OPEN
Compute Project

Open Hardware Management

Firmware Update SRS
Version 0.2

Revision History

Version	Date	Description
0.1	10/11/2013	Version 0.1 draft for general discussion.
0.2	8/27/2014	Incorporate feedback from team.

Contents

1. Summary

The specification identifies the firmware update requirements that all OCP compliant platforms and devices must adhere to. The requirements are broken into four sections:

- *General – general firmware update requirements*
- *Security – requirements related to the integrity and authenticity of the image*
- *Control – requirements related to how firmware updates are controlled*
- *Status – requirements related to how/what status is available*

The Firmware Update solution mentioned in this document encompasses not just the application the user interfaces with but also the capabilities of the embedded firmware on the platform.

2. License

3. General Requirements

[FW.GEN.01] Firmware update solution shall support programming the BIOS image.

In some motherboard designs, the BIOS image can be updated through the BMC if the EEPROM that stores the image is also connected to the BMC.

[FW.GEN.02] Firmware update solution shall support programming the BMC image.

The BMC image is typically stored on an EEPROM and is controlled by the BMC itself.

[FW.GEN.03] Firmware update solution shall support programming the embedded NIC firmware images, when applicable.

Some embedded NICs support firmware update via sideband signals from the BMC. However, because the sideband interface is slow, many NIC vendors only support firmware update via in-band over PCIe from host processor.

[FW.GEN.04] Firmware update solution should support programming the PSU firmware images.

PSU firmware is typically updated through PMBus which can be connected to a BMC.

[FW.GEN.05] Firmware update solution shall support programming the firmware for RAID components.

Traditionally RAID firmware is accomplished in-band over PCIe interface. RAID firmware updates done over sideband is not always supported by RAID vendors. When/If it is supported over sideband, the firmware update process would take a relatively long time considering the file size of RAID firmware and the inherently slow speed of the SMBus.

[FW.GEN.06] Firmware update solution shall support programming the firmware for Switches.

[FW.GEN.07] Firmware update solution shall support programming the firmware at the rack level.

The firmware update solution should not only consider the programmable devices at the FRU level but also how it can be scalable to support firmware updates at the rack level.

[FW.GEN.08] Firmware update solution shall support programming the firmware at the chassis level.

The firmware update solution should not only consider the programmable devices at the FRU level but also how it can be scalable to support firmware updates at the chassis level.

[FW.GEN.09] Firmware update solution shall support programming the firmware for all drives.

The firmware update solution needs to support firmware update of any captive drives in the platform as well as any drives connected via adapter cards (e.g. SAN).

[FW.GEN.10] Firmware update solution shall leverage industry standards so a consistent command structure can be applied for any firmware being updated.

For ease of use, the firmware update solution must have a common user interface to support any programmable device. The command may also support additional options/parameters to carry OEM data needed by a particular device.

[FW.GEN.11] Firmware update solution must provide a means to query the current firmware versions loaded on all programmable devices.

Identify all installed firmware images on all programmable devices. Since there is no standard format for version numbers, this requirement is limited to the query method/API. Version syntax is beyond the scope of this requirement.

[FW.GEN.12] Firmware update solution should provide the capability to restore configurable settings back to a default state.

This could be accomplished by the firmware update solution or a capability of a programmable device. Implementation is at the discretion of the supplier.

A potential use case could be a user wants to restore system back to factory default. If the user updates the firmware to the original factory firmware, the configurable settings should not change per FW.GEN.14. This requirement will provide the user with the capability of also restoring the configurable settings back to default.

[FW.GEN.13] Firmware update solution should have the capability to save and restore configurable settings of programmable devices within the platform.

For security reasons, usernames and passwords are excluded. This allows for cloning configurable settings of like systems.

[FW.GEN.14] Firmware update solution should retain all configurable settings through a firmware upgrade.

When a firmware update is applied, any existing device settings shall be retained and new settings shall be set to default. If there is a conflict between a new setting in its default state and an existing setting, the vendor shall default both to the default state.

[FW.GEN.17] All firmware image version numbering should use a versioning system such that software can distinguish a new version from an older version by the version number.

Note: Standard for version numbering is yet to be defined.

4. Security Requirements

[FW.SEC.01] Firmware update solution and supporting code shall provide a way to authenticate the image prior to programming the image to the device.

The embedded firmware must authenticate the firmware image against a known root of trust to prove chain of custody of an image. That is, ensure the image has come from the intended supplier and it was not maliciously tainted.

[FW.SEC.02] Firmware update solution and supporting code shall provide a way to do an integrity check of the image prior to programming the image to the device.

This step will check to ensure the image is intact as intended and there was no corruption while the image was transferred or at rest.

[FW.SEC.03] Firmware update solution shall perform an integrity check on the payload of the image as it transfers to the device.

When an image is transferred from some storage to the device being programmed, an integrity check shall be performed on the data that is being transferred.

[FW.SEC.04] Platform firmware shall support secure boot *[Need to list specific standards to adhere to here]*.

Platform firmware must provide a method for ensuring installed firmware and drivers are authentic and fully intact before executing the code.

[FW.SEC.05] Firmware update solution shall generate an event log when the firmware update is requested. The event log shall contain: the user who requested the firmware update, IP from where the request came from (if remote), time stamp information when the firmware update was requested, filename, current version and new version.

5. Control Requirements

[FW.CON.01] Firmware update solution must provide a method for providing a backwards compatibility check before applying the image update.

At times there are rev locks between firmware images. The firmware update solution should be made aware of rev locks (e.g. in an image header file) and fail out of the firmware update if the rev lock condition is not met.

[FW.CON.02] Firmware update solution should provide a method for cancelling a firmware update that is currently in progress.

In the event a user wants to disrupt a firmware update in progress, the firmware update solution must allow for a graceful exit and not impact the running firmware.

[FW.CON.03] Firmware update solution shall provide a mechanism reporting the last known good firmware version for a given device.

Firmware update solution must retain the last known good firmware version that was installed on the device before a firmware update was applied. In the event of a failure during firmware update, the user may want to revert back to this last known version.

[FW.CON.04] Firmware update solution should provide the capability of staging a firmware update ("update roll-out") so it can be applied in the next maintenance window.

In order for a device to start executing a new firmware image, the device will typically need to be reset. This reset is sometimes inherent in the device firmware that supports update and sometimes the reset must be requested. For devices that require a reset request, the firmware update solution shall provide method for notifying the device whether or not a reset should be

done immediately following program. If a user instructs no reset, the new firmware will be staged for next reset/reboot.

[FW.CON.05] Firmware update solution should provide the capability of rolling back ("update roll-back") to a previous known firmware image if a firmware update fails.

In the event the firmware update fails, to avoid bricking the system, the firmware update solution should allow for an automatic recovery path by reverting back to a known good image.

[FW.CON.06] Firmware update solution shall support the ability to retrieve firmware payloads from devices.

In addition to programming a firmware image, the firmware update solution must also have the ability to read back an image from a programmed device in such a way that it can be reapplied to the same or different device.

6. Status Requirements

[FW.STAT.01] Firmware update solution shall provide a means for providing real time status (e.g. % complete, bytes transferred, etc…)

[FW.STAT.02] Firmware update solution shall provide an upgrade status (e.g. success, failure, staged).

Firmware solution shall provide a method to report back to the user the current state of the firmware upgrade process.

[FW.STAT.03] Firmware update solution shall provide a common method for logging status and error messages regardless of device being programmed.

Preferred location is the System Event Log (SEL) in the BMC. Events that should be logged include: firmware update failures, successful updates, indication a rollback occurred, indication a firmware update was cancelled, etc…

7. Terms and Abbreviations

Intentionally left blank.