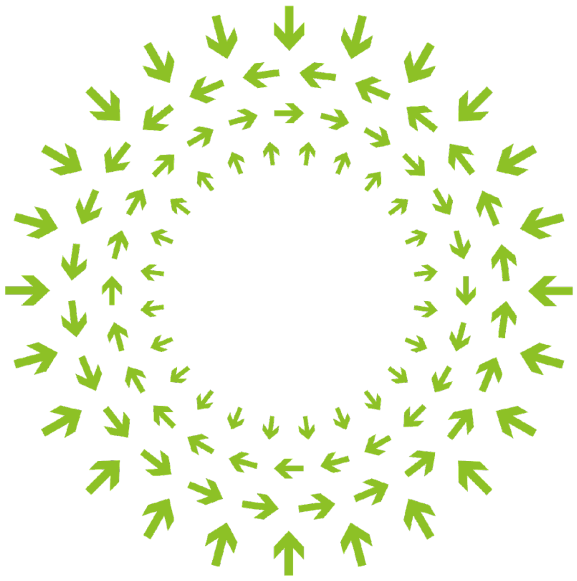

OPEN

Compute Project

DNX-based Distributed Disaggregated Chassis Routing System Evolution (V2): Specification of Hardware and Capabilities

Rev 2.0

Author: Shivakumar Mysore, ms3158@att.com
Author: Rehan Karim, rk2717@attcom

Table of Contents

1. License

1.1. OCP CLA

Contributions to this Specification are made under the terms and conditions set forth in Open Compute Project Contribution License Agreement (“OCP CLA”) (“Contribution License”) by:

AT&T
You can review the Contributor License(s) for this Specification on the OCP website at https://www.opencompute.org/legal-documents. For actual executed copies of either agreement, please contact OCP directly.
Usage of this Specification is governed by the terms and conditions set forth in Open Compute Project Hardware License – Permissive (“OCPHL Permissive”)
Notes:

1) 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.

1.2 Acknowledgements

The Contributors of this Specification would like to acknowledge the following companies for their feedback: **Broadcom, UfiSpace, DriveNets, Accton (Edge-core)**

2. OCP Tenets Compliance

2.1. Openness

The goal of the DDC specification has always been to encourage openness and maximize reuse of common hardware through disaggregation of hardware from software. The specification tries to keep the requirements to a set that can be used in a variety of use cases and environment. Open hardware specifications enable designs from multiple ODMs and software implementations from different NOS vendors.

2.2. Efficiency

DDC solution address efficiency in the following manners:

- New white box designs can quickly leverage new silicon technology that offers higher performance at lower power and cost per bit.
- White box designs limited to [1RU to 3 RU] offers predictability to Rack/Cabinet designs to scale-out a large system.
- Though large system consisting of white box cannot compete with Monolithic Modular Chassis systems in terms of power and space, it makes it simpler for ODM to design. The white boxes can be used in across several configurations which minimizes hardware SKU and maximize reuse and manufacturing scale.
- See section 5.5 for efficiency of DCP3/DCP4 as compared to DCP1/DCP2

2.3. Impact

AT&T deployment experience with DDC and white box and observed significant reduction in hardware cost.

Disaggregation of hardware and software enables AT&T to reuse the exact same hardware with a different NOS vendor to meet the requirements of different use cases.

2.4. Scale

DDC concept is built on modular unitized white box design that can be put together to address a larger scale-out system configuration. With the current silicon technology and white box specifications, it is possible to build a system that range from 4Tbps to 192Tbps. See section 5.5 for scale comparison of DCP3/DCP4 to DCP1/DCP2

3. Revision Table

Date	Revision #	Author	Description
11/30/2021	1.0	Len Ciavattone Prasanna Ramachandran Janet Peng Tuan Duong	DDC Version 2 Spec rev 1.0 ● J2C-WB-VA ● J2C-WB-VB ● 2xJ2C+ WB-VA ● 2xJ2C+ WB-VB ● PTP Timing across DDC
Date	2.0	Shivakumar Mysore (ATT) Rehan Karim (ATT) Shakik Johnson (ATT) Run Almog (Drivenets) Evgeny Sandler (Drivenets) Ray Chang (UfiSpace) Andrew Lui (UfiSpace) Qiyao Zhong (Edgecore)	DDC Version 2 Spec rev 2.0 ● Firmware upgrade ● Security Requirements ● Materials Requirements

4. Scope

This document defines technical specifications for the AT&T Distributed Disaggregated Chassis Routing Systems used in Open Compute Project which was started with the specification titled:
Hardware Specifications and Use Case Description for J2-DDC Routing System
located at:
https://www.opencompute.org/documents/ocp-hardware-specifications-and-use-case-description-for-j2-ddc-routing-system-pdf
For this document, this version 1.1 specification will be referred to as **[DDC-V1.1Spec]**

Since the initial contribution in 2019, the project has moved into successful deployment into production network for AT&T. As a follow-on, AT&T would like to contribute additional hardware specifications and capabilities specification for the DDC routing system that use the Broadcom DNX family of chips as evolutionary steps to keep up the progress of the technology.

A nomenclature in the document name from “J2” to “DNX-based” has been implemented to accommodate for evolution of the chips within the DNX family. DDC is built upon a chip family that implements a fabric-based technology for scale-out of hardware while enabling all the hardware modules to operate in unison as a single routing system. Thus, if there is another chip vendor that offer another capable

fabric-based technology, it is possible that the DDC concept be applied to that technology.

This specification is built on top of the previous specification version 1.1 by introducing two new white box hardware specifications meeting all the common requirements specified in **[DDC-V1.1Spec].** The new hardware components will use new technology and those differences will be covered here. In version 1.1, it was pointed out that PTP timing across the DDC system was lacking due to time constraints and project deliverable. Version 2.0 proposes a mechanism for transporting PTP timing across the DDC system

The organization of this document. The OCP template used for the **[DDC-V1.1Spec]** is different from the current template for OCP Specifications. Since this is built on **[DDC-V1.1Spec]** the approach is copying the requirements specified in the sections of [**DDC-V1.1Spec]** to the new sections in [DDC-V2 Spec]. New details and requirements which were not covered previously are described in detail in the appropriate section of this specification

5. Overview

DDC – Distributed Disaggregated Chassis is a framework to build a large forwarding system that is functionally equivalent to the traditional monolithic modular chassis systems. The **key innovation** is **disaggregation of the software and the hardware**. The hardware components in the DDC are made up of white boxes of line cards, fabric, compute, and management connectivity. The **software** is what makes it possible for the white boxes to function **as a single forwarding system**. DDC-V1.1 Spec covers this in detail. Table below summarizes the progress of the hardware specifications for the DDC. Currently DDC specifications leverages the Broadcom DNX Fabric technology. In the future, the DDC concept can also be applied to other merchant silicon technologies that offer equivalent fabric capabilities for scale-out.

DDC-v1	DDC-v2	Function	Description	ODM SKU/Model	DriveNets NOS	Cisco NOS
DCP100 ==>	DCP1	Line Card	40x100G + 13x400G Fabric	UfiSpace-S9700-53DX DNI-AGCXD40V1	UfiSpace & DNI	UfiSpace
DCP400 ==>	DCP2	Line Card	10x400G + 13x400G Fabric	UfiSpace-S9700-23D	UfiSpace	UfiSpace
DCF48 ==>	DCF1	Fabric	48x400G Fabric	UfiSpace-S9705-48D DNI-AGCC048	UfiSpace & DNI	UfiSpace
DCF24 ==>	DCF2	Fabric	24x400G Fabric	Accton - ??		
DCM ==>	DCM1	Management	48x(1/10G) + 6x100G	Accton AS-5916-54XL	Accton	Accton
DCC ==>	DCC1	Control	Skylake or Later COTS	COTS Server	HP or Dell Servers	HP DL380P Gen 10
	DCP3	Line Card	36x400G + 40x400G Fabric	UfiSpace-S9710-76D	UfiSpace	TBD
	DCP4	Line Card	64x25G + 12x100G + 6x400G Fabric	UfiSpace-S9701-82DC	UfiSpace	TBD
	DCCM1	Ctrl+Mgmt	64x25G + 12x100G + 6x400G Fabric	UfiSpace-S9701-82DC	TBD	TBD

Figure 1: ODM SKU Model NOS vendors

5.1. DDC-RS Configuration Sizes

With the specified DCP and DCF components, it is possible to systematically build DDC-RS to different scales with differing fabric / oversubscription requirements. To reduce complexity and variations, the following configurations are recommended and proposed fabric and management interconnect. The details are described in the excel workbook: “ATT-OCP-J2_DDC_ConfigurationsConnectivity.xlsx”.

Configuration	Scale	Sample Deployment	Elements
Stand Alone	4-Tbps	Stand Alone	{1-DCPx}
Small-1	16-Tbps	1 Cabinet	{1-DCF48, 2-4 DCPx, 2-DCM, 2-DCC}
Small-2	32-Tbps	1 and 1/2 Cabinets	{2-DCF48, 2-8 DCPx, 2-DCM, 2-DCC}
Medium-1	96-Tbps	5 Cabinets	{7-DCF48, 2-24 DCPx, 2-DCM, 2-DCC}
Large-1	192-Tbps	9 Cabinets	{13-DCF48, 2-48 DCPx, 4-DCM, 2-DCC}

Figure 2: DDC-RS Naming Cluster

5.2. DCP3 - 36/32-400G NIF + 40x400G Fabric (New)

DCP3 is a 2xJ2C+ white box design that targets both the service provider WAN and DCI of cloud service provider use cases.

- DCP3 is designed to support line rate MACSEC on every port at 400G.
- The OP2 TCAM is a factory installable item that service provider with large route scale use cases will need.
- In DCI, OP2 may not be needed so CSP desire more NIF instead. The design includes internal MUXes that make this option achievable with a single design.

5.2.1 DCP3 Key Requirements

- The NIC chip used to support the 2x10G SFP+ DDC Mgt connection must be class-C or better.
- Class-C NTM module with Stratum 3E OCXO to support Class-C timing is required for this design.
- This box can be 2RU or 3RU in physical design. The option depends on how much ZR/ZR+ optics need to be supported in this box.
- For the DDC specification, a single DCP3 only need to support ~50% of the NIF with ZR/ZR+ optics

- Cross FABRIC Serdes routing as illustrated in section 5.2.3 is needed to operate in standalone mode and DDC fabric scale-out mode.

Feature	Description
Network	32/36x400G QSFPDD (dependent on OP2 Option)
Fabric	40x400G QSFPDD
Craft	RJ45 Serial Console Micro USB Serial Console USB3.0 Type-A
NM	1x10/100/1000 RJ45 Mgt 2x10G SFP+ DDC Mgt (PTP-Support)
Timing	SMB-10Mhz In/Out SMB-1PPS In/Out
NPU	BRCM 2xJ2C+ (BCM88850 or BCM88851) (SKU Option)
Route Scale	BRCM OP2 BCM16K x 4 (Factory Option)
CPU	Daughter Board: Skylake-D Generation or Later
BMC	Aspeed AST2400
Synchronization	Stratum 3E OCXO, Synch-E IEE 1588V2, T-BC/OC, T-TC, Class-C
LED	System Status LED Per Port Link + Act LEDs FAN Status LED PSU Status LED 2Digits 7-Seg Beacon LED
AirFlow	Redundant Hot Swap Fan, Port to Power (F2B)
PSU	Redundant Hot Swap AC or DC
MACSEC	For SKU with BCM88850. Linerate MACSEC on all 400G Network Ports. BCM88851 does not support MACSEC.

Figure 3: Key Figures for DCP3

5.2.2 DCP3 High Level Block Diagram

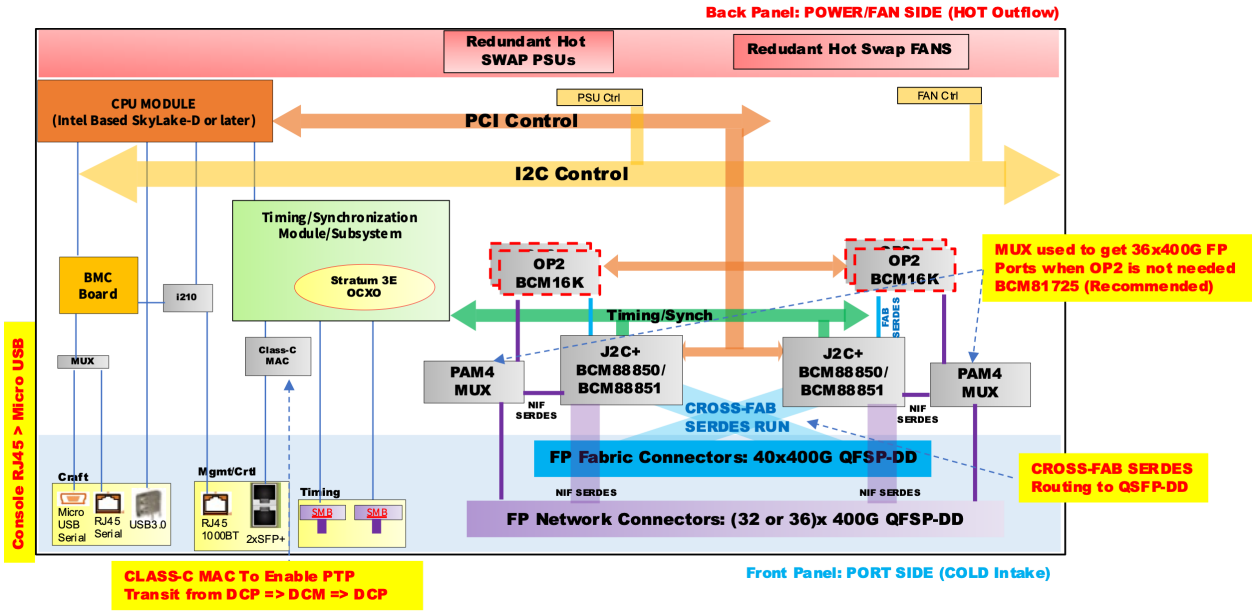

Figure 4: DCP3 High Level Block Diagram

5.2.3 DCP3 Fabric Serdes Routing Considerations

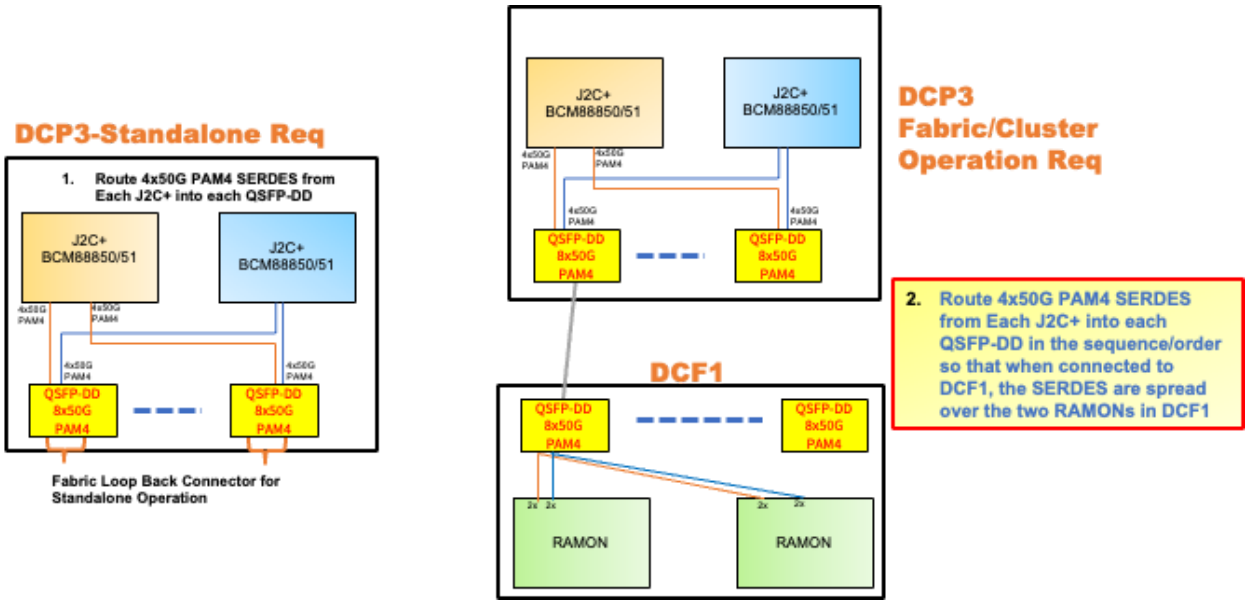

Figure 5: Illustration of Cross Fabric Serdes Routing in DCP3 Design

5.3. DCP4/DCCM1 - (64 SFP28 & 12 QSFP) NIF + 6x400G Fabric (New)

- DCP4 is a 1xJ2C white box design that offers a more efficiency when 1G/10G/25G support is needed in a line card for a DDC deployment.
- In the DDC V1.1 Spec, the DCM1 (AS5916-54XL) is currently in use. The DCP4 can replace the DCM1 and function as DCM2.
- DCP4 can also replace both the DCM1 and the DCC1 to be designated as DCCM1.

5.3.1 DCP4/DCCM1 Key Requirements

- The NIC chip used to support the 2x10G SFP+ DDC Mgt connection must be class-C or better.
- Class-C NTM module with Stratum 3E OCXO to support Class-C timing is required for this design.
- DCP4 needs to support GNSS Receiver and T-GM function.
- 10Mhz, 1PPS, and TOD support are also required for DCP4 design for some use cases that may require them.

Feature	Description
Network	64x(1G/10G/25G) SFP+ 10x100G QSFP [Breakout:4x(10G/25G)QSFP] 2x100G QSFP [Not Breakout]
Fabric	6x400G QSFPDD
Craft	RJ45 Serial Console Micro USB Serial Console USB3.0 Type-A
NM	1x10/100/1000 RJ45 Mgt 2x10G SFP+ DDC Mgt (PTP-Support)
Timing	SMB-10Mhz In/Out (In or Out - Configurable) SMB-1PPS In/Out (In or Out - Configurable) SMA-GNSS Input RJ45-(TOD & 1PPS) (In & Out)
NPU	BRCM J2C BCM88802
Route Scale	BRCM OP2 BCM16K (Factory Option)
CPU	Daughter Board: Skylake-D generation or Later
BMC	Aspeed AST2400
Synchronization	Stratum 3E OCXO, Synch-E, Class-C IEE 1588V2, T-BC/OC, T-TC,T-GM
LED	System Status LED Per Port Link + Act LEDs FAN Status LED PSU Status LED 2Digits 7-Seg Beacon LED
AirFlow	Redundant Hot Swap Fan, Port to Power (F2B)
PSU	Redundant Hot Swap AC or DC

Figure 6: Key Figures for DCP4

5.3.2 DCP4/DCCM1 High Level Block Diagram

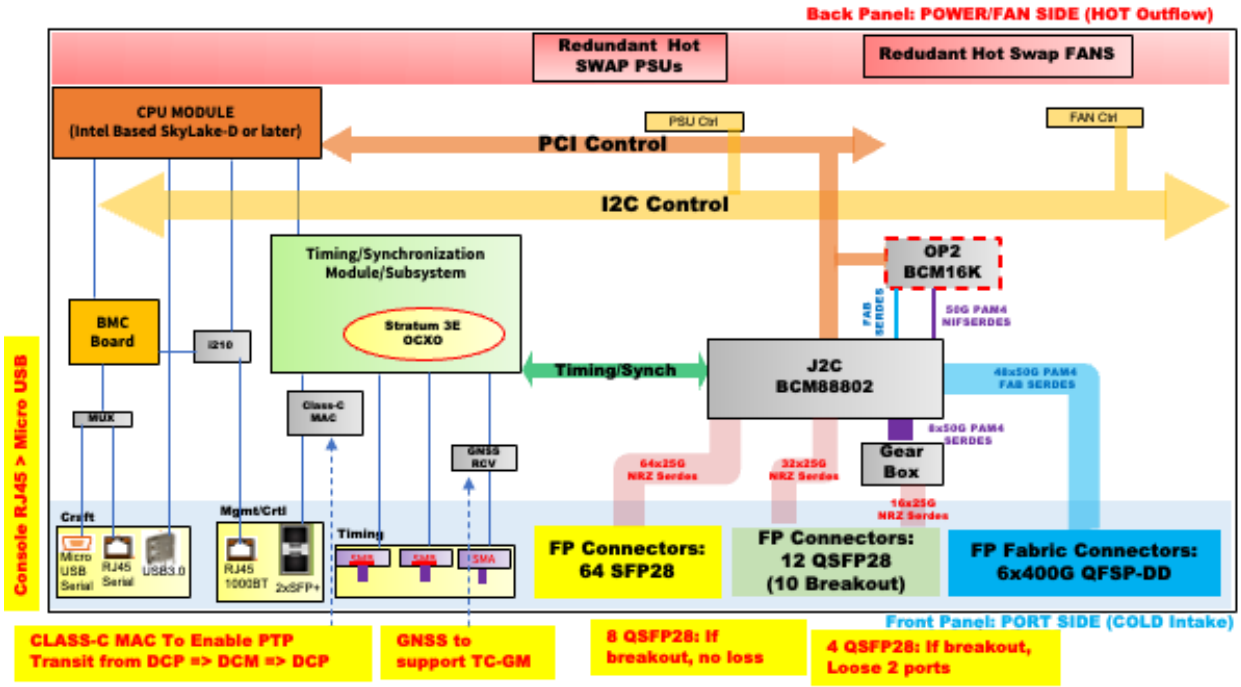

Figure 7: DCP4/DCCM1 High Level Block Diagram

5.4. DCCM1 Use Case

DCCM1 – Concept is to combine the function of DCM1(AS-5916-54XL) and DCC1(HP-DL380P Gen10) into one white box. This can be accomplished using the DCP4 with a Maximum CPU module option:

SkyLake-D 16-Core (present) or IceLake-D 20-Core (future)
4x64G ECC DRAM
RAID0 2x512G SSD

The spreadsheet included here illustrates a possible way to cable using the DCCM1. This cabling plan anticipates future DNX evolution where the DDC will go to 10 DCF2 white boxes instead of the current 13 DCF1 for a large configuration.

DCCM1-Cabling Details

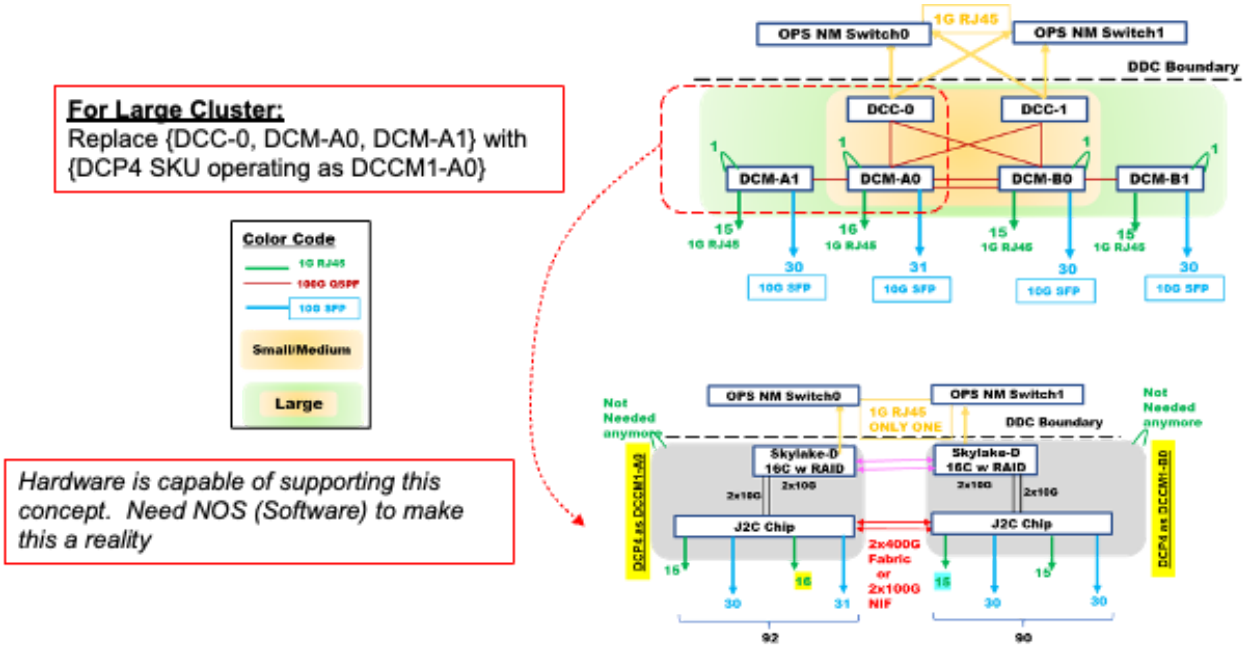

Figure 8: Assessment of Internal NM cabling needs of a DDC for combined DCCM1 white box

5.5. Inserting DCP3 and DCP4 into existing 7-Medium1 or 13-Large1 DCF Deployments

- **One DCP3 = 3*DCP2** in terms of Network ports and Fabric Ports in (2RU, 3000W) versus (6RU, 6000W).
- DCP3 (with OP2) offers same route scale as DCP2 but supports 4X- VOQ over DCP2.
- **2*DCP4 (4RU, 2600W) = 1*DCP1 (2RU, 2000W)** for Equivalent Network and Fabric switching BW.
- DCP4 offers better route scale (~75%) and Queues scale (2x) over DCP1.
- DCP4 offers a more efficient use of resources over DCP1 when more 10G/25G interfaces are needed.
- DCP1 loses 50% of ports when used in breakout mode.

6. Rack Compatibility

The white boxes specified in this need to comply to [DDC-V1.1Spec] which is shown below:

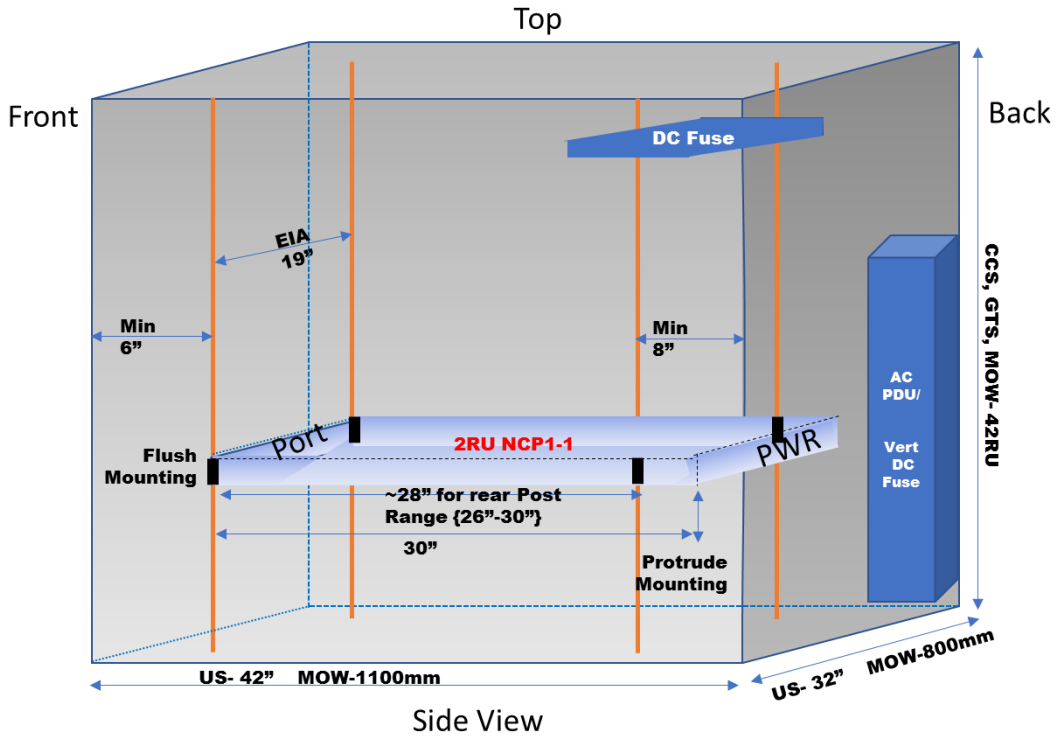

Figure 9: Rack Compatibility

7. Physical and Common Specifications

7.1 Physical Design Constraints

With the proper cabling the following consideration must be made for the physical platform dimension (does not include handles or SFP). All cabling must be front accessible and adhere to AT&T bend radius standards as specified in ATT-TP76300, section J part 2.10.

- **Width:** 19" rack mount EIA cabinet standard with 4 post mounting.
- **Depth:** Maximum 30" depth to allow for Cabling and Power clearance in a 42" deep cabinet.
- Height: 1RU, 2RU(Preferred), Up to 3RU (New to DDC-V2 to accommodate 400G/800G ZR/ZR+ as necessary)
- Airflow: Front to back.
- Access: Front Access for fibers and cables. Rear Access for Power and FAN FRUs.

- Temperature: Range {20C to + 50C} Ambient. Typical 26C.
- Environmental Spaces: AT&T - {CCS, GTS, MOW Tele-Houses}

The objective of the designs for the DDC-RS components is to be able to mount them in the AT&T standard cabinet which are EIA- 19", 42" deep and 42RU high. This cabinet dimensions are used in the three environmental spaces {CCS, GTS, MOW} that is targeted for these use cases. Figure 5 illustrates the typical cabinet and **minimum** mounting clearances required to allow for cabling and power cables and PDUs. For example, the front rails need to be located minimum 6 inches back from the front of the cabinet. For this use case, it may require 7 inches to allow for sufficient fiber bend radius or fiber densities for example. The rear mounting rails need to have 8 inches minimum. For a 30" deep piece of equipment, this would mean the front mounting ears are flushed and the rear mounting rails needs to be able to adjust to 27 inches {= 42-7-8}. That is why the rear mounting rails need to be adjustable from {26" – 30"}.

The significances of the different environmental spaces are the NEBS compliant requirements, more precisely for AT&T, TP76200/TP450 Level1/Level3 requirements. This will be spelled out in more details in another section.

7.2. Common Design Components

The white box design should adopt modularity and reuse where possible to achieve scale and minimalize sparing complexity. The common components in the white box are the following:

- PSU FRU
- Fan FRU
- The Craft, Management, and Console interfaces
- The 7 segment 2-digits LED display/beaconing
- BMC,
- X86 host CPU
- Network Timing Module (New in DDC V2)

7.3 X86 CPU Considerations

For DDC V2 Specification, the x86 CPU needs to meet the following design requirements:

1. Needs to be designed as a modular daughter board that can be reused in multiple white box configurations where CPU cores, DRAM, SSD are factory installable options.
2. Needs to be designed with sufficient signals to be connected to the Main switch board that will allow it to accommodate a couple generations of CPUs. (Current is Skylake-D and future is IceLake-D)

3. Enough signals between CPU daughter board and main switch board to accommodate “PUNT-Path” from MAC to CPU directly.
4. Needs to be designed to accommodate a range of CPU sizes from 4 cores up to max 16/20 cores from. Minimal memory to max DRAM memory.
5. CPU daughter board needs to support RAID-0 SSD.
6. Redundant BIOS Flash
7. TPM 2.0
8. Operation without Coin-Cell Battery for TELCO and with Coin-Cell as other providers deem fit.
9. All NIC must be supported by DPDK (refer to http://dpdk.org/doc/nics).

7.3.1 Trusted Platform Module

The latest Trusted Platform Module (2.0 or greater) shall be used for secure storage of keys and certificates in a hardware chip and is an integral part of creating a Secure Boot environment so that the device cannot be easily taken over, such as by booting from a USB drive.

The design of the TPM must be a factory orderable option. The reason is in certain use cases, it is not desirable to have the TPM installed due to foreign country import/export rules.

This is a future proof design specification because it is dependent on the availability of ONIE secure boot. If TPM was ordered to be mounted, then Initial software releases will operate with TPM disabled in BIOS and use regular ONIE Boot process.

If the CPU board was ordered without the TPM mounted, then the BIOS must support boot up without the TPM present.

7.3.2 Coin-Cell Lithium Battery

Typical X86 design specifies the use of a Coin-Cell Lithium battery to maintain the RTC. However, the presence of this battery conflicts with the TP76200/TP76450 requirements. As such, the Cell Site Gateway Router design does not have a Coin-Cell battery.

However, there is an issue with correct TPM operation, should TPM be activated in the future, without the presence of the battery to maintain the RTC following a power loss. A ticket was documented with Intel and a work around in the BIOS is needed from the manufacturer. This is documented in Intel Ticket # (00260505).

Coin-Cell battery is a factory orderable item. As such, other providers may choose to use the battery.

7.4 Network Timing Module Implementation

DDC-V1.1 specification was lacking a solution to be able to synchronize PTP across the DDC. The new DCP3 and DCP4 specifications addresses this short-coming with a Network Timing Module and a Class-C or better NIC that drives the 10G SFP connection from the DCP to the DCCM1. Figure 7.4 illustrates at a high level how PTP can be synchronized across DCPs in a DDC. This synchronization is only available on the newer DCPs.

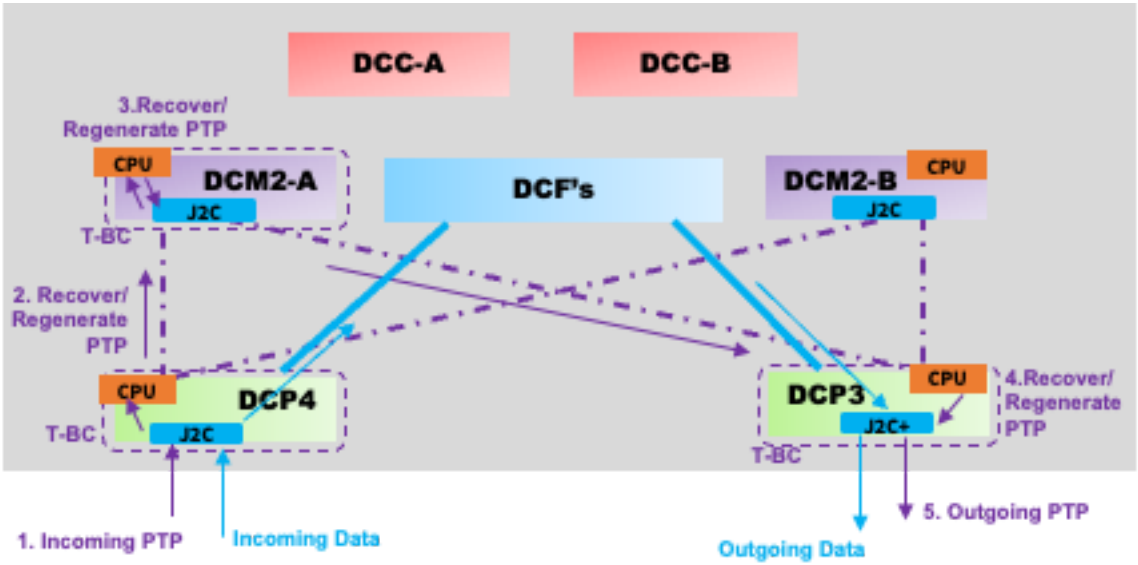

Figure 10: PTP synchronization across DCPs within a DDC

1. DCP3 and DCP4 designs includes a specification of a class C or better NIC for the 2x10G SFP that connects to the DCM in a DDC configuration.
2. The DCM hardware needs to be upgraded to the DCP4.
3. PTP synchronization can only occur between the newer DCPs (i.e., DCP3 or DCP4)
4. In a brown field deployment with older DCP1 and DCP2, careful planning of PTP path is needed to avoid traversing older DCP's

Software is needed to cause each DCP-DCM-DCP to act as a T-BC within the DDC. All new DCP white box specifications for DCC must include the NTM in the design. With the current DNX generation, the PTP timing is transferred across via the DCM. Feature request is already in place for the future DNX generations to transfer the PTP timing across the DCF fabric interface.

8. Thermal Design Requirements

The components used in the J2 DDC-RS needs to meet the following requirements.

- AT&T TP76200 (Issue 20) & TP76450 (v17) for Level 3.
- Copies of this document and general information about AT&T's environmental equipment standards can be found at https://ebiznet.sbc.com/sbcnebs/

8.1. Thermal Shutdown

NEBS/TP76200 compliant equipment should have the ability to be configured to shut down when the thermal threshold is exceeded or continue to operate until the equipment fails completely. Configurable means that the "user" can select the thermal overload behavior. This must be set through software. The default should always be to implement equipment shutdown in a thermal event.

Shutdown means that all non-essential functions of the chassis are powered off and only temperature monitoring capability remain such that, if the thermal event ends, the chassis will autonomously reboot and restore service. One way of accomplishing this is to have the management hardware command the power supplies to shut off their main outputs but maintain an auxiliary power bus that powers the management/monitoring functionality.

9. I/O System

9.1. Craft/Management Interfaces for DDC common I/O system

The following table lists the craft and management interfaces that are needed on the white boxes specified in section 5.

Craft Type		QTY	Purpose
Micro USB Serial		1	Console
RJ-45 USB Serial		1	Console
USB Port		1	USB data access
RJ-45 10M/100M/1G		1	Ethernet OOB Powered on Standby Power rail
10G SFP+ ports		2	For Internal Management Communication for the Modules in a large DDC-RS system

Console:
Only one Serial input can be active for the Console. RJ45 RS232 is higher priority than Micro-USB. RJ45 RS232 will be active if both interfaces are connected by user. The Serial console needs to support default selectable between the BMC or the X86 CPU. (customer-provisioned choice).

OOB Management:
The RJ-45 OOB Ethernet management port needs to be operational even when the system is in the shutdown mode. As such it needs to be designed using the standby power rail. It also needs to provide simultaneous connectivity to the X86 CPU and the BMC. The Intel I-210 NIC is specified to use for the RJ-45 OOB Ethernet Management. Design should allow to shutdown Ethernet OOB access to the BMC as needed.

USB port:
USB port can be used to provide access to external USB drive for initial system setup or system rebuild. However, for security reasons, once the system is up and running under the control of the NOS, the hardware/firmware of the box needs to provide a mechanism to turn off access to this external USB port. This lock needs to persist even after a cold boot, warm boot, other reset mechanism. To unlock this would require authentication to access the utility to set the registers to unlock it.

2x10G SFP+:
These two 10G SFP+ ports provide connections to the X86 Host CPU. This will be used for communication with the Route-Engine Controller of the DDC-RS when this is one component of the larger DDC-RS.

10. Rear Side Power, I/O and Midplane

10.1. Fan Module Specifications

The DDC-RS environment will have rear access. Fans needs to be Redundant, field replaceable and hot swappable. Fans are accessible from rear panel.

10.2. Power Supply Specifications

- DC PSU accept nominal -48V DC Power.
- AC PSU operates from 200-240V 50-60Hz input range.
- The power supplies shall meet the 80 Plus Gold or better for high efficiency.
- There shall be redundant, field replaceable, hot-swappable power supplies.
- The power supplies should support loss of power indicators to facilitate BMC
- Two grounding screws on rear panel of the system.

Table below shows the PSU efficiency under different loads for different 80-Plus standards.

	20% Load	50% Load	100% Load
80 Plus	80%	80%	80%
80 Plus Bronze	82%	85%	82%

80 Plus Silver	85%	88%	85%
80 Plus Gold	87%	89%	87%
80 Plus Platinum	90%	92%	89%

11. Rack Implementation

See Section 6 in this document

12. Mechanical

12.1. Recessed Reset Button Behavior

When the recessed reset button is depressed for less than equal to 10 seconds and released, then this should cause a warm reset of the whole white box.

When the recessed reset button is depressed for greater than 10 seconds and released, then this should generate an interrupt with a vector code to the X86 CPU.

12.2. Silk Screen Recommendations

It is a strong recommendation that the manufacturer choose Pantones, Contrast, and Font Size that MAXIMIZE visibility to the field technicians working in low light, tight spaces, and crowded cabling conditions.

Silk screen should be clear and avoid possible confusion or misinterpretations.

Best is to solicit feedback prior to implementation of silk screen.

12.3. LED Operations Recommendations

Refer to AT&T Hardware Common Systems Requirements for recommendations on system and interface LED colors and operations.

The indicator lamps (LEDs) must convey the information described in 4. The number, colors, and flash behaviors are desired but not mandatory.

LED Name	Description	State
PSU1	Led to indicate status of Power Supply 1	Green - Normal Amber - Fault Off – No Power

PSU2	Led to indicate status of Power Supply 2	Green - Normal Amber - Fault Off – No Power
System	LED to indicate system diagnostic test results	Green – Normal Amber – Fault detected
FAN	LED to indicate the status of the system fans	Green – All fans operational Amber – One or more fan fault
LOC	LED to indicate Location of switch in Data Center	Blue Flashing – Set by management to locate switch Slow flashing – System is in standby state Off – Function not active
SFP- LEDS	LED built into SFP(28) cage to indicate port status	On /Flashing – Port up (flashing indicates activity) Green – Highest Supported Speed Off – No Link/Port down
QSFP LEDs & Breakouts	Each QSFP28 has four LEDs to indicate status of the individual 10-25G ports	On Green/Flashing – Individual 25G port has link at 25G. (yellow for 10G) Green – Highest Supported Speed Amber – Lower Supported Speed Off – No Link/Port down
OOB LED	LED to indicate link status of 10/100/1000 RJ45 management port	On /Flashing – Port up (flashing indicates activity) Green – Highest Supported Speed (1G) Amber – Lower Supported Speed (100M/10M) Off – No Link/Port down

Table 1: LED Definitions (Recommended)

12.4. Port Numbering Specifications

AT&T numbering standard for white boxes starts from zero {0,1,2, ….}, increasing from Left to Right. This applies to Ports, FAN and PSU and other FRU (Field Replaceable Units). Numbering starting from Zero also applies to break out ports which is more dependent upon software implementation as opposed to hardware and silk-screening implementations.

Manufacturer has a degree of freedom for the numbering with respect to vertical grouping. For examples, the following schemes are acceptable.

NOTE: The port numbering illustration shown in these tables does not reflect the actual number ports specified for the DCPs. It is just to illustrate the acceptable numbering scheme.

0	2	4	6	8	10	12	14	16	18
1	3	5	7	9	11	13	15	17	19
20	22	24	26	28	30	32	34	36	38
21	23	25	27	29	31	33	35	37	39

Table 2: 2 grouping numbering schemes: Upper/lower port grouping. Sequentially Numbered Upper grouping then followed by Lower grouping

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

Table 3: 4 rows numbering scheme: Sequentially number each row then move to next row

0	4	8	12	16	20	24	28	32	36
1	5	9	13	17	21	25	29	33	37
2	6	10	14	18	22	26	30	34	38
3	7	11	15	19	23	27	31	35	39

Table 4: 1 Grouping numbering scheme. Sequentially number the ports in a column then move to next column

As shown in the system block diagram, the numbering for the SFP groups of ports should start from 0 and groups of QSFP form factor ports should also start from 0. The table below illustrates this concept.

0	2	4	6	8	10	12	14	16	18		0	2	4
1	3	5	7	9	11	13	15	17	19		1	3	5

Table 5: 0-19 are ports of one physical grouping characteristics (e.g., SFP Pluggable). 0-5 are ports of different physical grouping characteristics (Example: QSFP-Pluggable)

13. Motherboard Power System

13.1 Watchdog Implementation

Design needs to include a watchdog mechanism prevent the system from being stuck in a “hanged” state. This can be done via one or combination of the following:

1. Dedicated watchdog circuitry.
2. Intel TCO watchdog

3. Or some kind of watchdog implementation between BMC and X86 host.

The choice of implementation is left to the manufacturer as long as this is documented and appropriate drivers or mechanisms to leverage this capability from the NOS is provided.

13.2 Internal Glue Logic and Controls

The specification leaves freedom to manufacturer to use any combination of discreet logic/PLD/CPLD/FPGA necessary to implement of the specification. It is recommended to use current practice and provide I2C interface to the Host for control of PSU, FAN, Optics, and any other key components such as reading of registers or erasing and flashing of NVRAM, EEPROM, Flash,… The manufacturer must provide the instructions and drivers to access these components as part of the BSP- Baseboard Support Package, so that NOS development can access and control these components as necessary.

13.3 Dying Gasp Guidance

Dying Gasp is not a required feature for this application mainly because of the environment in which these boxes will be used. They are AT&T or Leased facilities with redundant power feeds and back-up power.

13.4 Supported Optics

The white boxes specified for DDC usage has NIF and Fabric Optics. Fabric Optics are limited to AOC or 400G SR optics. The white box design needs to be able to support optics from {DAC, AOC, SR, FR, LR, ER, ZR/ZR+}. For the {ER, ZR, ZR+} support the design need to support power and cooling for up to 50% optics operating at {ER, ZR, ZR+}. This is the minimum specification. If the design can support more, then it is better.

13.5 Number of MAC addresses and Address Constraints

Each NCP will be configured with a block of 256 MAC addresses.

Each J2 can resolve up to 64 different (38MSB of Mac address) blocks of 1024 (10 LSB of Mac Address). In the DDC Large cluster, the maximum number of NCP is 48. So worst case for Large DDC cluster is 48 different blocks of addresses.

13.6 HW BMC Specification

The system will be designed with Baseboard Management Controller (BMC) to allow for remote lights out operations, management, and access.

The most important requirement for the BMC is that it must be secure. The BMC will be connected to WAN and/or Internet connections. As shown in the System Block Diagram, the BMC has an Ethernet connection which is a shared external connection to RJ45 OOB Management Ethernet Port with the X86 host.

The long-term goal is to use Open BMC when Open BMC supports the required features needed for the operational/business mode in a secure way. In the interim, a commercial BMC implementation is acceptable. Regardless of the open or commercial implementation, the firmware must be capable of disabling this Ethernet access when needed based upon the use case or operating environment.

The following requirements are specified for the BMC.

- Dual Flash memory to support remote reliable in-band BMC Software upgrade.
- Power management: On/Off of control system, Host CPU, and MAC where it makes sense. That is, it does not make sense if a component is powered off which completely cuts off access to the BMC to power it back on. In this case, this component should always be powered on the standby power.
- Temperature monitoring
- Voltage monitoring
- Fan control
- Reset control
- Host CPU boot up status
- Serial number / unique identifier
- Board revision ID
- I2C interfaces to Host CPU, USB, temperature sensors, and voltage controllers.
- Monitoring detect signals – including loss of power from the power supplies.
- Must support Redfish Implementation.
- Must support IPMI 2.0 host mode to provide the following capability via the IPMI interface to allow NOS transition development:
  - ✔ temperature reading and alarms at 3 levels (minor, major, critical) for Processor modules, Chassis, power supply, fans, Broadcom chipset.
  - ✔ status information for fans, power supply, interface modules, processor modules, fan tray

13.7 Detection of insertion and removal of optical/DAC pluggable modules

The White box CPLD or glue Logic design should be able to detect insertion and removal of pluggable optical modules on the NIF and Fabric ports and notify the x86 host with an interrupt and a vector code.

13.8 Resets

RESETS in the Design of all DCP/DCF/DCCM1 white boxes

- Software reset of BMC –
  1. SSH to BMC and issue reset or reboot
  2. Reset BMC from x86 host via IPMI command

- Software reset of x86
  1. SSH into x86 and issue reset or reboot
  2. Reset x86 host from BMC via IPMI command

- Hardware Level reset of BMC via setting on the CPLD on the board
- Hardware Level reset of x86 via CPLD settings.
- Hardware Level reset of x86 or BMC via watchdog timer between x86 and BMC
- Hardware Level reset of the entire white box via 12-volt power rail.

13.9 ONIE

Fulfillment of the ONIE hardware specification as laid out here:
https://opencomputeproject.github.io/onie/design-spec/hw_requirements.html

13.10 BMC Software

ODM needs to start support Open BMC with Redfish implementation with the new DCP3 and DCP4 specifications. Because DDC V1 is currently in deployment with Commercial BMC with IPMI 2.0 is the current implementation, IPMI 2.0 needs to be continued to be supported to give time to work with NOS vendors to migrate to Redfish.

14. Environmental and Regulations

The components used in the J2 DDC-RS needs to meet the following requirements.

- AT&T TP76200 (Issue 20) & TP76450 (v17) for Level 3.

- Copies of this document and general information about AT&T's environmental equipment standards can be found at https://ebiznet.sbc.com/sbcnebs/

15. Environmental Requirements

The components used in the J2 DDC-RS needs to meet the following requirements.

- AT&T TP76200 (Issue 20) & TP76450 (v17) for Level 3.

- Copies of this document and general information about AT&T's environmental equipment standards can be found at https://ebiznet.sbc.com/sbcnebs/

16. Prescribed Materials

The components used in the J2 DDC-RS needs to meet the following requirements. AT&T TP76200 (Issue 20), TP76450 (v17) for Level 3, and TP76201.

Copies of this document and general information about AT&T's environmental equipment standards can be found at https://ebiznet.sbc.com/sbcnebs/

Compliance to material regulation directives should be from accredited labs and meet the following regulatory directives.

- REACH
- RoHS
- WEEE

17. Software Support (recommended)

Please document any software tools used to validate the hardware design and include test and validation using virtual simulation, design decisions based upon digital models, or proof of manufacturability via 3-D tools.

18. System Firmware

The goals of this section are:

- Reduce firmware upgrade complexity of the DDC
- Provide the capability to upgrade the firmware on DDC components independently from the NOS version
- Define clear separation between HW and SW for root cause analysis
- Define NOS and ODM agnostic interfaces

This section includes specifications both for DDC ODMs and NOS vendors.

The Distributed Disaggregated Chassis (DDC) concept allows creating a large-scale network element (e.g., Routers, Switches) based on the interconnected set of hardware devices (aka DDC components). A distributed piece of software (aka distributed NOS) runs on all chassis components. The disaggregated approach assumes that DDC components are manufactured by multiple ODM vendors and distributed NOSs are developed by multiple software vendors. In this way, customers have the flexibility to choose the hardware and software (i.e., NOS) for their DDC from a variety of vendors, dramatically reducing the TCO of the DDC systems.

DriveNets is the first distributed NOS vendor for large-scale routers based on the DDC model (aka cluster). DriveNets DDC NOS controller is a one-stop-shop for all DDC life-cycle management actions including cluster deployment and upgrade. NOS controller allows the user to manage the cluster deployment and cluster upgrade actions

eliminating the need for automation tools development and manual procedures for the following tasks:

- Firmware upgrade on each DDC component
- NOS software deployment/upgrade on each DDC component

As of the time of writing, the largest DNOS GA cluster may include up to 63 DDC components. And there are at least three ODM vendors supported by DNOS for DDC components:

- UfiSpace
- Edgecore Networks
- Delta Networks

As part of the DDC upgrade, NOS controller manages component upgrades in a specific sequence to avoid cluster integrity corruption. For example, first, the NOS controller components are upgraded in the cluster, then all data plane components and fabric components, and finally, the internal ethernet switches are upgraded. This sequence may change according to design considerations. NOS controller manages the firmware and software upgrades for each component in the cluster.

18.1 Problem Statement

Each DDC component includes multiple sub-components that runs firmware. Occasionally, a firmware upgrade may be required (e.g., for bug fixes or performance optimization). However, each firmware version may require a different firmware upgrade sequence. For example, a component that includes BMC, BIOS, and CPLD sub-components may require the BIOS to be upgraded first, then all CPLDs, then the restart of the device and finally BMC upgrade. Moreover, such a sequence may change as firmware versions advance due to implementation considerations. Consequently, each ODM vendor may require a different firmware upgrade sequence. Therefore, implementing a firmware upgrade sequence for all ODM vendors as part of NOS upgrade logic may be challenging.

In addition, for an upgrade planning purposes, NOS should provide to user the following information about upcoming upgrade prior to entering the maintenance window. Such information is estimated list of upgraded firmware components, estimated firmware upgrade time and firmware upgrade pre-requisites validations. Due to the factors described above, NOS should be aware of all ODM vendors and firmware versions details to provide firmware upgrade estimations and pre-requisite validation. Maintaining this information as part of NOS is not practical.

This specification defines a method that decouples the firmware upgrade sequence logic from the NOS while allowing NOS to manage the firmware upgrade on each DDC component as part of the cluster upgrade process as mentioned in section 18.2. Moreover, the defined method may be used for firmware upgrade by any other management systems besides DDC NOS controller throughout all stages of the white

box device provisioning. Starting from the ODM release, going through the devices setup at VAR site and ending with device deployment at the customer site.

18.2 Firmware upgrade process

To detach the firmware upgrade logic from the NOS, an ONIE firmware installer method is used as described in section 18.3. The good thing about ONIE is the fact that it can be used for all upgrade activities on the device besides firmware (e.g. NOS, HW Diagnostic OS). This makes ONIE a universal tool for upgrading the device.

The ODM releases a firmware upgrade package as a tarball. The tarball includes the following:

- ONIE firmware installer -
    - Upgrade script – a script that defines the firmware upgrade sequence for the ONIE
    - Upgrade tools – tools that are used to upgrade firmware components
    - Firmware images – a target version for upgraded firmware components
- Metadata file – information for the NOS that describes the ONIE firmware installer as follows:
    - List of target versions for firmware upgrades (for each firmware component)
    - Platforms, hardware revisions, and hardware builds supported by the installer
    - Firmware upgrade prerequisites (e.g., ONIE version)

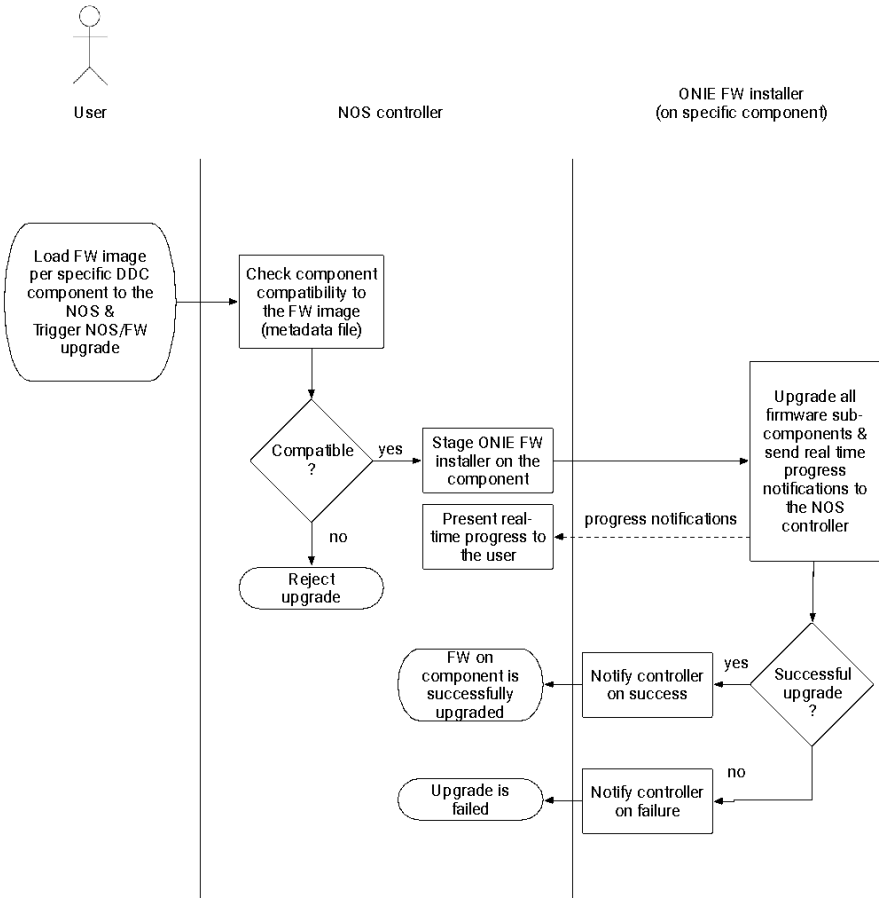

Figure 11: Firmware upgrade process in DDC

The above defines the ONIE-based firmware upgrade process in the DDC: the user receives the firmware image (tarball) from the ODM, loads the tarball to the NOS, and triggers the NOS upgrade (or firmware upgrade on a specific DDC component).
If NOS upgrade includes firmware upgrade on specific DDC component, the NOS verifies the component's capabilities with the firmware image (using the information provided by the image metadata). If the component is compatible (i.e., platform type, hardware revision, ONIE version on the component), the NOS triggers the firmware upgrade by staging the ONIE installer on the DDC component. Otherwise, the NOS rejects the upgrade process.

The same process is followed on all DDC components according to the NOS defined sequence if the whole DDC is upgraded. I.e., first on all NCCs, then on all NCPs and NCFs, and finally on all NCMs.

When the ONIE boots on the staged component, it checks which firmware components need upgrading. A component running the same firmware version as the target version in the package will not be upgraded. Otherwise, the ONIE firmware installer will upgrade the component even if the target version is older than the one already installed. (See section 18.2 for details on the steps performed by the ONIE firmware installer). During

the upgrade process, the ONIE firmware installer sends real-time progress notifications to the NOS controller. The NOS controller presents the real-time progress for each component to the user.

If the ONIE firmware installer fails to upgrade at least one firmware on the DDC component, it notifies the NOS controller of upgrade failure; otherwise, it notifies the NOS of a successful upgrade. If the NOS receives a failure notification on at least one component, the upgrade is considered failed. It is assumed that the user will replace the failed device according to ODM recommendation in the field.

18.2.1 Firmware package tarball

The firmware package is provided to the customer by the ODM as a tarball (“<*file name*>.tar”) and can be loaded to the NOS. The file’s name may be any name selected by the ODM, it is expected that this name will uniquely describe a content of the file in the human readable manner.
The tarball shall include at least the following files:

- metadata file: “metadata.json”
- ONIE firmware installer file: “<*file name*>.updater”

18.2.2 Metadata format

The metadata file shall use the JSON format with the following properties:

- metadata_version: the version of the metadata encoding (for backward compatibility). Metadata version is using two decimal values separated by dot (“.”) operator, e.g. 0.1 or 1.0.
- “package_type”: type of the package is set to “firmware” in order to distinguish ONIE installers for FW upgrade from other types of ONIE installers (e.g. for DiagOS upgrade)
- package_version: a number representing the set of firmware components’ target versions in the package. After firmware upgrade process on the device is accomplished the package_version value should be presented by the “onie_fwpkg” utility output under “version” column.
- onie_version: the required ONIE version to run the firmware installer
- platforms: an array of platform names supported by this package. The platform names are presented in the platforms array according to the value stored in the ONIE board EEPROM as defined by https://opencomputeproject.github.io/onie/design-spec/hw_requirements.html;
    - Each platforms object includes an array of supported hardware revisions. The platform hardware revision is stored in the ONIE EEPROM as defined by https://opencomputeproject.github.io/onie/design-spec/hw_requirements.html. If metadata.json does not include hardware revision of the device, ONIE installer will not proceed with firmware upgrade on the device. For each revision object, the following properties shall be specified:

- max_total_seconds: the maximum amount of time required for upgrading all firmware components on the component (including all device reboot times as part of the upgrade process). If the ONIE firmware installer does not send a “success” notification within this amount of time, the NOS controller will deduce that the firmware upgrade of the component has failed
- fw_components: an array of firmware sub-components that may be upgraded by the ONIE firmware installer. The following properties shall be specified for each fw_component object:
    - fw_coponent: the name of the firmware component, using the following convention: “<component name>-<index>. E.g., “CPLD-MB CPLD1” where “CPLD” is a name and “MB CPLD1” is an index. If there is no index for the firmware component, just a component name is specified. E.g. “BIOS”.
    - min_src_version: the minimum currently running firmware component version that can be upgraded
    - version: the target version that the ONIE firmware installer will install.
    - image_name: the name of the firmware image file within the ONIE firmware installer package. This parameter is not relevant for the NOS but may be used by the ONIE firmware installer script. Any image name convention may be used by the ODM.
    - fw_upgrade_seconds: the maximum amount of time required for upgrading the specific firmware component. The NOS uses this parameter before the upgrade process to estimate the firmware upgrade time

Below is an example of metadata.json file content. Please refer to Section 18.4 for metadata examples of specific ODM vendors.

```
{
   "metadata_version": "0.5",
   "package_version": "1.0.1",
   "onie_version": "2017.08",
   "package_type": "firmware",
   "platforms":[
      {
         "XYZ-W1":[
            {
               "ALPHA":
               {
                  "max_total_seconds": "1890",
                  "fw_components": [
                     { "fw_component": "BMC",      "min_src_version": "1.5", "version": "8.21",
"image_name:": "xxxx.bin","fw_upgrade_seconds": "540"},
                     { "fw_component": "CPLD-CPU",      "min_src_version": "0.0", "version": "0.30",
"image_name:": "xxxx.bin","fw_upgrade_seconds": "170"},
                     { "fw_component": "CPLD-MB",      "min_src_version": "0.0", "version": "9.11",
"image_name:": "xxxx.ima","fw_upgrade_seconds": "170"},
                     { "fw_component": "BIOS",      "min_src_version": "0.0", "version": "3.19",
"image_name:": "xxxx.bin","fw_upgrade_seconds": "120"},
                     { "fw_component": "ETH-1G", "min_src_version": "0.0", "version": "7.13",
"image_name:": "xxxx.rpd","fw_upgrade_seconds": "10" },
                     { "fw_component": "ETH-10G",      "min_src_version": "0.0", "version": "5.10",
"image_name:": "xxxx.bin","fw_upgrade_seconds": "125"},
                  ]
               },
               "BETA":
               {
```


18.2.3 Real-time progress notifications format

The firmware upgrade process may take a long time. For example, the firmware upgrade of a component with multiple firmware sub-components may take ~45 minutes. The ONIE firmware installer sends notifications on each step, reporting the upgrade's progress to the user.
There are two channels for sending progress notifications:

- Simple HTTP GET requests with notification messages encoded into the URL path – this channel is already implemented by DriveNets and several ODMs and is deployed in the field.
- The Redfish channel – the ONIE firmware installer sends notifications as Redfish events.

18.2.3.1 Simple HTTP notifications

The Simple HTTP notification channel assumes that the ONIE firmware installer sends HTTP GET requests to the NOS controller. There might be more than one NOS controller, and it is assumed that the NOS provides controller addresses and parameters as part of the ONIE firmware installer staging on the component.
The ONIE firmware installer shall get the parameters from the NOS controller. The following parameters shall be used:

- Optional: PROTOCOL:
    - HTTP - default
    - HTTPS (without authentication)

- Mandatory: ADDRESS_1, ADDRESS_2, ADDRESS_3 … : a list of n addresses of remote servers for sending notifications to (FQDN or IP)
- Optional: PORT: Destination L4 port (1-64K) – default: 80
- Optional: TIMEOUT: HTTP timeout (sec) – default: 5 sec

The NOS controller is expected to pass the above parameters to the ONIE firmware installer. The proposed method is that after staging the ONIE FW installer as per link, NOS controller will create a file "/mnt/onie-boot/onie/update/ocp_install.conf". Since the /onie/update/ folder is mounted by the ONIE, once booted ONIE firmware installer will have the access to the "ocp_install.conf" text file. The "ocp_install.conf" text file shall include all the above parameters, one parameter at line:

```
ADDRESS_1=192.168.1.1
ADDRESS_2=192.168.1.2
ADDRESS_3=192.168.1.3
PORT=80
TIMEOUT=5
```

When ONIE firmware installer boots up, it shall read the parameters from the ocp_install.conf file. The ONIE firmware installer shall send notifications for every step in the upgrade process. Each notification shall include the following information regarding the currently upgraded firmware component:

- version: version of the notification format, for backward compatibility. The version value should be equal to the "metadata_version" value in the metadata.json file (see section 18.2.2)
- serial-number: S/N of the upgraded white box device (i.e. DDC component)
- fw-component (CPLD-CPLD CPU / CPLD-MB CPLD1 / BIOS / BMC / ALL etc.)
    - ALL – represents notification for the whole upgrade process
    - The fw-component name is according to the convention defined in the metadata.json file (see section 18.2.2)
- status:
    - 0: FW upgrade NOT_NEEDED
    - 1: FW upgrade STARTED
    - 2: FW upgrade DONE
    - 3: FW upgrade FAILED
    - 4: FW upgrade REBOOT

The above information is sent as an HTTP/HTTPS GET request in the following format:

GET {PROTOCOL}://{ADDRESS}:{PORT}/update-status/{*version*}/firmware/{*serial-number*}/{*fw-component*}-{*status*}

An example of the sent notification:

GET HTTPS://10.1.1.1:443/update-status/1.0.1/firmware/WXCSD23ASD/BMC-4

Please relate to Section 18.4 for an example of simple HTTP notification implementation example.

18.2.3.2 Redfish notifications

Compatibility with Redfish hardware management is required to enable Redfish notifications, which will be implemented in the future.

18.2.4 ONIE firmware installer flow

The exact logic and the upgrade sequence of the ONIE firmware installer are determined by the ODM and may vary depending on the type of platform and target firmware versions. However, there are general requirements from the installer:

- The installer shall generate notifications (as per section 18.2.3.1 or section 18.2.3.2) for each firmware sub-component upgrade step
- The installer shall manage all the steps of the firmware upgrade sequences, including intermediate platform restart steps if required by the firmware upgrade sequence
- The installer shall generate FAILED or DONE status notifications for the overall upgrade process after finishing the firmware upgrade sequence for all the firmware sub-components on the DDC component

Figure 4 describes the proposed general firmware upgrade sequence on the DDC component:

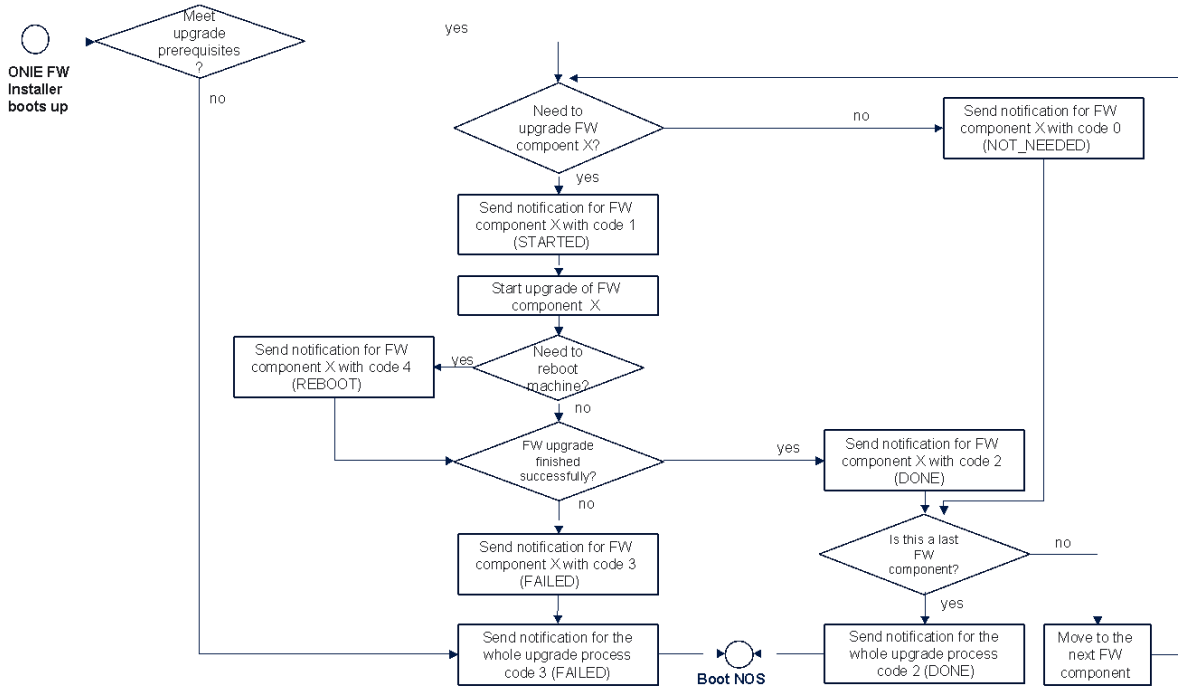

Figure 12: ONIE firmware installer flow

Please refer to Section 18.5 for examples of ONIE firmware installer implementation example.

18.2.4.1 Force and skip upgrade options

It is expected that ONIE FW installer will compare the currently running firmware version per every FW sub-component on the DDC component with target FW version in the package. If the currently running FW version is equal to the target FW version, no firmware installation for the sub-component will be done. Otherwise, ONIE FW installer will install a target version (even if a target version is lower than currently running version).

To force firmware installation for the sub-components that are running versions that are equal to the target version, a special parameter FORCE_UPGRADE is used as following:

- FORCE_UPGRADE = no (default): no firmware installation is done for firmware sub-component which is running firmware version equal to the target version in the package
- FORCE_UPGRADE = yes : firmware installation is done for firmware sub-component even if it is running version equal to the target version in the package
- FORCE_UPGRADE is an optional parameter, hence NOS is not required to specify it when staging ONIE firmware installer.
- The NOS controller is expected to set FORCE_UPGRADE parameter as part of the “ocp_isntall.conf” file defined by the section 18.2.3.1 as following:

```
ADDRESS_1=192.168.1.1
ADDRESS_2=192.168.1.2
ADDRESS_3=192.168.1.3
PORT=80
TIMEOUT=5
FORCE_UPGRADE = yes
```

To skip firmware installation for the sub-components that are running versions that are not equal to the target version, a special parameter SKIP_UPGRADE is used as following:

- SKIP_UPGRADE = list of fw_component names (e.g. BMC, BIOS, CPLD-CPU): list of firmware components specified by the SKIP_UPGRADE parameter will not be upgraded by the installer
- SKIP_UPGRADE is an optional parameter, hence NOS is not required to specify it when staging ONIE firmware installer.
- SKIP_UPGRADE parameter may be specified together with FORCE_UPGRADE parameter

    - If specific component is specified in the list of SKIP_UPGRADE, it will not be upgraded even if “FORCE_UPGRADE=yes” is set
- The NOS controller is expected to set SKIP_UPGRADE parameter as part of the “ocp_isntall.conf” file defined by the section 18.2.3.1 as following:

```
ADDRESS_1=192.168.1.1
ADDRESS_2=192.168.1.2
ADDRESS_3=192.168.1.3
PORT=80
TIMEOUT=5
SKIP_UPGRADE = BMC, BIOS
```

18.3 ONIE firmware installer

The ONIE firmware installer can be used to execute firmware upgrades in place of the NOS. Since each hardware vendor may have different ways of conducting firmware upgrades, performing the firmware upgrade with ONIE will allow for a standardized flow with the NOS.

The ONIE firmware installer is provided by the hardware vendor and should adhere to the following design concepts:

1) It should define what components could be updated through the multi-updater, list them in the metadata, and block unknown/unsupported hardware revisions and platforms.
2) It should define the proper update flow and order of components.
3) It should define possible ODM parameters that provide firmware update flexibility
4) It should minimize the update time and reboots.
5) It should provide a user-friendly firmware version list and update record for users to check in the NOS.

18.3.1 ONIE firmware installer from Edgecore

Edgecore’s Multiple-Firmware Updater (MFU) is designed according to the firmware updater architecture in ONIE. It can be started either manually in ONIE Rescue Mode, or automatically in ONIE Update Mode.

Before starting the MFU, the user can optionally provide an *ocp_install.conf* file for the MFU to send HTTP GET notifications.

If the user wants to start the MFU manually, the user can enter ONIE Rescue Mode and use the *onie-self-update* command to run the MFU.

If the user wants to start the MFU from the NOS, the NOS can use the *onie-fwpkg add* command to stage the MFU for execution. The NOS will then boot the switch into ONIE Update Mode. ONIE Update Mode will then automatically find and run the MFU. After the MFU starts, it will check firmware versions running in the switch to determine for each component whether update is needed. The MFU will then update components in a pre-defined order. Supported components, as defined in *metadata.json*, are *ONIE, BIOS, BMC, Diag, CPLD-Main, CPLD-CPU*, and *CPLD-Fan*.

As the MFU runs, it will send HTTP GET notifications to servers listed in *ocp_install.conf*. Supported status are *NOT_NEEDED(0), STARTED(1), DONE(2), FAILED(3),* and *REBOOT(4).* Each notification will contain one status for one component processed. The update procedure for some components may reboot the switch as needed. If the MFU passes, the final notification will contain *All-2*, meaning the All Components（*All*）are *DONE(2)*.

After the MFU finishes, it will restore the boot order and boot back to whatever was the default boot option before the MFU started. Result can be viewed using the *onie-fwpkg* and *onie-fwpkg show-results* commands.

18.3.2 ONIE Firmware Installer from UfiSpace

UfiSpace's Multiple Firmware Updater (MFU) follows standard ONIE flow to perform firmware update in ONIE update mode.

Users can either use *onie-fwpkg* command to stage the updater and *onie-boot-mode/efibootmgr* command to go to the ONIE update mode, or execute it directly in ONIE rescue mode.

The MFU will automatically identify the machine, block unsupported HW revisions, and handle the firmware dependency and compatibility.

It also controls the update flow to update the components listed in the metadata sequentially, optimize the update time, and provide timeout and retry mechanism. UfiSpace's MFU supports all of the parameters mentioned in the previous sections, and also the notification mechanism.

When the firmware update completes, a necessary system power cycle will be taken, and the system will go back to the NOS if the NOS exists, or the ONIE install mode to get ready for NOS installation.

In addition to the commonly defined update.log, UfiSpace's MFU also supports *update_details.log* to record the detailed tool logs for further debugging when issues

occur. Besides, the MFU provides an *ufi-fw-version* as the current firmware list for the use of any cases.

18.4 Real-time notification implementation examples

An example of wget based bash implementation for sending HTTP notifications:

```
update_status()
{
    #the following values are comming from ENV:
    # - serial_number
    # - url
    # - url2
    # - url3
    # - version

    code=$1 #status code
    if [ ! -z "${code}" ] && [ ! -z "${serial_number}" ]; then
      if [ ! -z "${url}" ]; then
        /usr/bin/timeout -t ${wget_timeout} /usr/bin/wget -q -O /dev/null
http://${url}:${update_port}/update-state/${notif_ver}/firmware/${serial_number}-${code
} &
      fi
      if [ ! -z "${url2}" ]; then
        /usr/bin/timeout -t ${wget_timeout} /usr/bin/wget -q -O /dev/null
http://${url2}:${update_port}/update-state/${notif_ver}/firmware/${serial_number}-${cod
e} &
      fi
      if [ ! -z "${url3}" ]; then
        /usr/bin/timeout -t ${wget_timeout} /usr/bin/wget -q -O /dev/null
http://${url3}:${update_port}/update-state/${notif_ver}/firmware/${serial_number}-${cod
e} &
      fi
```

18.5 ONIE firmware installer implementation examples

In this chapter we introduce some implementation examples.

18.5.1 Data structure

There are some key files that could be included.

- *fw-install.sh* – Initiate firmware update and control firmware update process
- *fw-version.make* – Specify the version of the firmware updater.
- *fw-common.conf* – Anything that would be commonly used can be defined in this file.
- *metadata.json* – Please refer to section 18.2.2.
- *ufispace-libs-tools.tar.gz* – Any tools, libraries, etc.
- *bios/bmc/ … etc* – Folders including update scripts, component images, and possible config files.

```
ufispace_s9700_53dx/
|-- firmware
|   |-- fw-install.sh
|   |-- fw-version.make
|   |-- fw-common.conf
|   |-- metadata.json
|   |-- bios
|   |   |-- ufispace-s9700_53dx-bios_v1.0.bin
|   |   `-- update_bios.sh
|   |-- bmc
|   |   |-- ufispace-s9700_53dx-bmc_v1.0.bin
|   |   `-- update_bmc.sh
|   |-- cpld
|   |   |-- ufispace-s9700_53dx-cpu_cpld_v1.0.rpd
|   |   |-- ufispace-s9700_53dx-mb_cpld_c1_v1.0.rpd
|   |   |-- ufispace-s9700_53dx-mb_cpld_cx_v1.0.rpd
|   |   `-- update_cpld.sh
|   |-- eth-10g
|   |   |-- ufispace-s9700_53dx-10g_v1.0.bin
|   |   `-- update_10g.sh
|   |-- eth-1g
|   |   |-- ufispace-s9700_53dx-1g_v1.0.bin
|   |   `-- update_1g.sh
|   |-- plx
|   |   |-- ufispace-s9700_53dx-plx_v1.0.bin
|   |   `-- update_plx.sh
|   |-- psm
|   |   |-- ufispace-s9700_53dx-psm_v1.0.hex
|   |   `-- update_psm.sh
|   `-- ufispace-libs-tools.tar.gz
|-- INSTALL
|-- busybox
|   `-- conf
|       `-- config
|-- demo
|   `-- platform.conf
```

fw-install.sh

```
#!/bin/sh

#
# Copyright (C) 2019-2021 Jay Lin <jay.tc.lin@ufispace.com>
#
# This script is the entry point of the ONIE firmware update
# mechanism.
# A machine uses this script to update "firmware", such as:
# BMC, CPLD, BIOS, EEPROM, PSM, ...

. ./fw-common.conf

### 1. BMC update ###

log_fw_update "================ Updating BMC ================"
bmc/update_bmc.sh || {
log_fw_update "ERROR: Failed to update BMC"
update_status "BMC" "$FU_FAILED"
exit 1
}
log_fw_update "==================== Done ===================="

if [ "$update_status" == "$FU_NOT_NEEDED" ]; then
    update_status "BMC" "$FU_NOT_NEEDED"
else
    update_status "BMC" "$FU_DONE"
fi


### 2. CPLD update ###

log_fw_update "================ Updating CPLD ================"
```

fw-common.conf

```
#Firmware upgrade status code
FU_NOT_NEEDED=0
FU_STARTED=1
FU_DONE=2
FU_FAILED=3
FU_REBOOT=4

function update_status()
{
    fw_component=$1
    code=$2

    #protocol, detault is http
    if [ -z "${PROTOCOL}" ]; then
        PROTOCOL="http"
    fi
    #port, default is 80
    if [ -z "${PORT}" ]; then
        PORT="80"
    fi
    #timeout, default is 5
    if [ -z "${TIMEOUT}" ]; then
        TIMEOUT="5"
    fi

    if [ ! -z "${code}" ] && [ ! -z "${sn}" ]; then
        if [ ! -z "${ADDRESS_1}" ]; then
            /usr/bin/timeout -t ${TIMEOUT} /usr/bin/wget -q -O /dev/null
"${PROTOCOL}://${ADDRESS_1}:${PORT}/update-status/${fw_ver_num}/firmware/$
{sn}/${fw_component}-${code}" &
        fi
        if [ ! -z "${ADDRESS_2}" ]; then
            /usr/bin/timeout -t ${TIMEOUT} /usr/bin/wget -q -O /dev/null
```

fw-version.make

```
#
# Copyright (C) 2019-2021 Jay Lin <jay.tc.lin@ufispace.com>
#

# The firmware version is a free form string
FW_VERSION  = ufispace-S9700_53DX-fw-v1.1.0
```

19. Hardware Management

- See Section 13

20. Security (only for Platform Boards and Systems)

All AT&T White Box initiatives have integrated TPM 2.0 hardware modules installed. (As indicated in section 7.3.1) Although hitherto they have been deployed in a 'disabled' state; AT&T is preparing to leverage this built-in security hardware, along with security elements of Unified Extensible Firmware Interface (UEFI) to combat successful malicious attacks on our DDC White Box network.
TPM has the following capabilities:

1. Performing public key cryptographic operations
2. Computing hash functions
3. Key management and generation
4. Secure storage of keys and other secret data
5. Random number generation
6. Integrity measurement
7. Attestation
8. Hard disk encryption

With TPM 2.0 enabled, vendors can setup inbound checks against PCR measurements from a remote or local verifiers; together with other security firmware interfaces such as, UEFI secure boot, TPM minted EK certificates and a host of other hardware security modules (HSM), to ensure the security of their disaggregated white box environment.

21. References (recommended)

[1] "Title", publication year, publication journal/conference/standard, volume, pages, link to publication if available
[2] OCP Profiles - https://github.com/opencomputeproject/OCP-Profiles
[3] Redfish Interop Validator - https://github.com/DMTF/Redfish-Interop-Validator
[4] Redfish Service Validator - https://github.com/DMTF/Redfish-Service-Validator
[5] Redfish Service Conformance Check - https://github.com/DMTF/Redfish-Service-Conformance-Check

22. Tables and Figures

Appendix A – Checklist for IC approval of this Specification (to be completed contributor(s) of this Spec)

Complete all the check list items in the table with links to the section where it is described in this spec or an external document.

Item	Status or Details	Link to detailed explanation
Is this contribution entered into the OCP Contribution Portal?	Pending to be presented to community	Link to OCP Contribution portal database to be provided
Was it approved in the OCP Contribution Portal?	Pending to be uploaded to Portal	Link to OCP Contribution portal database to be provided
Is there a Supplier(s) that is building a product based on this Spec? (Supplier must be an OCP Solution Provider)	Yes	UfiSpace
Will Supplier(s) have the product available for GENERAL AVAILABILITY within 120 days?	Yes	UfiSpace

Appendix B-UfiSpace - OCP Supplier Information and Hardware Product Recognition Checklist

(to be provided by each supplier seeking OCP recognition for a Hardware Product based on this specification)

Company: UfiSpace
Contact Info: andrew.lui@ufispace.com

Product Name: Disaggregated Core and Edge Router
Product SKU#: S9710-76D and S9701-82DC
Link to Product Landing Page:
S9710-76D: Click here for Specifications
S9701-82DC: Click here for Specifications

The following is needed for OCP hardware product recognition:

For OCP Inspired™

- All Suppliers must be a Silver, Gold or Platinum Member.
- Declare product is 100% compliant with specification
- Complete the OCP Inspired™ Product Recognition Checklist, which includes hardware management conformance checks and security profile.

For OCP Accepted™

- All Suppliers must be an OCP Member. All corporate membership levels are eligible.
- Complete the OCP Accepted™ Product Recognition Checklist, which includes hardware management conformance checks, security profile and open system firmware conformance checks.
- Submit a design package meeting OCP Hardware Design Guideline Contribution Checklist (if not already submitted by the contributor). If already submitted, declare the product is 100% compliant with the design package.
- Submit a firmware package including a firmware image, build scripts, documentation, test results and a tool that verifies modifications
- Submit the BMC source code, if applicable to product type

Please complete the OCP Inspired™ Product Recognition Submission Checklist or OCP Accepted™ Product Recognition Checklist and the following table

Item	Details	Links
Which Product recognition?	OCP Accepted™	TBD
If OCP Accepted™, who provided the Design Package?	UfiSpace	TBD
Where can a potential adopter purchase the product?	UfiSpace	OCP Marketplace