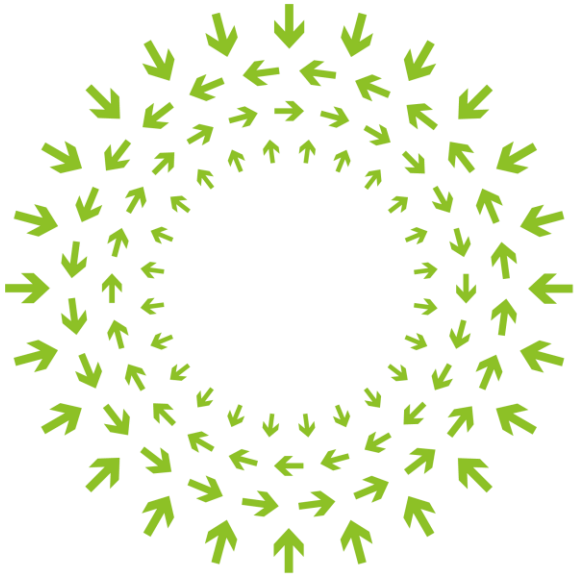

OPEN

Compute Project

Firmware upgrade for DDC

v0.2

Authors:

Evgeny Sandler, System Architecture Group, Drivenets

Jay Lin, Senior Staff Engineer, UfiSpace

Mike Liu, Software Design Assistant Manager, Delta Network International

Daniel Chung, Sr. Technical Manager, Accton

Tuan A Duong, Lead Member of Tech Staff, Network Infrastructure & Services, AT&T

Shivakumar Mysore, Principal Member of Tech Staff, Network Infrastructure & Services, AT&T

Version History

Ver	Author	Changes	Date
0.1	Evgeny Sandler	First draft	28-Feb-21
0.2	Evgeny Sandler	Change the method for passing parameters for the real-time progress notifications (section 6.3)	2-Aug-21

Table of Contents

VERSION HISTORY 2

TABLE OF CONTENTS 3

1. LICENSE (OCP CLA OPTION) 4

2. VOCABULARY 5

3. SCOPE 5

4. DDC OVERVIEW 5

5. PROBLEM STATEMENT 6

6. FIRMWARE UPGRADE PROCESS 6

6.1 Firmware package tarball 8

6.2 Metadata format 8

6.3 Real-time progress notifications format 9

6.3.1 Simple HTTP notifications 10

6.3.2 Redfish notifications 11

6.4 ONIE FW installer flow 12

6.4.1 Force and skip upgrade option 12

7. APPENDIX I – REAL-TIME NOTIFICATION IMPLEMENTATION EXAMPLES 14

7.1 Simple HTTP notification example 14

8. APPENDIX II – ONIE FIRMRWARE INSTALLER IMPLEMENTATION EXAMPLE 15

8.1.1 Data structure 15

8.1.2 Examples 16

1. License **(OCP CLA Option)**

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:

[DRIVENETS]
[UFISpace]
[Accton]
[Delta Networks International]
[AT&T]

Usage of this Specification is governed by the terms and conditions set forth in **[Open Compute Project Hardware License – Permissive ("OCPHL Permissive")] ("Specification License").**

Note: The following clarifications, which distinguish technology licensed in the Contribution License and/or Specification License from those technologies merely referenced (but not licensed), were accepted by the Incubation Committee of the OCP:

[None].

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.

2. Vocabulary

DDC – Distributed Disaggregated Chassis

NOS – Network Operating System

DNOS – DriveNets NOS

ODM – Original Design Manufacturer

TCO – Total Cost of Ownership

GA – General Availability

FW - Firmware

Tarball – tar archive

VAR – Value Added Reseller

3. Scope

This document defines the technical specifications for the firmware upgrade of DDC components. The DDC concept is specified as part of the OCP Network subproject. The DDC specification can be found here: https://www.opencompute.org/documents/ocp-hardware-specifications-and-use-case-description-for-j2-ddc-routing-system-pdf

The goals of this specification 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

The document includes specifications both for DDC ODMs and NOS vendors.

4. DDC overview

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 componets 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 distirbured 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 (see DDC specification for details). This sequence may change according to design considerations. NOS controller manages the firmware and software upgrades for each component in the cluster.

5. 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 fw components, estimated fw 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 in order to provide firmware upgrade estimations and pre-requisite validation. Maintaining this information as part of NOS is not practical.

This proposal offers 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 4.

Moreover, the proposed 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.

6. Firmware upgrade process

To detach the firmware upgrade logic from the NOS, we propose to use the ONIE firmware installer method. 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 upgrade method of NOS and HW DiagOS by ONIE is out of scope of this document.

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)

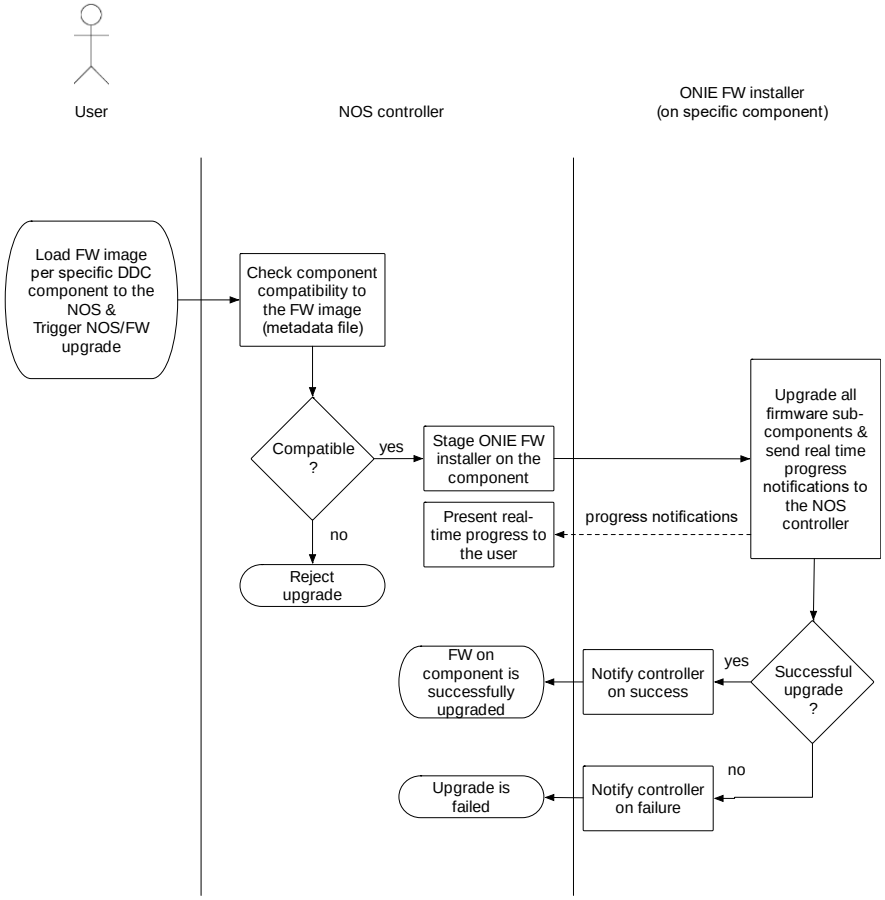

Figure 1 – firmware upgrade process in DDC

Figure 1 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 6.4 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.

6.1 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”

6.2 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 **Error! Reference source not found.** 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":
                    {
                        "max_total_seconds": "1890",
                        "fw_components": [
                            { "fw_component": "BMC",        "min_src_version": "1.5", "version": "8.22", "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": "9.12", "image_name:": "xxxx.ima","fw_upgrade_seconds": "170"},
                            { "fw_component": "BIOS",       "min_src_version": "0.0", "version": "3.20", "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"},
                        ]
                    }
                }
            ]
        }
    ]
}
```

6.3 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.

6.3.1 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 6.2)
- 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 6.2)
- 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 7.1 for an example of simple HTTP notification implementation example.

6.3.2 Redfish notifications

The list of Redfish event messages for passing the notifications and the mechanism by which the NOS controller subscribes to these event messages on every DDC component to be defined.

6.4 ONIE FW 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 6.3.1 or section 6.3.2) 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 2 describes the proposed general firmware upgrade sequence on the DDC component:

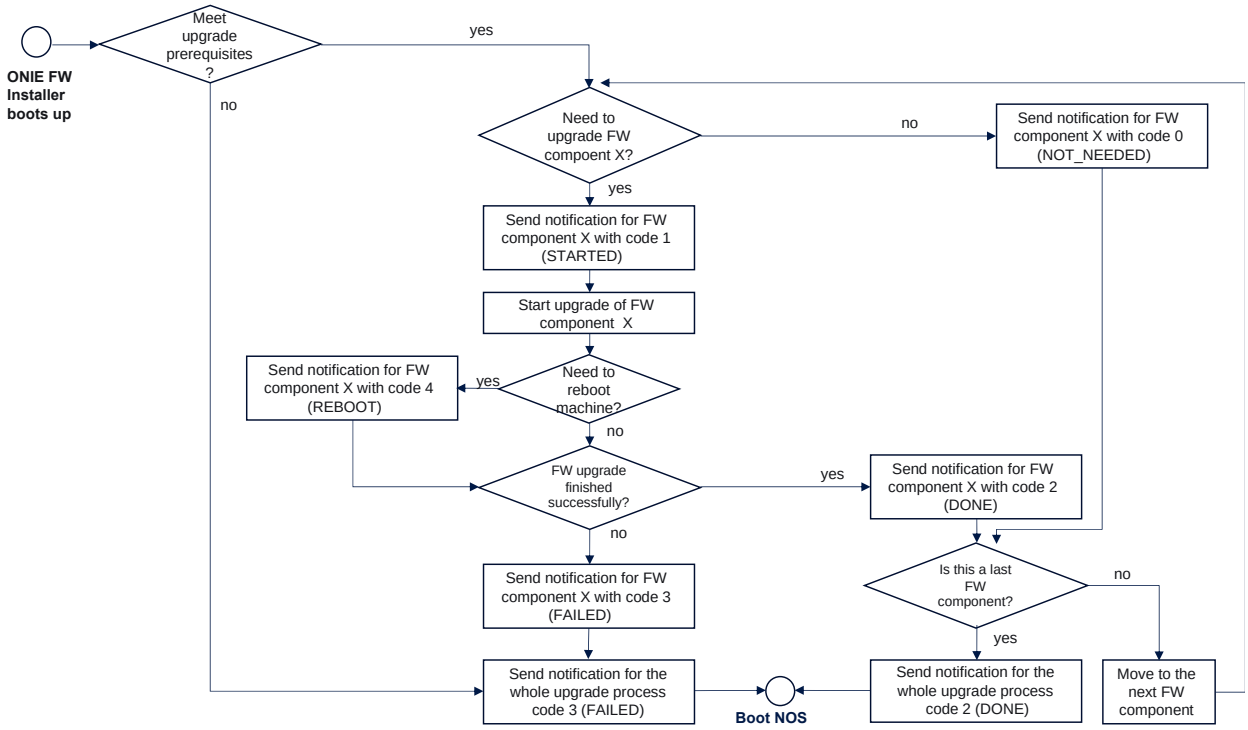

Figure 2 – ONIE firmware installer flow

Please refer to 8 for examples of ONIE firmware installer implementation example.

6.4.1 Force and skip upgrade option

It is expected that ONIE FW installer will compare the currently running firmare 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 targer 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 fw sub-component which is running firmware version equal to the target version in the package
- FORCE_UPGRADE = yes : firmware installation is done for fw 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 6.3.1 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 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 6.3.1 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
```

7. Appendix I – Real-time notification implementation examples

7.1 Simple HTTP notification example

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
```

8. Appendix II – ONIE firmrware installer implementation example

In this chapter we introduce some implementation examples.

8.1.1 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 Chapter 7.
- *ufispace-libs-tools.tar.gz* – Any tools, libraries, and 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
|-- installer
|   `-- install-platform
|-- installer.conf
|-- kernel
|   |-- config
|   `-- series
|-- machine.make
`-- rootconf
    `-- sysroot-lib-onie
        |-- init-platform
        `-- sysinfo-platform
```

8.1.2 Examples

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 ================"

...

log_fw_update "Update complete. No errors detected."
exit 0
```

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 ${TIMEOUT} /usr/bin/wget -q -O /dev/null
"${PROTOCOL}://${ADDRESS_2}:${PORT}/update-status/${fw_ver_num}/firmware/${sn}/${fw_component}-
${code}" &
        fi
        if [ ! -z "${ADDRESS_3}" ]; then
            /usr/bin/timeout -t ${TIMEOUT} /usr/bin/wget -q -O /dev/null
"${PROTOCOL}://${ADDRESS_3}:${PORT}/update-status/${fw_ver_num}/firmware/${sn}/${fw_component}-
${code}" &
        fi
   fi
}
```

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
```