP0267_1.1.0.pdf (dmtf.org)

[8] OCP NVMe Cloud SSD Specification (v1.0)

License

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