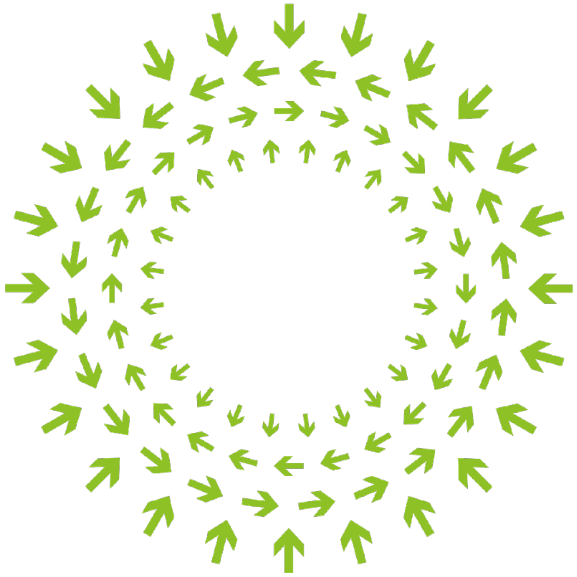


OPEN Compute Project

800G Distributed Disaggregated Chassis Routing System Evolution (V3)

Revision 5.0

Base Specification
Effective Oct 6, 2023

Author: Rehan Karim, Director Whitebox Design & Engineering - AT&T Labs
Author: Shivakumar Mysore, Lead Member of Technical Staff - AT&T Labs
Author: Douglas Olsen, Principal Member of Technical Staff - AT&T Labs

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”)

Note:

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.

1.2 Acknowledgements

The Contributors of this Specification would like to acknowledge the following companies for their feedback: **UfiSpace, HPE, Intel, Drivenets.**

- Kei Lee, AVP Technical Sales, UfiSpace
- Ken Chen, Manager of Hardware Solution, UfiSpace
- Wesley Lou, Manager of Hardware Solution, UfiSpace
- Will Chang, Manager of Technical Marketing, UfiSpace
- Felipe Pastor, Technical Specialist, Intel
- Akshay Tyagi, Product Marketing Manager, Intel
- Jay Lynn Cassio, Cloud Application Engineer VROC, Intel
- Run Almog, Head of Product Strategy, Drivenets
- Rodney Richter, Chief Technologist, HPE

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 5.3.4 DCP5 as compared to DCP3/DCP4

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 921.6Tbps. See section 5.5 for scale comparison of DCP5 as compared to DCP3/DCP4

3. Revision Table

Date	Revision #	Author	Description
10/6/2023	5	Shivakumar Mysore	Jay Lynn Cassio, Cloud Application Engineer VROC, Intel added to acknowledgements section

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:

Version 1 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

Version 2 Hardware Specifications and Use Case Description for the DDC Routing System located at:
https://www.opencompute.org/documents/20220920-ddc-v2-ocp-specification-ufispace-edit-docx-1-pdf

For this document, the version 1.1 specification will be referred to as **[DDC-V1.1Spec]** and the version 2.0 specification will be referred to as **[DC-V2.0Spec]**.

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 and version 2.0 by introducing two new white box hardware specifications meeting all the common requirements specified in **[DDC-V1.1Spec]** and **[DDC-V2.0 Spec]**. 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, however it was still not within the fabric. Version 3.0 will propose a mechanism for transporting PTP timing across the DDC system including the fabric.

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-V3 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.

DDCv2 Introduced High capacity 400G NCP3/DCP3 and a 25G NCP4/DCP4 with potential NCC+NCM

5.1 DDC-v3 Introduce J3/R3 architecture for 800 G and 922T system

DDCv3 specification of our distributed disaggregated chassis routing system

- Promote more collaboration and vendors to use base specification to develop 800G DDC platforms using J3/R3
- Increases Capacity from 192 T to 922 T (64 *14.4)
- Increased Port density on LCs and Fabric (Port exhaust happens before BW exhaustion).
- Interoperability with existing 192 T cluster with 922 T cluster
- Support 2 SKU for catering to industry standard form factor such as QSFP-DD and OSFP
- New PTP and IEEE1558v2 support- timing enabled through Ramon3 fabric.
- NOS roadmap and enhancements
- NCC Roadmap from Gen 10 to Gen 11, also 2U to 1U
- Improved MU functionality with SSD FW
- Still planning to use Air cooling, No liquid cooling
- Intel VROC functions is developed on DCP and DCC product

Note: Uplifting existing Backbone core from J2/ramon based 192 T to J3/R3 based 992T is out of scope for this spec. May be a new document will cover the challenges and options.

DDC-v1	DDC-v2	DDC-v3	Function	Description	ODM SKU/Model	DriveNets NOS	Cisco NOS
DCP100==>	DCP1	DCP1	Line Card	40x100G +13x400G Fabric	UfiSpace-9700-53DX DNI-AGCXD40V1	UfiSpace & DNI	UfiSpace
DCP400==>	DCP2	DCP2	Line Card	10x400G+13x400G Fabric	UfiSpace-9700-23D	UfiSpace	UfiSpace
DCF48 ==>	DCF1	DCF1	Fabric	48x400G Fabric	UfiSpace-9705-48D DNI-AGCC048	UfiSpace & DNI	UfiSpace
DCF24 ==>	DCF2	DCF2	Fabric	24x400G Fabric	Ufispace	UfiSpace & DNI	UfiSpace
-	-	DCF3	Fabric	64x800G Fabric	UfiSpace-9725-64E	UfiSpace	UfiSpace
DCM ==>	DCM1	DCM1	Management	48x(1/10G) + 6x100G	Accton-AS-5916-54XL	Accton	Accton
DCC ==>	DCC1	DCC1	Control	Skylake or Later COTS	COTS Server	HP or Dell Servers	HP DL380P Gen 10
-	DCP3	DCP3	Line Card	36x400G+40x400G Fabric	UfiSpace-9710-76D	UfiSpace	
-	DCP4	DCP4	Line Card	64x25G+12x100G+6x400G Fabric	UfiSpace-9701-82DC	UfiSpace	
-	DCCM1	DCCM1	Ctrl+Mgmt	64x25G+12x100G+6x400G Fabric	UfiSpace-9701-82DC	TBD	
-	-	DCP5	Line Card	18x800G (or 36x400G) +20x800G Fabric	UfiSpace-9720-56ED	UfiSpace	

Figure 1: ODM SKU Model NOS vendors

5.2 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 ½ 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 (x,+6 DCP	13-DCF48, 2-48 DCPx, 4-DCM, 2-DCC
Large-1	992-Tbps	11 Cabinets((3 + 8 DCP)	20-DCF, 64 DCPx, 4-DCM, 2-DCC

Figure 2: DDC-RS Naming Cluster

5.3 DCP5/NCP5 – 18x800G NIF + 20x800G Fabric (New)

DCP5 is a J3 white box design that targets both the service provider WAN and DCI of cloud service provider use cases.

- DCP5 is designed to support line rate MACSEC on every port at 800G.
- J3 has a bigger internal TCAM compares to J2 and J2c+ which can cover most of the scenarios, so that there is no external TCAM interface reserved.

5.3.1 DCP5 Key Requirements

- The NIC chip used to support the 2x25G SFP28 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 DCP5 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	18x800G or 36x400G QSFP-DD (depend on the configuration)
Fabric	20x800G QSFP-DD
Craft	RJ45 serial console port USB3.0 Type-A Micro USB serial console port
NM	2 x 25G SFP+ Mgmt. port (PTP-Support) 1 x 10/100/1000 RJ-45 Mgt. port
Timing	10MHz input / output SMB 1PPS input / output SMB
NPU	BRCM J3 (BCM88860)
Route Scale	Internal TCAM
CPU	Daughter Board: Icelake-D Generation or Later
BMC	Aspeed AST2620
Synchronization	Stratum 3E OCXO ITU-T Synchronous Ethernet (SyncE) IEEE 1588v2 T-TC, T-BC/OC Class C
LED	Power and System status LEDs per unit Link and Activity LEDs per port FAN Status LED PSU Status LED per PSU
AirFlow	Redundant and Hot-swappable Fan, Port to Power (F2B)
PSU	Redundant Hot Swap AC or DC
MACSEC	Line Rate MACSEC on all 800G Network Ports

Figure 3: Key Figures for DCP5

5.3.2 DCP5 High Level Block Diagram

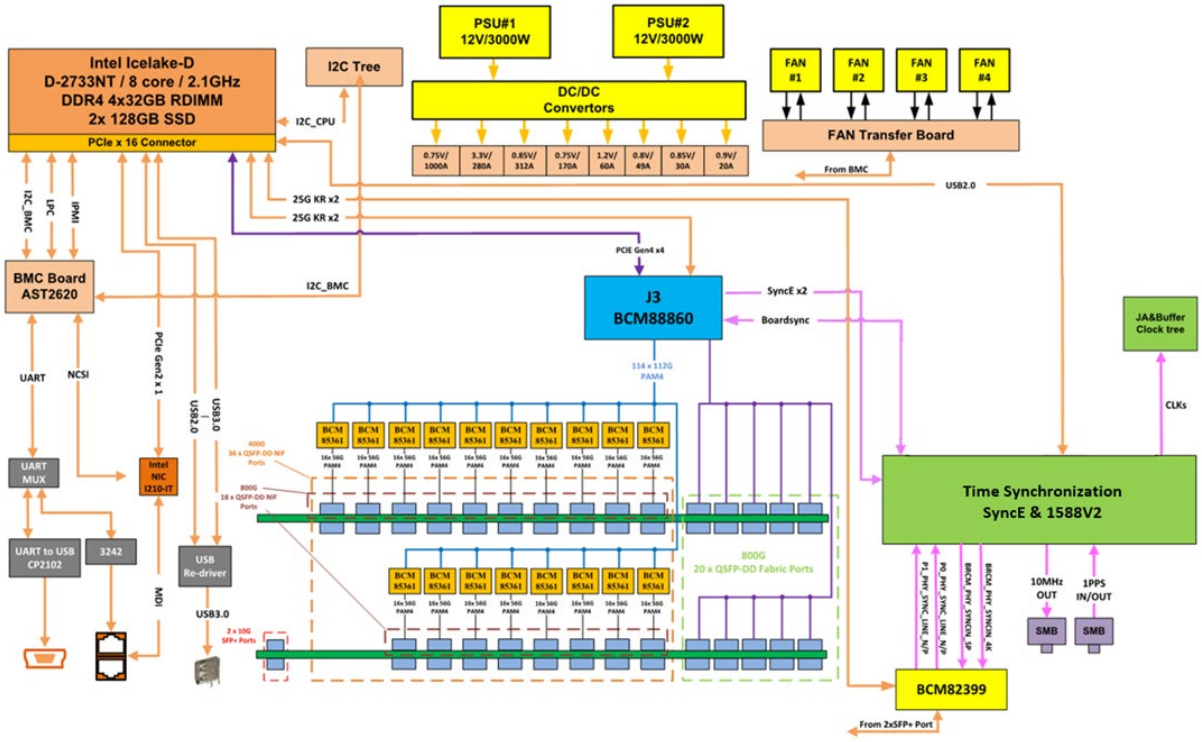


Figure 4 DCP5 High Level Block Diagram

5.3.3 DCP5 Fabric Serdes Routing Considerations

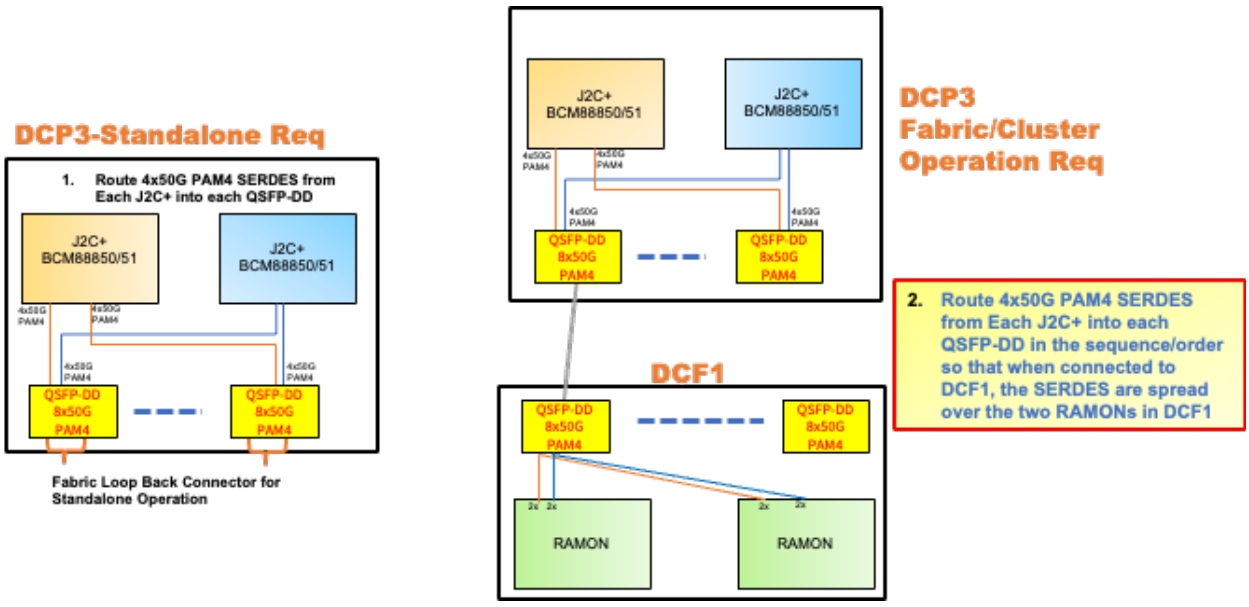


Figure 5: Illustration of Cross Fabric Serdes Routing in DCP3 Design

Cross fabric design is not applicable to Romon3 as we use only one Ramon3.

5.3.4 Comparison Chart of DCP/NCP Models

Following is the table of Line cards (AKA DCP/NCP) which were designed using different Broadcom chips catering to variety of capacity/port density/speed.

Spec. \ Model	NCP1	NCP2	NCP3	NCP4	NCP5
ASIC	Jericho2	Jericho2	**2x** Jericho2c+	Jericho2c	**Jericho3**
Switching Capacity	4.8Tbps	4.8Tbps	**2x** 7.2Tbps =14.4Tbps	2.4Tbps	**14.4Tbps**
CPU Subsystem					
CPU	**Broadwell-DE** 8-Core, 2.0 GHz	**Broadwell-DE** 8-Core, 2.0 GHz	**Skylake-D** 8-Core, 1.9 GHz	**Skylake-D** 8-Core, 1.9 GHz	**Icelake-D 8-Core, 2.1 GHz**
Port Configuration					
800G Fabric Ports	0	0	0	0	**20**
400G Fabric Ports	13	13	40	6	**0**
800G QSFP-DD	0	0	0	0	**18***
100/400G QSFP-DD	0	10	36	0	**36**
40/100G QSFP28	40	0	0	12	**0**
1/10/25G SFP28	0	0	0	64	**0**
Timing					
IEEE 1588	T-TC,	T-TC	T-BC, T-TC, T-TSC	T-GM, T-BC, T-TC, T-TSC	**T-BC, T-TC, T-TSC**
SyncE	NA	NA	Yes	Yes	**Yes**
1PPS	NA	NA	input/output	input/output	**input/output**
10MHz	NA	NA	input/output	input/output	**input/output**
GNSS	NA	NA	NA	input	**NA**
BITS	NA	NA	NA	NA	**NA**
TOD	NA	NA	NA	input	**NA**
Special Features					
TCAM external	Yes	Yes	Yes	Yes	**No (internal)**
IPSec	Yes	Yes	Yes	Yes	**Yes**
MACsec	No	No	Yes	No	**Yes**
OpenZR+	No	Yes	Yes	No	**Yes**

*18x800G ports only available if half of the 400G ports are disabled

5.3.5 Power Efficiency

Broadcom Jericho3 ASIC is crafted with 5nm process which provides better power efficiency than its predecessors. J3 is 14.4Tbps switching ASIC and consume around 700 Watts which can process around 0.02 Tbits with each watt it consumed. The number of J2c+ and J2 are 0.018 Tbits and 0.0129 Tbits per second accordingly.

5.4 DCF3 - 64x800G Fabric (New)

- DCF3 is a 1xRamon3 white box design that offers high density 800G fabric port to extend DDC scale.

5.4.1 DCF3 Key Requirements

- The NIC chip used to support the 2x25G SFP28 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.
- 10Mhz, and 1PPS support are required for DCF3 design for some use cases that may require them.

-

Feature	Description
Fabric	64x800G QSFP-DD
Craft	RJ45 serial console port USB3.0 Type-A Micro USB serial console port
NM	2 x 25G SFP+ Mgmt. port (PTP-Support) 1 x 10/100/1000 RJ-45 Mgt. port
Timing	10MHz input / output SMB 1PPS input / output SMB
CPU	BRCM R3 (BCM88920)
CPU	Daughter Board: Icelake-D Generation or Later
BMC	Aspeed AST2620
Synchronization	Stratum 3E OCXO ITU-T Synchronous Ethernet (SyncE) IEEE 1588v2 T-TC, T-BC/OC Class C
LED	Power and System status LEDs per unit Link and Activity LEDs per port FAN Status LED PSU Status LED per PSU
AirFlow	Redundant and Hot-swappable Fan, Port to Power (F2B)
PSU	Redundant Hot Swap AC or DC

Figure 6: Key Figures for DCF3

5.4.2 DCF3 High Level Block Diagram

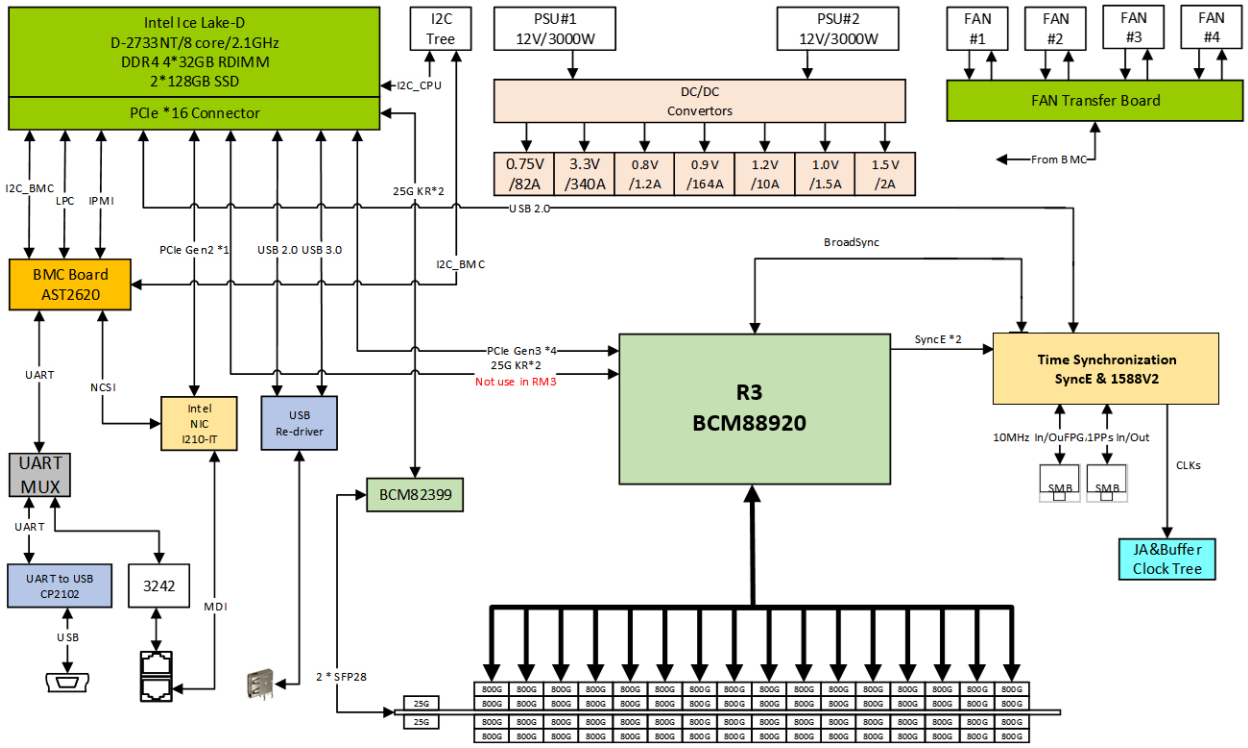


Figure 7 DCF3 High Level Block Diagram

5.4.3 Comparison Chart of DCF/NCF Fabric Models

Following table of Fabric cards (DCF/NCF) with associated Broadcam chips,capacity, port density and notably IEEE 1588 timing support

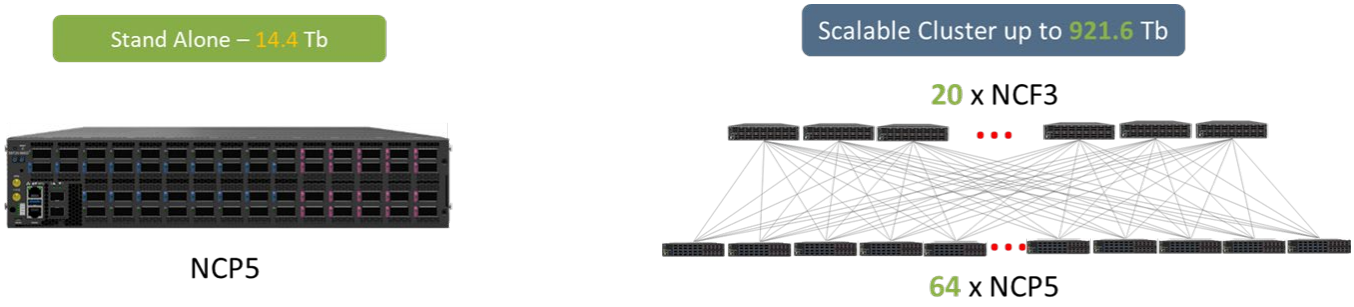
Spec. \ Model	NCF	NCF3
ASIC	**2x** Ramon	**Ramon3**
Switching Capacity	**2x** 9.6Tbps	**51.2Tbps**
Cell Throughput	8B cell/s	**21.6B cell/s**
CPU Subsystem		
CPU	**Broadwell-DE** 8-Core, 2.0 GHz	**Icelake-D 4-Core, 2.1 GHz**
Port Configuration		
800G Fabric Ports	0	**64**
400G Fabric Ports	48	**0**
Timing		
IEEE 1588	NA	**T-BC, T-TC, T-TSC**
SyncE	NA	**Yes**
1PPS	NA	**input/output**
10MHz	NA	**input/output**
GNSS	NA	**NA**
BITS	NA	**NA**
TOD	NA	**NA**

5.5 DCP5 & DCF3 scalability and interoperability with DCP3

- Some ports on DCF3 could be configured to support 400Gbps mode on demand and connect to both 400G fabric on J2/J2c+ boxes and remaining port with 800Gbps could be connected to J3 boxes.
- Fabric port on DCP5 could be configured to run 400Gbps mode and connect to DCF1
- DCP5+DCF3 can reach 921Tbps scale
- DCP5+DCF3 can use less space (15+48 2 RU units) compare with DCP3+DCF (37+48 2 RU units) to support 691Tbps scale

System Capacity(Tb)	#DCF(unit)	#DCP3(unit)	#DCF3(unit)	#DCP5(unit)
172	**10**	**12**	**4**	12
360	**20**	**25**	**8**	25
544	**30**	**38**	**12**	38
691	**37**	**48**	**15**	48
921	**-**	**-**	**20**	64

Another View of DDC v3 cluster topology and capacity with Purely NCP/DCP 5 and NCF/DCF 3



System Capacity(Tb)	#NCF3(unit)	#NCP5(unit)
14.4	**0**	1
230.4	**5**	16
460.8	**10**	32
921.6	**20**	64

6. Rack Compatibility

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

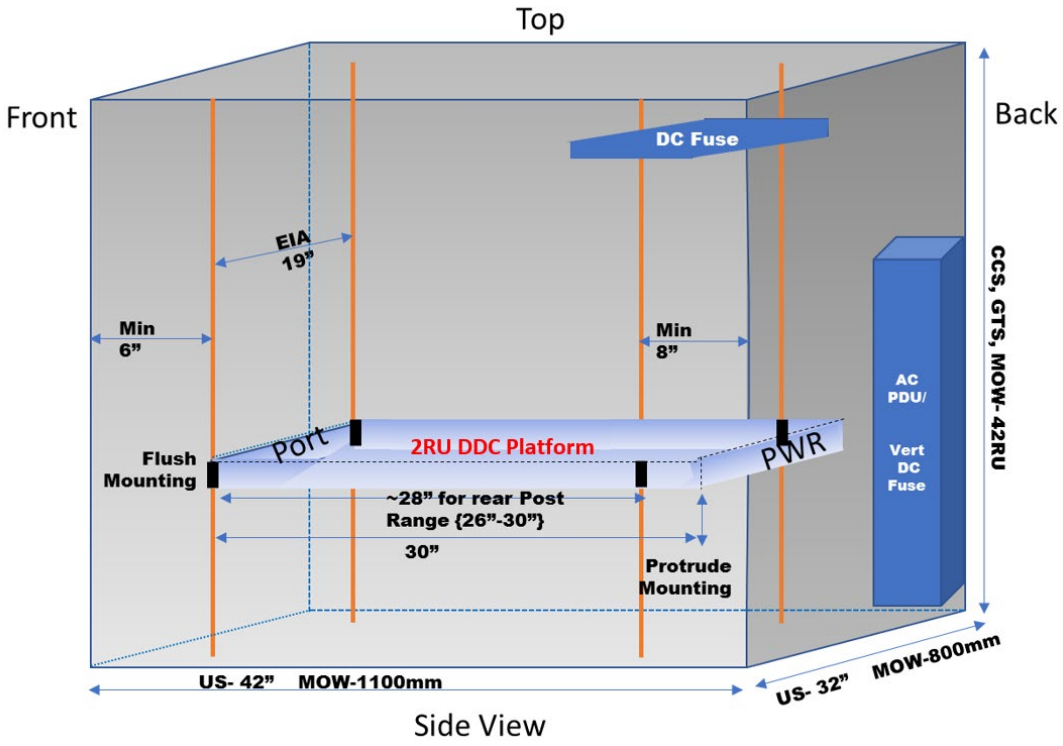


Figure 8: Rack Compatibility

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

7.3 X86 CPU Considerations

For DDC V3 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 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).
DDCv3 put SuperCap into design, which can provide necessary power to maintain the RTC while being compliant to the TP76200/TP76450 requirements.

7.4 Network Timing Module Implementation

DDC-V1.1 specification was lacking a solution to be able to synchronize PTP across the DDC. DDC-V2 addressed 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. The Figure below illustrates at a high level how PTP can be synchronized across DCPs in a DDC. This synchronization is only available on the newer DCPs.

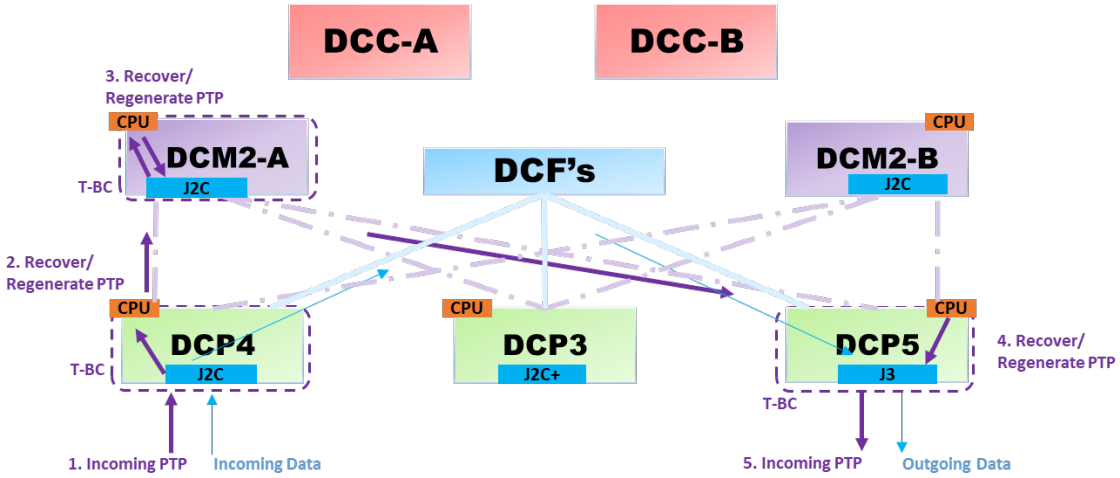


Figure 9: PTP synchronization across DCPs within a DDC

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.

DDC-V3 further defined the timing requirement on DCF box that allows the whole DDC cluster become timing aware.

1. DCF3 and DCP5 designs includes a specification of a class C or better NIC for the 2x25G SFP that connects to the DCM in a DDC configuration.

2. PTP synchronization can only occur between the newer DCPs (i.e., DCP3, DCP4 and DCP5)
3. In a brown field deployment with older DCP1 and DCP2, careful planning of PTP path is needed to avoid traversing older DCP's

8. Thermal Design Requirements

The components used in the J3/R3 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

Figure 10: 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

Figure 11: 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

Figure 12: 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

Figure 13: 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

Figure 14: 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.
- Highly recommended to use Redfish Implementation for all Devices DCPs/DCM/DCC
- 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. NOS should be able to detect the change and update the inventory info.

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 via BMC's IPMI watchdog timer
- Hardware Level reset of components supplied by 12-volt power rail. This is IPMI Power Cycle.

- In the newer boxes DCP5, Redfish needs to be implemented instead of legacy IPMI

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. 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

16. 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.

17. 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

DriveNets NOS challenges from 192 T ➔ 992T systems

- Timing support – per device and distribution within the DDC cluster
- New hardware introduced – J3/R3 from multiple ODM
- Higher scale (J3/R3 driven)
- Hybrid clusters
    - Mix of DCP versions
    - Mix of ASIC generations
    - Adheres to practicalities of the use case
- Additional use cases
    - Edge functionality – Tunneling, L2 OAM, SR, logical scale
    - Networking for AI/non AI workloads

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.

17.1 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 17.2. 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.

17.2 Firmware upgrade process

To detach the firmware upgrade logic from the NOS, an ONIE firmware installer method

is used as described in section 17.3. 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)

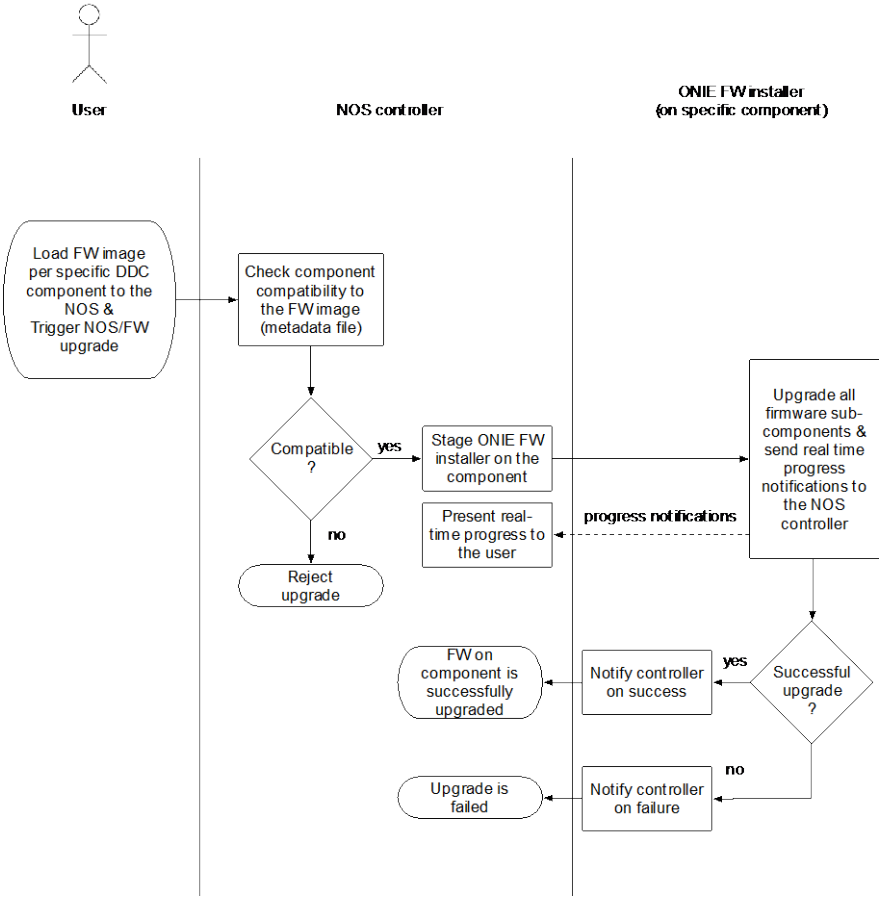


Figure 15: Firmware upgrade process in DDC

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 17.2 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.

17.2.1 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”

17.2.2 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 17.4 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"}
```

17.2.3 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.

17.2.3.1 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 17.2.2)
- 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 17.2.2**Error! Reference source not found.**)
- 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 17.4 for an example of simple HTTP notification implementation example.

17.2.3.2 Redfish notifications

Compatibility with Redfish hardware management is required to enable Redfish notifications, which will be implemented in the future for Newer Whitebox LCs supporting Redfish

17.2.4 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 17.2.3.1 or section 17.2.3.2) 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

Error! Reference source not found. describes the proposed general firmware upgrade sequence on the DDC component:

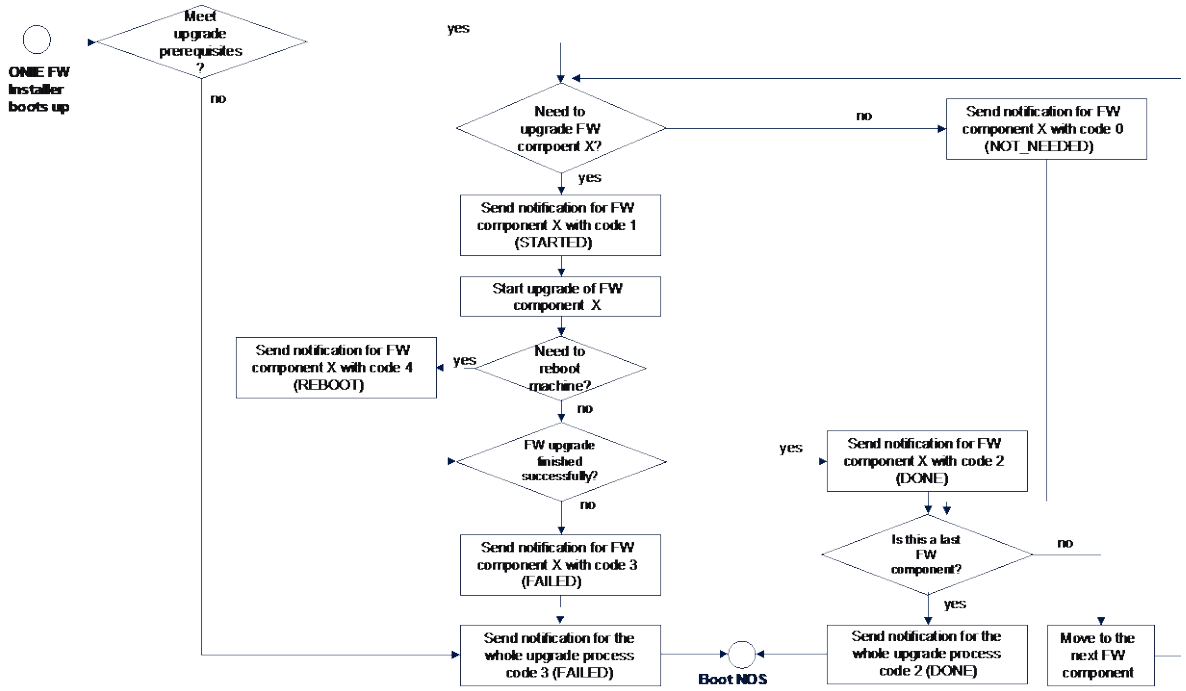


Figure 16: ONIE firmware installer flow

Please refer to **Error! Reference source not found.**17.5 for examples of ONIE firmware installer implementation example.

17.2.4.1 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 **Error! Reference source not found.**17.2.3.1 as following:

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

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 17.2.3.1 as following:

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

17.3 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.

17.3.1 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.

17.3.2 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.

17.4 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
```

17.5 ONIE firmware installer implementation examples

In this chapter we introduce some implementation examples.

17.5.1 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 17.2.2.
- *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
```

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"
```

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
```

fw-version.make

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

17.6 List of FW components

Example of field upgradable Firmware component list and possible ODM managed version for a given MultiUpdater. List includes SSD FW also. Currently PSU FW updates are not part of Multiupdater.

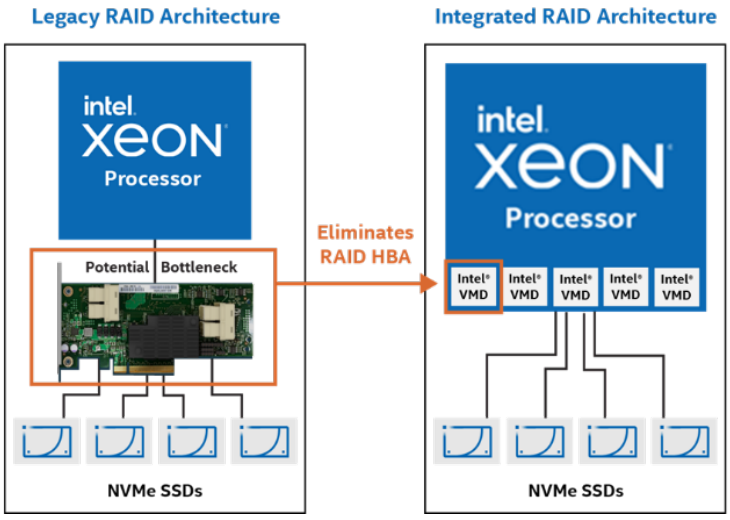
UfiSpace S9700-53DX Firmware Versions		Multi updater V1.1.0
Software	BIOS	T99O114T12_R04.16
Firmware	PCIe Switch	00000002
Firmware	CPU CPLD	0.30
Firmware	MB CPLD1	1.17
Firmware	MB CPLD2 to MB CPLD5	0.21
Firmware	ast2400e	3.65.000000
Firmware	Intel(R) Ethernet Connection X552 10 GbE SFP+	2.10
Firmware	Intel(R) I210 Gigabit Network Connection	3.25
Firmware	MB	R01190320 (Beta) R02191019 (PVT & later)
Firmware	SSD	N/A
Firmware	SSD	N/A
UfiSpace S9700-23D Firmware Versions		V1.1.0
Software	BIOS	T99O114T12_R04.16
Firmware	PCIe Switch	00000002
Firmware	CPU CPLD	0.30
Firmware	MB CPLD1	1.00

Firmware	MB CPLD2 to MB CPLD3	0.13
Firmware	ast2400e	3.65.000000
Firmware	Intel(R) Ethernet Connection X552 10 GbE SFP+	2.10
Firmware	Intel(R) I210 Gigabit Network Connection	3.25
Firmware	MB	R04190529 (Beta) R06191018 (PVT or later)
Firmware	SSD	N/A
Firmware	SSD	N/A
UfiSpace S9705-48D Firmware Versions		V1.1.0
Software	BIOS	T99O114T12_R04.16
Firmware	PCIe Switch	00000002
Firmware	CPU CPLD	0.30
Firmware	MB CPLD1	0.35
Firmware	MB CPLD2	0.23
Firmware	MB CPLD3	0.35
Firmware	MB CPLD4	0.23
Firmware	ast2400e	3.65.000000
Firmware	Intel(R) Ethernet Connection X552 10 GbE SFP+	2.10
Firmware	Intel(R) I210 Gigabit Network Connection	3.25
Firmware	TOP-MB	07200323
Firmware	BOT-MB	10200303
Firmware	SSD	N/A
Firmware	SSD	N/A

18 Intel® Virtual RAID on CPU (Intel® VROC) for Storage

18.1 Description

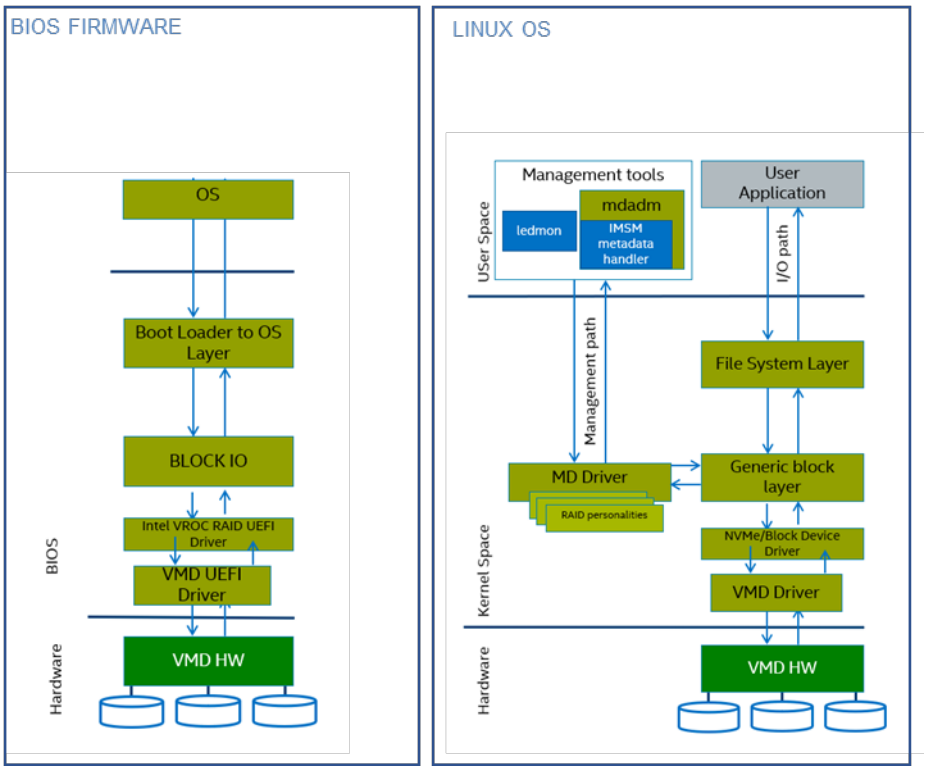
Intel® VROC is an integrated RAID solution on Intel® Xeon® Processors that provides redundancy, availability, and serviceability (RAS) for NVMe storage. Intel® VROC improves storage performance, lowers cost and improves power efficiency by eliminating the need of an external hardware RAID card. Other benefits are reduced complexity, space savings, and improved reliability by removing a single point of failure. As a part of DDC V3, Gen 11 NCC/DCC and NCP5/DCP5 Ufispace units are enabled with VROC features.



Intel VROC supports RAID 0, 1, 5, and 10. RAID level 1 can be selected for Boot. For this application RAID level 1 is recommended for boot. RAID 1 requires 2 devices. RAID 1 boot provides a mirrored copy of the Operating System on each NVMe device. This feature allows for long term serviceability for storage. In the event one device fails to respond, or completely fails, the system continues with servicing IO on the remaining healthy device with an exact copy of the Operating System.

18.2 Software and Firmware Components

Intel VROC includes both Linux OS and BIOS firmware components. The BIOs firmware include the Intel VMD UEFI driver and the Intel UEFI VROC driver to manage NVMe device enumeration and RAID handling. For Linux, the Intel VROC driver uses open source VMD and mdadm with a proprietary metadata, along with its management components.



18.3 Configuration Options

RAID 1 Boot on 2 NVMe devices can be customizable in size. The RAID 1 partition can be large enough to service the Operating system, but leave space on the devices for other purposes, such as logging and caching, and can implement Data RAID level 0 for increased capacity and performance.

The below tables illustrate the possible use cases with Intel VROC using only 2 NVMe devices. RAID 0 is mentioned due to the increased capacity and performance.

Use Case #1

Exposed Block Device	redundant	Purpose
RAID 1 on partial size of NVMe capacity	yes	Boot and Operating system redundancy
Logging	no	Section of NVMe to store logging data
Caching	no	Section of NVMe to cache data

Use Case #2

Exposed Block Device	redundant	Purpose
RAID 1 using entire capacity of NVMe	yes	Boot and Operating system redundancy
RAID 1 Logging	yes	RAID 1 Partition to store logging data
RAID 1 Caching	yes	RAID 1 Partition to cache data

Use Case #3

Exposed Block Device	redundant	Purpose
RAID 1	yes	Boot and Operating system redundancy
RAID 0 Logging	no	2nd RAID 0 on same 2 NVMe – striped for performance and increased capacity for logging data
RAID 0 Caching	no	2nd RAID 0 on same 2 NVMe – striped for performance and increased capacity for caching data

18.4 RAID Volume States

RAID volume will be in a “normal” state at creation. If one device fails, the state of the RAID volume will show as “degraded” until a new device is plugged into the system. The degraded state can continue without service and does not affect functionality or performance but is an indicator that a replacement device is needed for a redundant state to be maintained.

RAID Volume State	Description
Clean	RAID volume healthy, data is in sync
Clean, resyncing	RAID volume is initializing
Clean, Degraded	RAID volume has no redundancy due to one failed device
Clean, degraded, recovering	RAID volume detects newly inserted device and data is being copied across both devices

18.5 System Requirements

Intel® Volume Management Device (Intel® VMD) is an HBA-Like Controller on the uncore of the Intel Xeon Processor to manage NVMe. Intel VMD is the foundation for Intel VROC, and must be enabled on the platform to support Intel VROC.

18.5.1 BIOS Components:

BIOS	Purpose
Intel VMD - enabled	Allows VMD to manage NVMe
Intel VROC UEFI Driver	Allows Bootable RAID
Intel VROC HII Menu	GUI inside BIOS to configure/manage RAID

18.5.2 System OS components:

Intel VROC Linux is up streamed into the Linux community. Mdadm is used with Intel Proprietary metadata that is written to each device at the time of RAID creation. This metadata is used for management within the Operating System and within the BIOS environment. The Intel VMD driver is another Linux kernel component helping to manage the NVMe devices.

Linux OS Components for Intel® Xeon Gen 4 (Ice Lake)

Component	Kernel version or Component version
VMD	Kernel version – starting in 4.11 through 5.17
mdadm	4.2
Linux Kernel	<same as above>

Intel VROC Link to list of supported Linux Operating systems and Configurations.
https://www.intel.com/content/www/us/en/support/articles/000030310/memory-and-storage/datacenter-storage-solutions.html

18.5.3 Remote Deployment and provisioning

Intel VROC UEFI Driver packages include tools for deployment and provisioning RAID outside of the OS and BIOS menu.

The following tools can be used remotely to configure RAID and can be scripted into .nsh files for provisioning the platform with RAID, as well as running validation testing.

Component	Purpose
RcfgVROC.efi	This tool must match the version of the UEFI drivers being used. Configures RAID, and can mimic failing scenarios for testing purposes. NSH scripts for remote deployment can be created, using this tool.
RcmpVROC.efi	Use same version as driver version. Debugging purposes to ensure system compliance.
HWKeyChecker.efi	Same version as driver version. Debugging purposes, to ensure correct Hardware RAID key is installed and found by UEFI VROC drivers.

18.6 Error Handling Details

Intel VROC creates a robust system that is protected from kernel Panics and system hangs are avoided. In a RAID 1 configuration, in the event a device becomes unresponsive, Intel VROC will handle the error, without disruption to service, and the operation will continue on the next healthy device.

Intel VMD as a foundation of Intel VROC handles Advanced Error Reporting (AER) and provides logging within Linux, therefore providing protection to the rest of the PCIe bus, preventing whole system shut down. In Linux, the PCIe Bus driver or hotplug driver handles AER and logging. Upon receiving an interrupt at MSI-X 0 for each root port, the Intel® VMD driver uses a sophisticated algorithm to determine the precise device to apply error handling based severity. The actions taken are specific to the error on the device, to preserve the remaining eco system. <add that it goes into syslog>

Intel VMD will log the following errors. If a non-fatal error is encountered, the Intel VMD driver will log the error and standard recovery efforts are taken. If a fatal error is encountered, Intel VMD will treat the error as a “device surprise hot removed”. But system shall continue to operate without hang, reboot, or crash.

Advanced Uncorrectable Error	**Severity Register**
Poisoned TLP	Non-Fatal (0b).
Completion Timeout	Non-Fatal (0b).
Completer Abort	Non-Fatal (0b).
Unexpected Completion	Non-Fatal (0b).
ECRC Error	Non-Fatal (0b).
Unsupported Request Error	Non-Fatal (0b).
ACS Violation	Non-Fatal (0b).
Atomic Op Egress Blocked	Non-Fatal (0b).
TLP Prefix Blocked	Non-Fatal (0b).
MC Blocked TLP	Non-Fatal (0b).

19. DCC/NCC HPE Gen 10 to Gen 11 Changes

19.1 NCC/DDC in DDC v1 & DDC v2 Hewlett Packard Enterprise (HPE)

Existing AT&T Implementation of NCC per DDC V1 and V2 is as follows:

- HPE DL380 Gen 10 servers – Intel based CPUs - Skylake and Cascade Lake
- PCIe Gen 3, DDR4 memory, HPE Smart Array RAID Controller
- 2 xMellanox based Connect X-5 Dual Port 100Gb NICs
- TPM 2.0
- BMC - HPE iLO 5 (Integrated Lights-Out)
  - Supports IPMI & Redfish

19.2 Next Gen NCC/DDC proposed/Planned when DL380 reaches EOL/EOS

- Smart RAID, Mega RAID and Intel VROC options available
- PCIe Gen 5, DDR5 memory
- BMC - HPE iLO 6 (Integrated Lights-Out)
- Redfish supported (Recommended) and legacy IPMI functions.

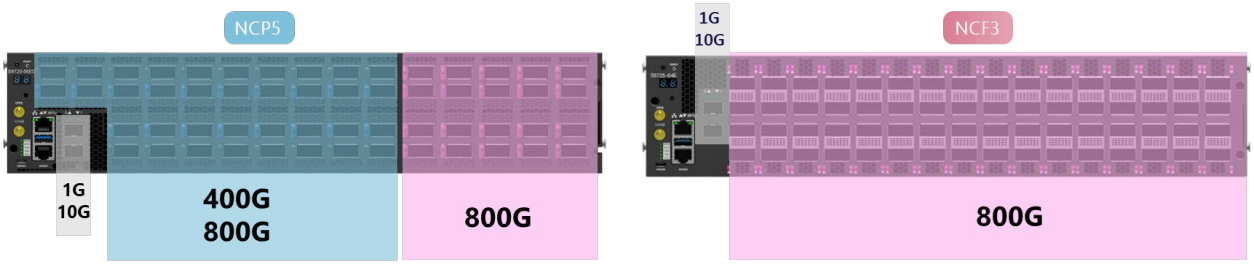
Note: Gen 11 System is not yet finalized. NEBS-3, thermal, NOS evaluation is in progress. New 1 RU system will be qualified instead of 2 RU system.
However, original proposal of 2 RU DL 380 with Gen 11 server is tabulated and compared against existing 2 RU Dl-380 Gen 10 system below. If DL 360 configuration is finalized before Global summit, we will update this section.

Specification	Gen10 DL380	Gen11-DL380
CPU	Cascade Lake – up to 28C Skylake – up to 28C	**Sapphire Rapids** – up to **60C** **HBM support**
PCIe	48x PCIe Gen3 lanes / socket	**80x** PCIe **Gen5** lanes / socket
Memory channels	6x DIMM channels / socket Support for 2 DIMMs per channel	8x DIMM channels / socket Support for 2 DIMMs per channel
Memory support	DDR4, up to 2933 MT/s 8GB to 128GB	**DDR5**, up to **4800** MT/s **16GB** to 256GB
SAS/SATA support	SFF/LFF	SFF/LFF

NVMe	Gen3; x4 connections U.2 drives	**Gen5;** x1, x2 & x4 connections U.3 **& EDSFF** drives
EDSFF	No support	**Gen5; E3.S 1T & 2T** drives
Power supplies	Up to 1600W	Up to **2200W**
Cooling	Air cooling	Air **& Hybrid*** cooling; DIMM blanks
Management engine	iLO 5	**iLO 6**

Purple color depicts Difference to Gen11; Bold: New on Gen11

20. Optics Support on NCP5/DCP5



Supported 3rd Party Optics				
1/10G	**400G**		**800G**	
SFP+ AOC SFP+ SR SFP+ LR	QSFP-DD AOC QSFP-DD SR8 QSFP-DD FR4 QSFP-DD LR4 QSFP-DD ER4	QSFP-DD ZR QSFP-DD OpenZR+ QSFP-DD DR4 QSFP-DD DR4+ QSFP-DD PLR4	QSFP-DD AOC QSFP-DD 2*DR4 QSFP-DD 2*FR4 QSFP-DD 2*LR4	QSFP-DD DR8 QSFP-DD DR8+

Above list is the Varity of 3 PO – 3rd Party Optics supported on the NCP5/DCP5 Model. Multiple Optics Vendors are being Qualified and available in the Market at the moment.

21. Hardware Management

- See Section 13

22. 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.

23. 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

24. Tables and Figures

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

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

Item	Status	Link to detailed explanation
Has this contribution been presented to an OCP Project group during a project call or engineering workshop?	Yes	To be presented in August Telco project call
Approval by Project Leads	Yes	To be approved prior to August Telco project call
Is this contribution entered into the OCP Contribution Portal?	Yes	To be contributed prior to August Telco project call
Was it approved in the OCP Contribution Portal?	Yes	To be approved prior to IC call

Appendix B - Contribution Process FAQs

As a contributor to a hardware specification, here are some questions that often come up.

Q1. What type of specification am I contributing to OCP?

a. **The base specification** for a de-facto standard (ex: interface type)
b. The base specification for a product <product type> (product may be coming but within the next 1-2 years)
c. Modification of an existing <type> specification (state which existing spec is being modified) resulting in a revised specification.
d. **Design specification** (based on an existing base specification) with more refined design details (product coming in 12-15 months)
e. A detailed **Product specification** for a <product type> for a very specific product being available in 3-6 months of approval of this Spec
f. If none of the above, please contact OCP Staff for better direction.

Q2. How do I know if what I am contributing will be accepted by OCP?

a. Before contributing any specifications, please contact either OCP Staff (Rob Coyle, Michael Schill) or the Project Lead for the Project that best represents your contribution. They will guide you as to what's the best form for your contribution. Project List here.

Q3. What is the contribution process for my hardware spec?

a. Follow the flow for your spec type here.

Q4. What if my spec is not developed yet and I want to collaborate with other companies?

a. Please contact either OCP Staff (Rob Coyle or Michael Schill) or the Project Lead for the Project that best represents your contribution. They will help you find other collaborators and help you with the contribution process for a multi-party contribution.

Q5. I have a question about the Contribution License Agreement (CLA).

a. Please contact OCP Staff and we can help you with questions.

Q6. Do I need to have a product in order to make a contribution?

a. Please see Q1. Some types of contributions do not result in a product. Some examples are whitepapers, case studies, OCP Ready Assessment, etc.. Please work with the OCP Staff on the better direction on your specification type.