Cisco Field Network Director

TPD Firmware File Format Specification

Location: https://github.com/CiscoDevNet/csmp-agent-lib/tree/osal/docs/Cisco_FND_TPD_FirmwareFormat_Spec.pdf

Modification History

Revision	Date	Originator	Comments
1	07 Feb 2025	Paul Duffy	Initial draft.
2	28 Feb 2025	Manojna CSL	Updated initial draft, sections 3 Firmware format, 3.1.1 App Header
3	03 Mar 2025	Manojna CSL	Added sections 5 add_tpdheader.py, 6 Glossary
4	28 Mar 2025	Manojna CSL	Updated sections 3, 3.3, 5.1. Added sections 5.2, 6

Table of Contents

1 Introduction

This document describes the structure of the Cisco Field Network Director (FND) Third Party Device (TPD) firmware file format which encapsulates a FND and OpenCSMP readable header with a third party firmware image for delivery to third party devices via FND and OpenCSMP firmware upgrade feature.

2 Conventions

All numeric attributes must be little endian encoded unless otherwise specified. Maximum size of the header is 256 bytes inclusive of a 4 byte CRC of the header.

3 Firmware File Format

An FND TPD firmware file consists of three sections:

1. **Header**: Metadata used by FND to process the firmware file. The size of the header is upto 252 bytes and a 4 byte CRC, totaling to a size of 256 bytes.

2. **Image**: The third party vendor binary image that will be processed by the target device's bootloader. This data is vendor specific and opaque to FND and OpenCSMP. Though there is no explicit limitation on the size of this binary image in FND, recommended to check for any limitations in the OpenCSMP agent for the respective target device platforms.

3. **Signature** (optional but recommended): It is recommended that the signature be included and that the target device validate the signature to confirm source of the firmware file and that the file has not been altered. The add_tpdheader.py helper script does not add this signature to the firmware file.

Figure 1 depicts the structure of a FND/OpenCSMP compatible TPD firmware file format

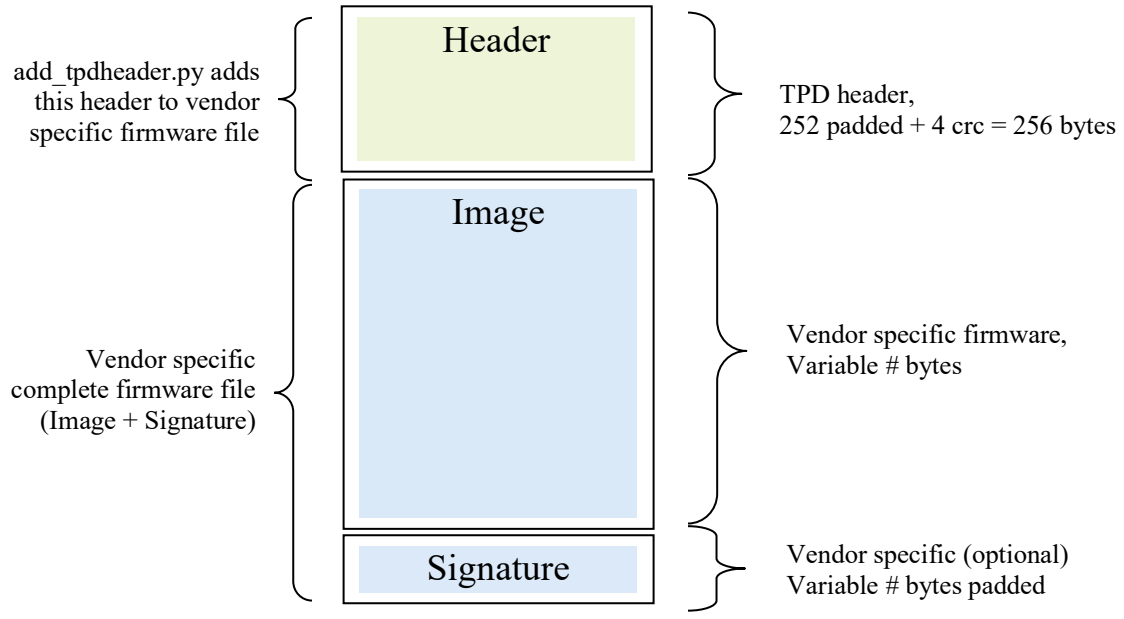

Figure 1 Firmware File Format

3.1 Header

The Header, which occupies the first 256 bytes of a firmware image file, consists of three components present in the following order:

1. The App Header (upto 252 bytes)
2. Padding (optional)
3. CRC (4 bytes)

3.1.1 App Header

The App Header contains meta-data used by FND to process the firmware file. The App Header must be structured as follows:

```
typedef struct {
  uint32_t hdr_version;
  uint32_t hdr_len;
  uint32_t app_rev_major;
  uint32_t app_rev_minor;
  uint32_t app_build;
  uint32_t app_len;
  char app_name[32];
  char app_git_branch[32];
  char app_git_commit[8];
  uint32_t app_git_flag;
  char app_build_date[16];
  char hwid[32];
  char sub_hwid[32];
  char kernel_rev[16];
} tpd_apphdr_t;
```

The fields of the app header must adhere to the definitions in the below Table 1.

hdr_version	Set to '2'
hdr_len	Set to '256'
app_rev_major	Integer representing the app version major number Eg: 6
app_rev_minor	Integer representing the app version minor number Eg: 8
app_build	Integer representing the app build number Eg: 99
app_len	Length of the firmware file INCLUDING header and image sections. DOES NOT include the signature or signature pad bytes.
app_name	String describing the application name Eg: TPD Firmware
app_git_branch	String indicating the source code control branch name Eg: master
app_git_commit	String indicating the source code control commit id Eg: 048841d
app_git_flag	Flag used with source code control and set to '1'
app_build_date	Build date of the form MMM DD YYYY Eg: Mar 15 2025
hwid	String indicating platform compatible with this firmware image. Eg: OPENCSMP Note: Ensure the same case-sensitive string is used for HardwareID in FND TPD onboarding metadata and in OpenCSMP Hardware Description TLV11
sub_hwid	String indicating a sub platform variant if any. Default value is 0x00
kernel_rev	String indicating the firmware kernel version if any

Table 1 Header Fields

3.1.2 Padding

The header must be padded to an entire length of 256 octets. The value of the pad byte must be 0x00.

3.1.3 CRC

The last 4 bytes of the header is a 32-bit Cyclic Redundancy Check (CRC). This CRC covers all App Header bytes, excludes header padding and the header CRC itself.
See section 4 for CRC calculation details.

3.2 Image

The image must immediately follow the header. The definition of the image content is left to the vendor and is opaque to FND and OpenCSMP. Upon delivery of a firmware file to a device via FND firmware upgrade feature, the device will typically decapsulate the vendor specific image from the firmware file and handover the image to the device bootloader for further processing.

The last 4 bytes of the header is a 32-bit Cyclic Redundancy Check (CRC) of the header excluding the padding and the CRC itself. See section 4 for CRC calculation details.

3.3 Signature

A vendor specific signature may immediately follow the image. It is optional but recommended that a signature be added to allow the device to authenticate the source of the firmware file and perform integrity check to confirm that the firmware file has not been altered. The signature must include firmware image bytes excluding the signature itself.

An algorithm such as ECDSA with SHA256 could be used for Signature verification. The signature block may be padded out to an arbitrary multiple of bytes to meet vendor specific requirements. However, the choice of algorithms, key-management, firmware verification process and bootloader implementation that reads the firmware image are all vendor specific.

Note:

- FND/OpenCSMP does not add or modify the firmware signature.
- FND/OpenCSMP does not verify firmware signature.
- FND/OpenCSMP does not take any action based on firmware signature.

4 CRC algorithm

The CRC used is a 32-bit CRC with the following generator polynomial:

$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

The CRC is computed in the normal mode (most significant byte first) and is initialized to 0 (zero) at the start.

The following sections show programmatic examples of computing the CRC in C and Python languages.

4.1 C language

The following function updates a CRC from a single byte in a stream.

```
uint32_t crc32MSBByte(uint32_t crc,
                      const uint8_t byte) {
  uint32_t i;
  crc = crc ^ ((uint32_t)byte << 24);

  for (i = 8; i > 0; i--) {
    if (crc & 0x80000000)
      crc = (crc << 1) ^ (0x04C11DB7);
    else
      crc = (crc << 1);
  }
  return crc;
}
```

4.2 Python

```
def crc32_msb_byte(rpolynom, crc, byte):
    crc = crc ^ ((byte << 24) & 0xFFFFFFFF)
    for i in range(8):
        if crc & 0x80000000 != 0:
            crc = ((crc << 1) & 0xFFFFFFFF) ^ rpolynom
        else:
            crc = ((crc << 1) & 0xFFFFFFFF)
    return crc
```

5 Creating a Cisco FND TPD Firmware file

5.1 add_tpdheader.py

This python helper script accepts a third party firmware binary image file, a configuration file in JSON format and adds a FND/OpenCSMP compatible header, pad optional bytes if needed, adds header CRC as discussed in the document to the input firmware image file to create an output firmware image with. This header enables third party firmware image to be uploaded into FND and transferred to end device via FND firmware upgrade feature.

The usage instructions for this helper script as indicated below

```
$ python3 add_tpdheader.py -h
usage: add_tpdheader.py [-h] input_file output_file config_file

Add Cisco FND compatible header to TPD firmware image

positional arguments:
  input_file   Input TPD binary file
  output_file  Output binary file with TPD header
  config_file  JSON configuration file

options:
  -h, --help   show this help message and exit


Eg:
$ python3 add_tpdheader.py <input-image> <output-image> <json-config-file>
```

This script is located at https://github.com/CiscoDevNet/csmp-agent-lib/tree/osal/tools

5.2 TPD configuration JSON file

add_tpdheader.py script accepts a configuration file in JSON format indicating the values for various fields of the TPD header to be added to the input firmware image.

A sample TPD configuration file in JSON with various fields and values is illustrated below:

```
{
    "major": "7",
    "minor": "1",
    "build": "2",
    "name": "CsmpAgent_Linux",
    "hwid": "OPENCSMP",
    "sub_hwid": "0",
    "branch": "osal",
    "commit": "048841d",
    "date": "Mar 15 2025",
    "kernelrev": "WISUN1.0"
}
```

Note:

- Fields **major, minor, build** indicate respective numbers of the firmware version. They must be non-zero positive integers.
- Field **name** refers to a readable name of the firmware.
- Field **hwid** refers to the hardware-id of the compatible device/platform. The default hwid is OPENCSMP. Note the hwid in the firmware image is case-sensitive and should match the hwid used in the FND metadata while onboarding.
- Field **date** typically indicates firmware build date, if not specified current date is used.

A sample config file is located at https://github.com/CiscoDevNet/csmp-agent-lib/tree/osal/tools

6 Caveats

1. Fields major, minor, build numbers must be non-zero positive integers.
2. hwid field in the firmware image is case-sensitive and should match the hwid used in the FND metadata while onboarding.

7 Glossary

1.	FND	Cisco Field Network Director
2.	OpenCSMP	Opensource CSMP Agent library
3.	CSMP	CoAP Simple Management Protocol
4.	CoAP	Constrained Application Protocol
5.	TPD	Third Party Device
6.	CRC	Cyclic Redundancy Check
7.	TLV	Type Length Value

End of Document