[White Paper:

Disaggregated Approach to Firmware Upgrade for white box and DDC]

Version 1.0

9/20/2021

EDITOR(S):

[Tuan Duong, AT&T]

[Mysore Shivakumar, AT&T]

[Prasanna Ramachandran, AT&T]

Executive Summary

Within the Open Compute Project (OCP), disaggregation of software and hardware has created opportunities of open ecosystem between software and hardware vendors and users. This white paper proposes a framework for handling firmware upgrades and advanced diagnostics on white box hardware following the theme of disaggregation that is still acceptable to a service provider environment.

Table of Contents

Introduction

As a white paper contribution to the Open Compute Project (OCP), this document proposes a framework and methodology for upgrading firmware and supporting advanced diagnostic on white boxes within the disaggregated hardware and software ecosystem, especially for the service providers use cases. It will discuss the key requirements from the service providers operational team. It highlights the challenges for the ecosystem NOS vendors, complexity of firmware release lifecycle for the ODM and how this approach is a reasonable compromise that supports the goals of OCP for disaggregation and open ecosystem.

Disaggregated Firmware Update Discussion

Description of white boxes and disaggregation
Figure 1 illustrates the high-level concept of white boxes technology and disaggregation.

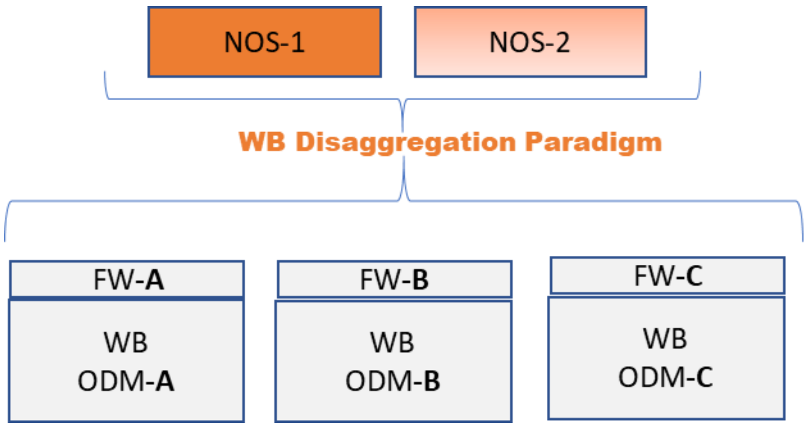


Figure1: WB Disaggregation Ecosystem

White boxes are commoditized hardware built on merchant silicon to a specification designed to address a set of needs. The hardware only becomes useful when it is paired with a software (NOS – Network Operating System) that will exercise the capabilities in the merchant silicon to meet the features required for specific use cases. When white boxes and disaggregation are implemented correctly, it will open the door to opportunities for hardware reuse which will increase volume and lower hardware development and manufacturing costs through NOS competition and innovation.

However, there are new challenges that need to be solved in the white boxes/disaggregation paradigm in order to make it attractive to the ecosystem for the Service Providers, ODM, NOS, and PPI partners.

Background

In 2019, AT&T began working with Broadcom and ODMs and NOS partners and contributed specification of DDC concept to OCP for building larger routing systems using white box and disaggregated hardware and software paradigm.

1. The first implementation of the DDC includes hardware from 3 manufacturers: UfiSpace, Edgecore and HPE
2. In addition, the same set of hardware was targeted for different use cases in the AT&T where each use case has different operational teams marching to different business needs and requirements/schedules.
3. AT&T also worked with two different NOS partners, DriveNets and Cisco, to develop the features for the DDC system relevant to the different use cases. Each NOS vendor has different operating environment and constraints.

Challenges

1. At the time, each hardware manufacturer supports a number of methodologies for upgrading firmware of different hardware components on their white boxes. For examples, for some components like CPLD, the ODM developed the firmware so the source code is available. For other components, like i210 NIC from Intel, the ODM can only obtain binaries for mainline operating system branches.

2. Operational teams within AT&T have become accustomed to operating OEM based networking equipment and expected a similar level of performance and simplicity. For example, all OEM networking equipment in the network today offer seamless upgrade of the firmware from the NOS. This means existing operational tools and methodologies have been optimized to work in this manner. If white box/disaggregation requires backing out to ONIE to upgrade certain firmware components in certain scenarios, this is not acceptable to the operations teams. New method should be as transparent as old CLI method of upgrade in the OEM systems. Minimum number of operator-initiated reboots with minimum downtime is a firm requirement from operational team.

3. The challenges for the NOS vendors were primarily business model and software infrastructure. The business model for one NOS partner at this point is purely NOS for white boxes. The other NOS partner, by contrast, has a large existing OEM business and is expanding to the white boxes/disaggregation opportunities. Furthermore, one NOS is built on a mainline open-sourced Linux branch whereas second NOS is built on a commercial/private Linux branch. As such, it is crucial that the white box/disaggregation model does not present an ever-growing development and support problem as the NOS supports more and more white boxes from different ODMs.

Problem Description

Figure 2a and 2b show a block diagram of the hardware components on the UfiSpace J2 white box (which has been contributed to OCP) that requires some firmware to run. Over the lifecycle of the white box in operation, it is inevitable that certain components will need firmware upgrades. The firmware upgrade is accomplished through the different utility programs as depicted in the yellow boxes.

Traditionally, the NOS would integrate those utilities into the NOS to support firmware upgrade control from the NOS as shown in Figure 3.

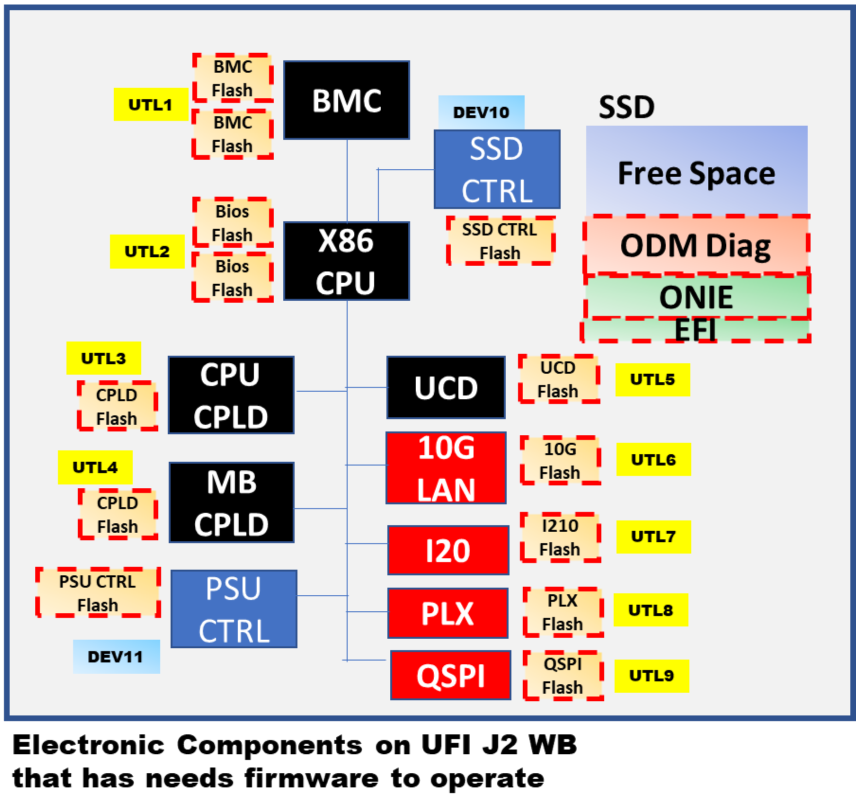
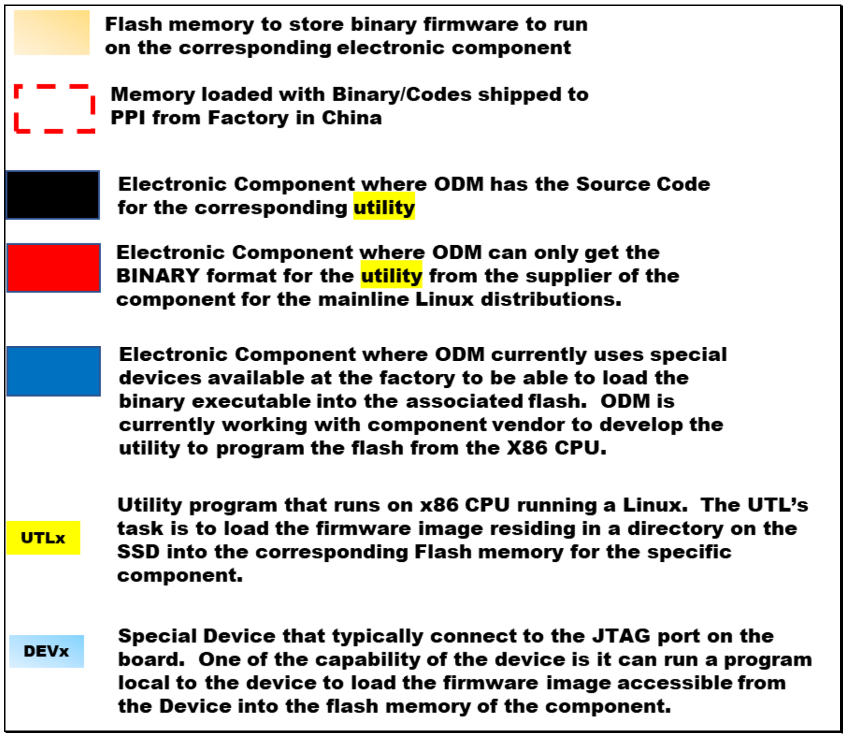


Figure 2a: Electronic Components on UfiSpace J2 WB requiring firmware



🟨	Flash memory to store binary firmware to run on the corresponding electronic component
⬚	Memory loaded with Binary/Codes shipped to PPI from Factory in China
⬛	Electronic Component where ODM has the Source Code for the corresponding utility
🟥	Electronic Component where ODM can only get the BINARY format for the utility from the supplier of the component for the mainline Linux distributions.
🟦	Electronic Component where ODM currently uses special devices available at the factory to be able to load the binary executable into the associated flash. ODM is currently working with component vendor to develop the utility to program the flash from the X86 CPU.
UTLx	Utility program that runs on x86 CPU running a Linux. The UTL's task is to load the firmware image residing in a directory on the SSD into the corresponding Flash memory for the specific component.
DEVx	Special Device that typically connect to the JTAG port on the board. One of the capability of the device is it can run a program local to the device to load the firmware image accessible from the Device into the flash memory of the component.

Figure 2b: Legend to understanding conventions used in Figure 2a.

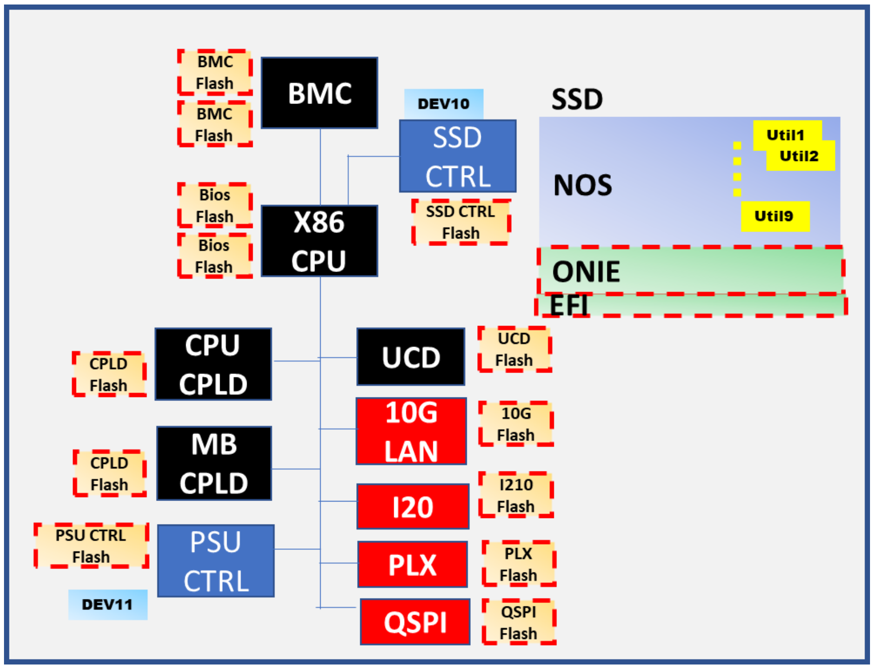


Figure 3: Traditional Approach – Integrate Utilities into NOS.

Issues with Traditional Approach

The traditional approach is to integrate these utilities into the NOS. These utilities will exist as part of the NOS in the NOS partition on the SSD. When the white box is in operational mode, the NOS can invoke these utilities while the NOS is in control of the white box.

There are a number of challenges in the traditional approach:

1. The issue is for some components, the source code of the utilities is available while for others, the utilities are available only in binary format for limited number of mainline OS/Linux distributions. The binary format is a challenge for NOS that are not built on mainline OS distributions to integrate and support those utilities.

2. In a disaggregated model, the problem can grow in complexity for the NOS if those utilities need to be integrated into the NOS. As the number of supported white boxes grows, the number of utilities and the versioning of each utility supported can become unmanageable.

3. The reverse problem is also present for the ODM as the same white box can work with a number of NOS providers where each NOS providers require different versioning support.

4. Potentially, there can be interdependencies of the versions of the components on white box which dictates a certain order/sequence that the firmware needs to be upgraded. This sequence can change depending on the issue/bug at hand. The ODM of the white box would be the best source to know these peculiar details. However, because the firmware upgrade has been integrated into the NOS, it is now the responsibility of the NOS to sequence these upgrades correctly or provide complicated instructions to operator to sequence these upgrades.

5. Finally, for the DDC version1 implementation, this issue is multiplied by 3 because the implementation involves hardware from three different manufacturers: HPE, Accton/EdgeCore, UfiSpace.

Workable Compromise

A compromised is being proposed in this white paper to address issues 1-5 listed above as well as address the existing requirement from the operations team of keeping the tools and procedures for upgrading white boxes solutions on par with the OEM experience. This is illustrated in Figure 4.

The framework and methodology consists of the following:

1. ONIE-based Multi-Firmware Updater implementation from ODM for the white box. This is an ownership of the ODM on how to best implement this Multi-Firmware Updater for the particular white box in the particular situation. This will give ODM freedom to implement to address any special sequencing required. The requirement is that this updater will be invoke using the ONIE-FW update framework.

2. A specification for meta-data file to exchange information between ODM Multi-Firmware Updater and NOS or ODM Advanced Hardware Diagnostics and NOS. This meta-data file becomes especially useful in a DDC system where white boxes from different ODMs are operating by under a NOS. This will enable to NOS to provide a consistent experience to the operator when reporting the status of the different hardware. The detailed specifications of this meta-data file is being submitted to OCP separately.

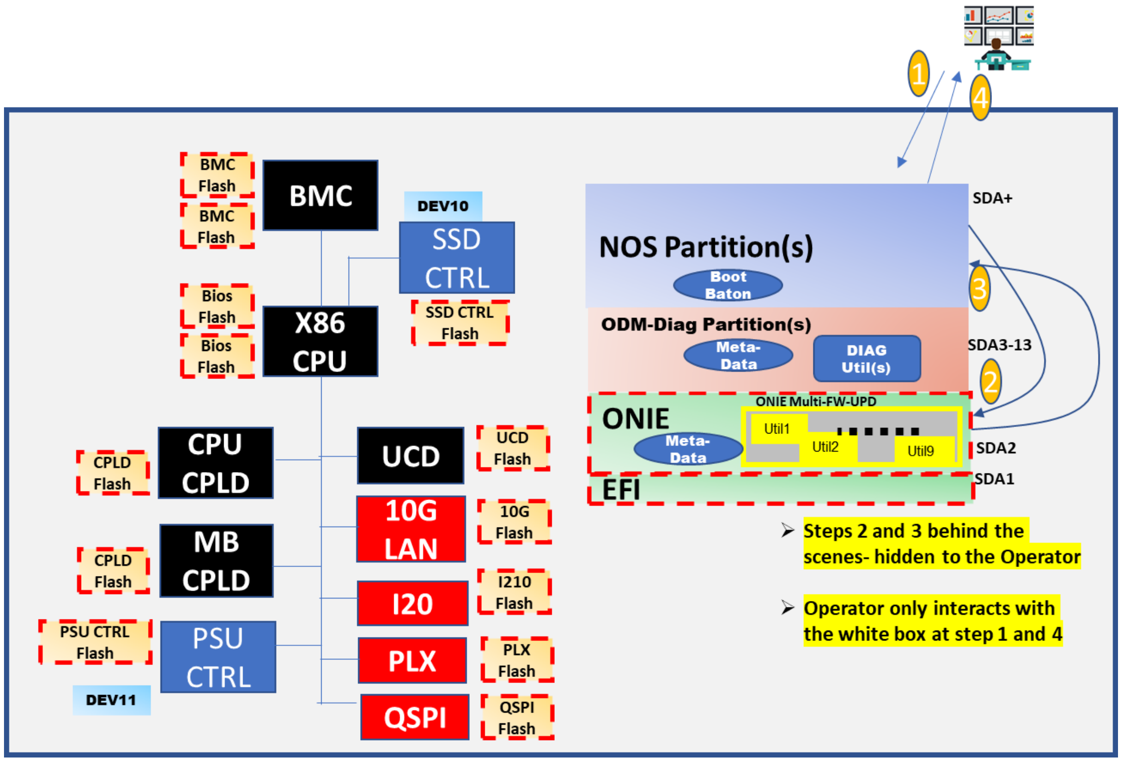


Figure 4: ONIE Multi-Firmware Updater Relay Boot approach

3. A relay approach to upgrading firmware which involves passing "Boot" baton between the NOS and Multi-Firmware Updater which resides in the ONIE partition. Since the NOS no longer integrates the utilities, the firmware update process will involve rebooting the white box back down to ONIE partition or ODM Diag partition and run the appropriate utilities from there. Once the utilities complete their tasks, then the Next-Boot flag is set back to NOS and a reboot is issued so the white box will reboot back into NOS. When the NOS comes online, it checks to see if it came online as following an ONIE Multi-Firmware Update or an ODM-Diag and takes appropriate action to gather log information and information from meta-data files. From the operator point of view, the operator is initiating just one

reboot. While the system in operating under the ONIE-FW update mode, it can reboot the white box as many times as needed before it passes the control back to the NOS. The key is the Multi-FW Updater must keep a local log file with sufficient details in case some error occurred so that we can try to analyze the root cause using the log.

4. ODM-based implementation for advanced hardware diagnostics – OPTIONAL. In some use cases, it is highly desirable to have advanced hardware diagnostic tools from the ODM to troubleshoot and isolate hardware problems or capture those transient hardware problems as they occurred in the field environment. Many times, these issues are not reproducible once the hardware is shipped back to the labs for testing and analysis. Though this is not part of firmware update, it uses the same concept "relay -booting" so it is being proposed as an optional capability. Again, the ODM-Advanced diagnostic needs to keep a local log file with sufficient details for root cause analysis in case of errors.

5. Storage partitioning convention – outlines the partitioning of the storage in a more dynamic/flexible manner versus hard-coding to accommodate the co-existence of the ODM Partition(s) after the NOS is installed on the white box. Traditionally, in the ONIE-NOS installation procedure, the NOS typically erases all partitions above the ONIE partition before installing itself. This is understandable for security reasons. This specification proposes that the NOS leave all partitions intact and create additional partitions into which to install itself.

Addressing Security Concerns

White box and disaggregation deployment in a service provider environment like AT&T faces a different set of requirements than that of a Cloud Service Provider. For one, service provider has thousands of smaller central office locations spread out over the country where most if not all are now operating in lights-out mode, that is no staff on-site. This among other requirements made it necessary to employ a PPI, Post Production Integrator, to "prepare" the white box before it is shipped to the central office for installation.

One of the key step in "preparing" the white box is to reimage all the firmware components residing on the white box, the CPU and BMC BIOS, and erasing and reimage the SSD onboard to the base-lined versions that has been certified for a particular use-case. Recall that an advantage of white box/ disaggregation is the freedom to use the same hardware with different NOS for different use cases. Each use case may be marching on different timelines and firmware release deployments.

Therefore, PPI step is necessary to address potential security concerns of the white box during transit from factory to PPI and to prepare the white box for different use cases. At the PPI, where the trust-establishment begin, is where the ONIE-NOS installation will take place for the white box after the SSD has been erased and reimaged with the ODM-Diag partition(s).

Thus, the NOS installation script behavior under the ONIE-NOS menu needs to be adjusted to accommodate the co-existence of the ODM-Diag partition(s). NOS installation script should support a case switch to not delete the ODM-Diag partition(s).

SSD-Partition Numbering

As hinted previously, the ODM-Diag partition(s) can be implemented in multiple partitions. Implementation will also be dependent on specific ODM-implementations. Currently, at least one partition, SDA3 is needed. However, in the future, it may be necessary to implement the ODM-Diag over multiple partitions. It is not clear at this point what the future implementation will look like.

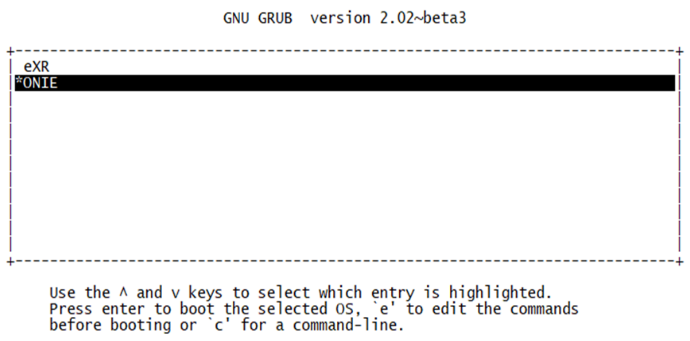
The issue is how the NOS-installation will take place? Is the NOS hard-coded to install in SDA3 or it can adjust appropriately. There are two recommendations:

1. NOS installs on next available partition. For example,
   - if the ODM-Diag takes up SDA3, then NOS installation starts at SDA4.
   - if the ODM-Diag takes up SDA3-SDA7, then NOS installation starts at SDA8.

2. Fixed Allocations:
   - Reserve SDA3-SDA13 for ODM
   - SDA14-SDA30 for NOS

ODM-Partition Installation

ODM-Diag Partition is optional. So it is recommended that the way to access the ODM-DIAG partition should be accessed through the ONIE-Sub menu. This way, the boot behavior for white box does not change. Figure 5a, 5b below illustrate this concept.

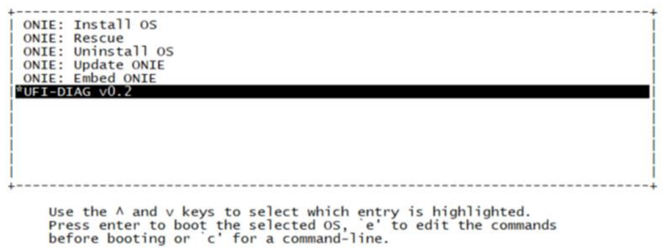


```
                    GNU GRUB  version 2.02~beta3

+----------------------------------------------------------------------------+
| eXR                                                                        |
|*ONIE                                                                       |
|                                                                            |
|                                                                            |
+----------------------------------------------------------------------------+

     Use the ^ and v keys to select which entry is highlighted.
     Press enter to boot the selected OS, `e' to edit the commands
     before booting or `c' for a command-line.
```

Figure 5a: NOS-ONIE boot option does not change



```
+----------------------------------------------------------------------------+
| ONIE: Install OS                                                           |
| ONIE: Rescue                                                               |
| ONIE: Uninstall OS                                                         |
| ONIE: Update ONIE                                                          |
| ONIE: Embed ONIE                                                           |
|*UFI-DIAG v0.2                                                              |
|                                                                            |
+----------------------------------------------------------------------------+

     Use the ^ and v keys to select which entry is highlighted.
     Press enter to boot the selected OS, `e' to edit the commands
     before booting or `c' for a command-line.
```

Figure 5b: ODM-Diag Boot Grub is one additional option under ONIE.

High Level Requirements on ONIE Multi-Firmware-Updater

This section list key features that the Multi-Firmware-Updater need to support (MUST).

Firmware Licensing

ODM will handle all the licensing necessary to upgrade all the firmware on the white box when the Multi-Firmware Updater runs.

Prerequisites Checking and Sequencing Logic

The Multi-Firmware Updater is responsible for prerequisites checking to insure smooth error-free firmware upgrade process. The logic and sequencing of the upgrade of the components on the white box is under the control of the Multi-Firmware Updater. The Multi-Firmware Updater logic can be adjusted over time to yield the best and most reliable performance.

During the prerequisites checking, the normal behavior is to skip the firmware flashing process if the component is already at the designated version/revision level. This would speed up the firmware upgrade in the field for better performance.

FORCED Upgrade Switch

As described earlier, for the service provider environment, when the PPI is employed, the PPI will use this Multi-Firmware Updater force flash all the firmware images on the white box even if it is the same version. This is to address security concerns discussed earlier of tampering during transit from the factory to the PPI. Thus, the Multi-Firmware Updater needs to support the FORCE switch so that it will flash the firmware even if the component is already at the correct version/revision level.

Local Log File

The Multi-Firmware Updater normally outputs very detailed information to the console what is happening during the upgrade process. The Multi-Firmware Updater needs to keep this same level of screen logging in a local file that will be accessible to the NOS once the upgrade process is complete. In most cases, the upgrade process will be successful. This log file will be useful in failure-analysis scenarios. As the ONIE-Partition is limited in size, the local log file implementation should be done in a controlled fashion.

For the ODM-Diag partition and advanced diagnostic scripts, a local log file with sufficient details also needs to be kept on the ODM partition when the advanced diagnostic tests are run. The NOS should be able to access this local log file once the system is booted back into NOS.

CI/KGI for BIOS and BMC flash1 and Flash2

Many white boxes have been designed with dual BIOS flash and dual BMC flash. For the BIOS, these are called CI (Current Image-Flash1) and KGI (Known Good Image-Flash2). For the BMC, these are referred to as flash1 and flash2. The normal operation is to always try to boot from flash1 first. If for some reason, flash1 fails or is corrupted, then the system will try to boot from flash2 which is acting as a backup.

When there are two flash, the following issue comes up.

1. Should both flash run the same version
2. Should one flash1 run one version and flash2 run a previous version

After much debate and trade-off analysis, it is felt that to keep things simple, both flash should run the same version. As such, the Multi-Firmware Updater by default must install the same version of firmware on the dual flash of the BIOS and BMC on white boxes that support such configuration.

In case there is come particular use case in the future where the different versions are needed, then the Multi-Firmware Updater can be enhanced to support that capability with a switch (similar to the FORCE switch) during invocation.

Appropriate Use Case Discussion

As hinted earlier, the Multi-Firmware Updater approach may not be appropriate in all use cases. The downside of this approach is the operator will be able to access the white box via inband methods while the firmware upgrade process is underway. For this duration, the operator is “blind”.

For service provider environment like AT&T, these upgrades are typically performed during a maintenance window where affected customers have been notified and traffic has been diverted. In these use cases, this approach is acceptable.

In the mobility backhaul environment where there are tens to hundred of thousands of devices in the network, this approach may not be suitable. The traditional approach of integrating the firmware update into the NOS may be the way to go. Of course, this would be a higher development cost solution. That is really the trade-off.

Explanation of Meta-Data File for DDC.

The objective of this white paper was to propose a disaggregated approach to upgrading firmware on white boxes when used as a single white box as well as within the DDC cluster. The key difference between a single pizza box and a DDC cluster lies in where the “NOS brain” is running. In the single pizza box, the NOS brain is running on the white box. In a DDC cluster, the “NOS brain” is running on the dedicated cluster controller, which can be a COTS server.

In a DDC, the operator through the "NOS brain" can initiate firmware upgrade on a number of white boxes simultaneously. Thus the "NOS brain" needs to be able to keep track of individual white box upgrade activity/status and report back to operator. This is the primary purpose of the specification of the meta-data file between the NOS and the Multi-Firmware Updater which the Multi-Firmware Updater also need to support.
The specification of the meta-data file is being addressed in a separate white paper or specification being championed by DriveNets, Accton, UfiSpace, and Delta Networks for contribution to OCP as well.

Conclusion

This white paper proposed an approach to do firmware upgrade on white box and white boxes within a DDC that can be implemented in a disaggregated fashion from the NOS. This will reduce the burden on NOS development and will give ODMs more opportunities to put more value into the firmware and tools for their products. It is hoped that this approach will be more widely accepted by the ecosystem because it provides a cleaner demarcation of responsibilities and expertise between NOS and ODM while still meeting the performance requirements and objectives for the service providers and PPI for the majority of network use cases.

Glossary

OCP – Open Compute Project

CI- Current Image

KGI – Known Good Image

ODM – Original Design Manufacturer

OEM – Original Equipment Manufacturer

NOS – Network Operating System

PPI – Post Provider Integrator

DDC- Disaggregated Distributed Chassis

References

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

Appendix A. [Title]