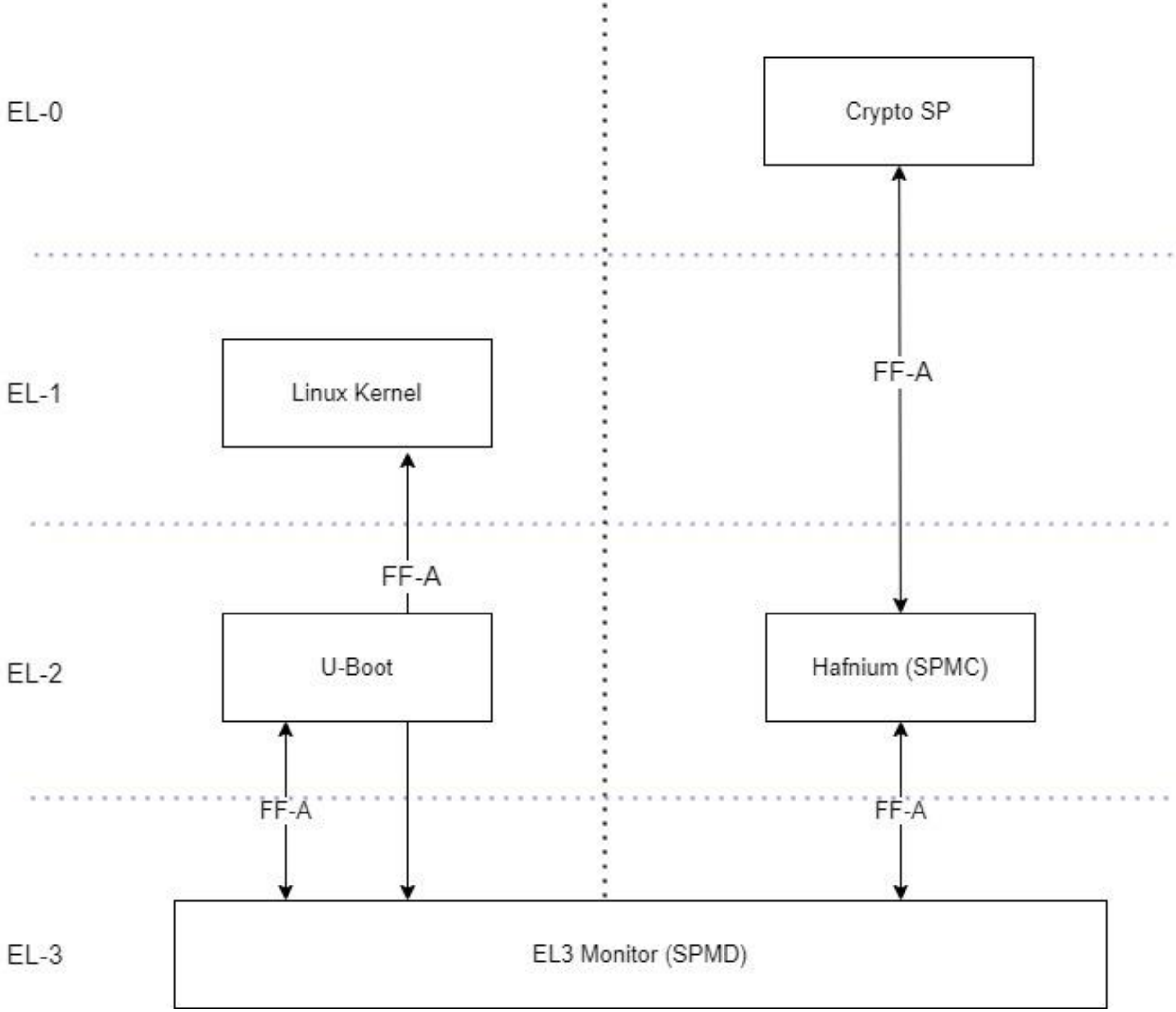
Firmware Update in Total Compute Platform

Manish Badarkhe and Davidson Kumaresan
22/09/2022

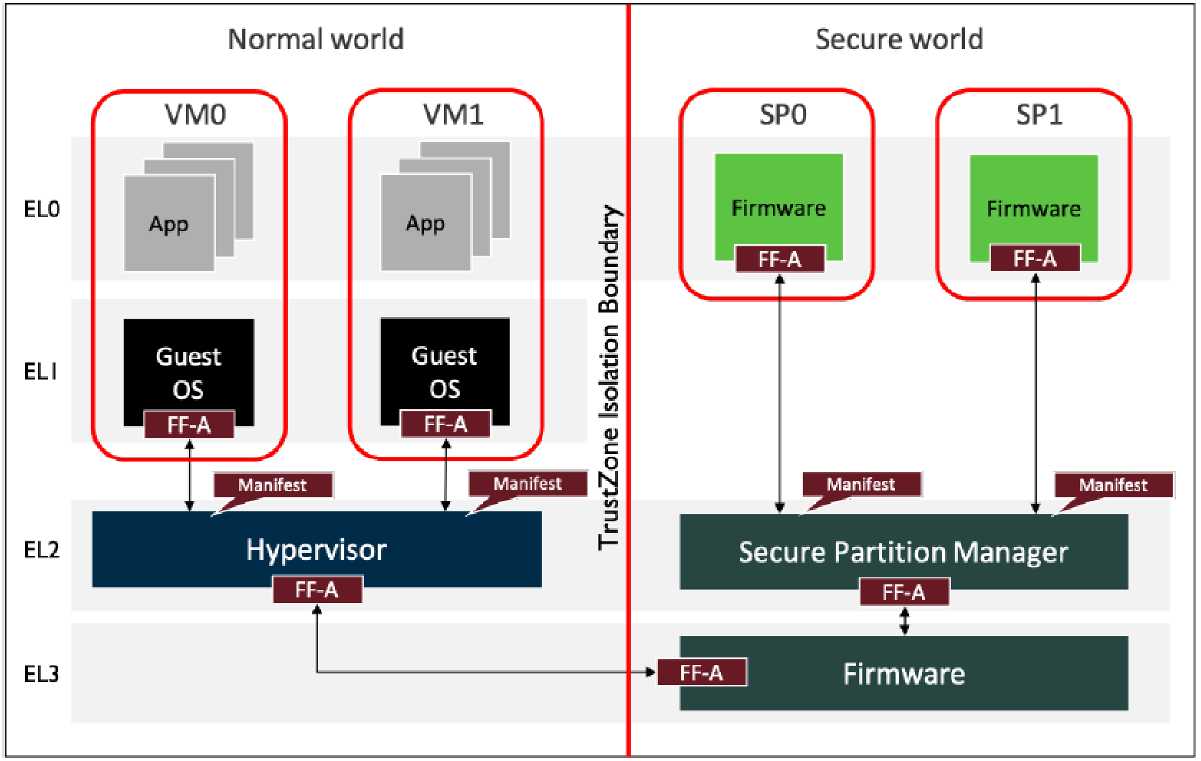
Agenda

- Total compute software stack
- Firmware update spec revision
- EFI UpdateCapsule runtime service
- Call flow
- Work yet to be done

TC Software Stack



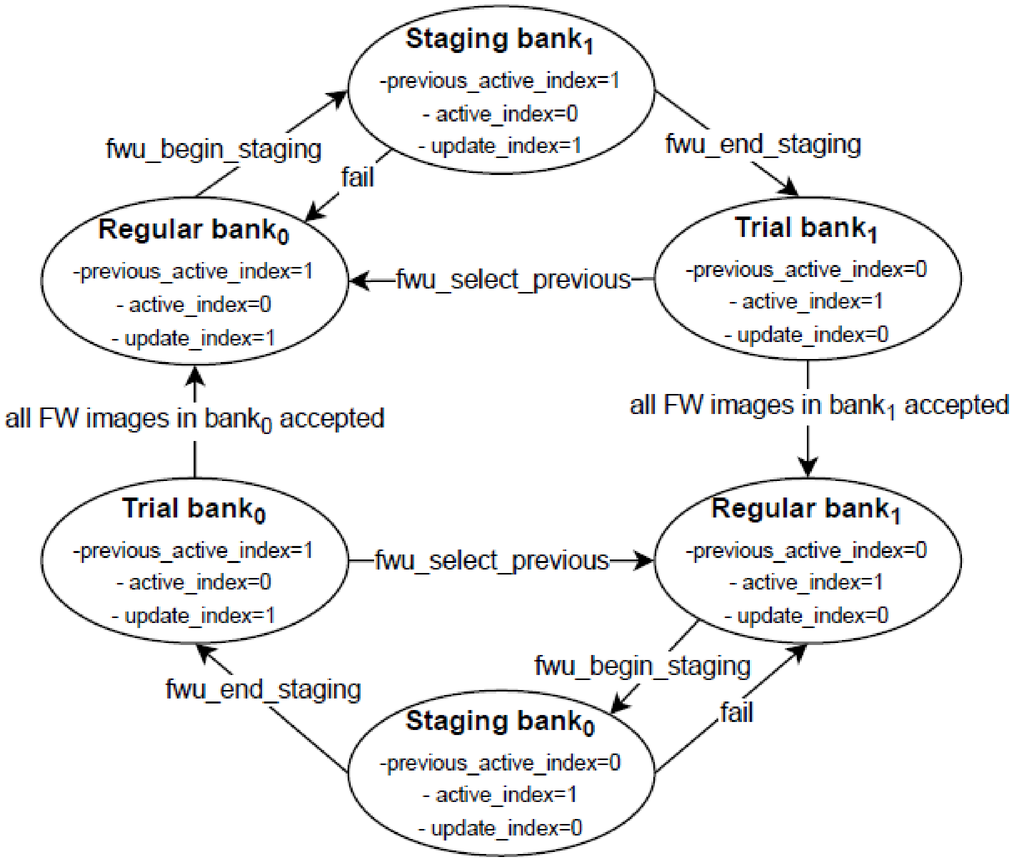
FF-A



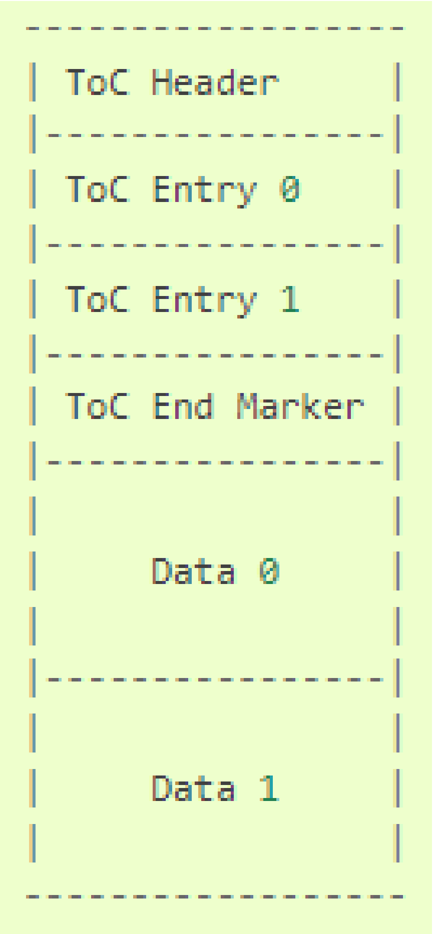
- Setup and Discovery interfaces
- Scheduling interfaces
- Messaging interfaces
- Memory management interfaces
- Notification interfaces
- Status reporting interfaces

PSA Firmware Update terminology

- Banks
- Image Directory
- Staging area
- States



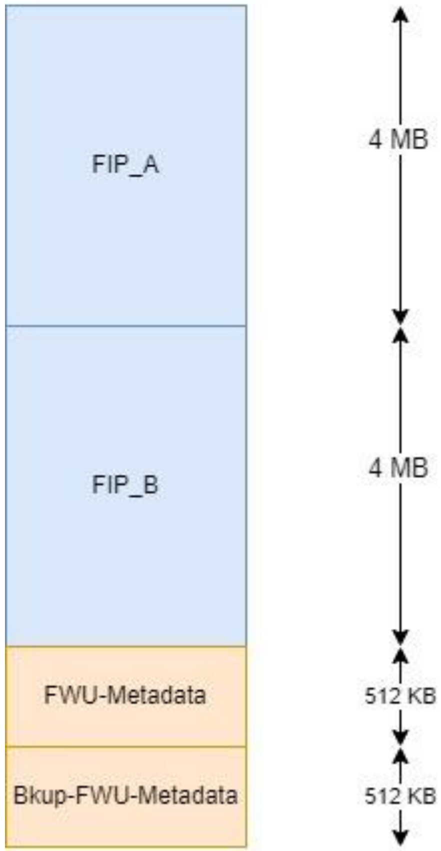
Firmware - FIP



```
------------------
| ToC Header      |
|-----------------|
| ToC Entry 0     |
|-----------------|
| ToC Entry 1     |
|-----------------|
| ToC End Marker  |
|-----------------|
|                 |
|     Data 0      |
|                 |
|-----------------|
|                 |
|     Data 1      |
|                 |
------------------
```

- FIP in TC has
  - BL2 (loader)
  - SCP_RAMFW
  - BL31 (secure monitor)
  - BL32 (Hafnium)
  - Secure Partitions – Trusted services and trusted OS (trusty/optee)
  - BL33 (U-Boot)
- This FIP is referred as firmware here and the images present in the FIP are upgradable.

Partition Layout



FIP_A

4 MB

FIP_B

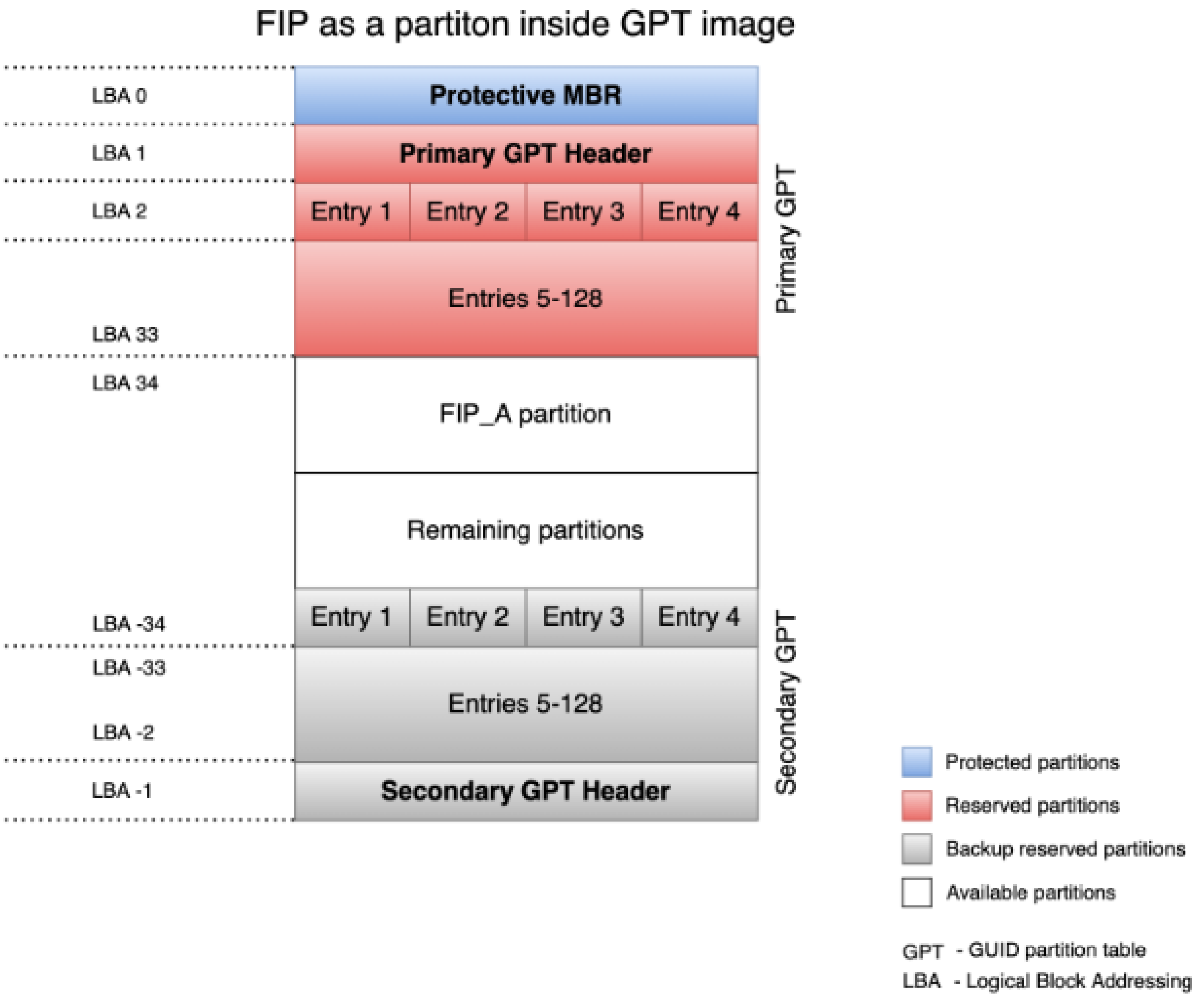
4 MB

FWU-Metadata

512 KB

Bkup-FWU-Metadata

512 KB



FIP as a partiton inside GPT image

LBA 0 — Protective MBR

LBA 1 — Primary GPT Header

LBA 2 — Entry 1, Entry 2, Entry 3, Entry 4

Entries 5-128

LBA 33

LBA 34 — FIP_A partition

Remaining partitions

LBA -34 — Entry 1, Entry 2, Entry 3, Entry 4

LBA -33 — Entries 5-128

LBA -2

LBA -1 — Secondary GPT Header

Primary GPT

Secondary GPT

🟦 Protected partitions

🟥 Reserved partitions

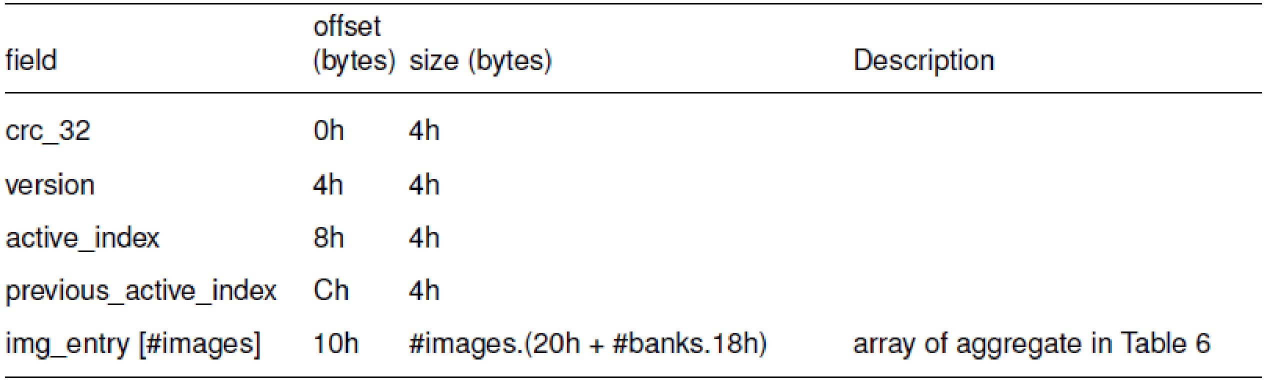
⬜ Backup reserved partitions

□ Available partitions

GPT - GUID partition table
LBA - Logical Block Addressing

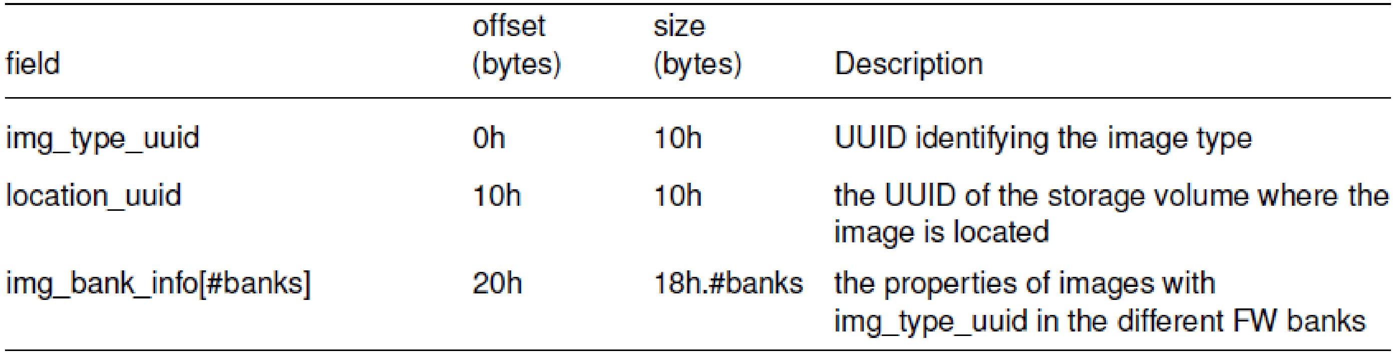
Metadata

Table 5: Metadata version 1



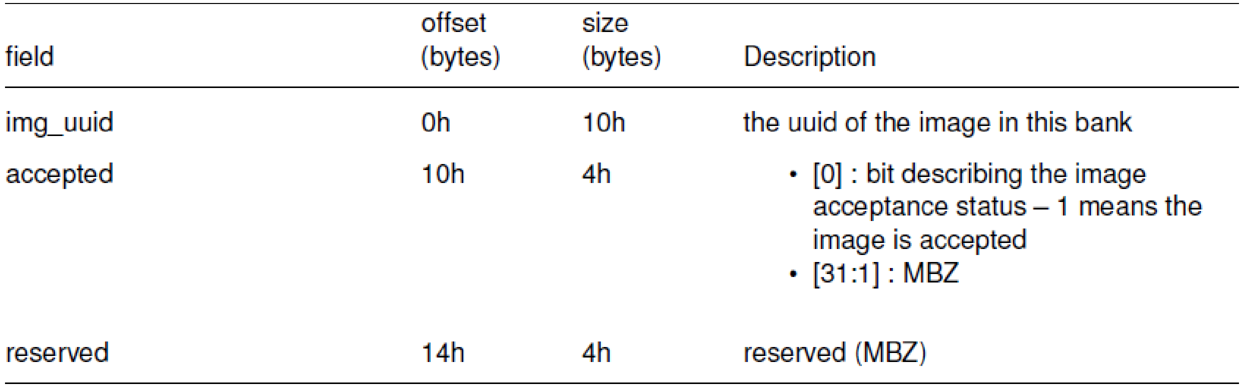
field	offset (bytes)	size (bytes)	Description
crc_32	0h	4h	
version	4h	4h	
active_index	8h	4h	
previous_active_index	Ch	4h	
img_entry [#images]	10h	#images.(20h + #banks.18h)	array of aggregate in Table 6

Table 6: Metadata image entry version 1



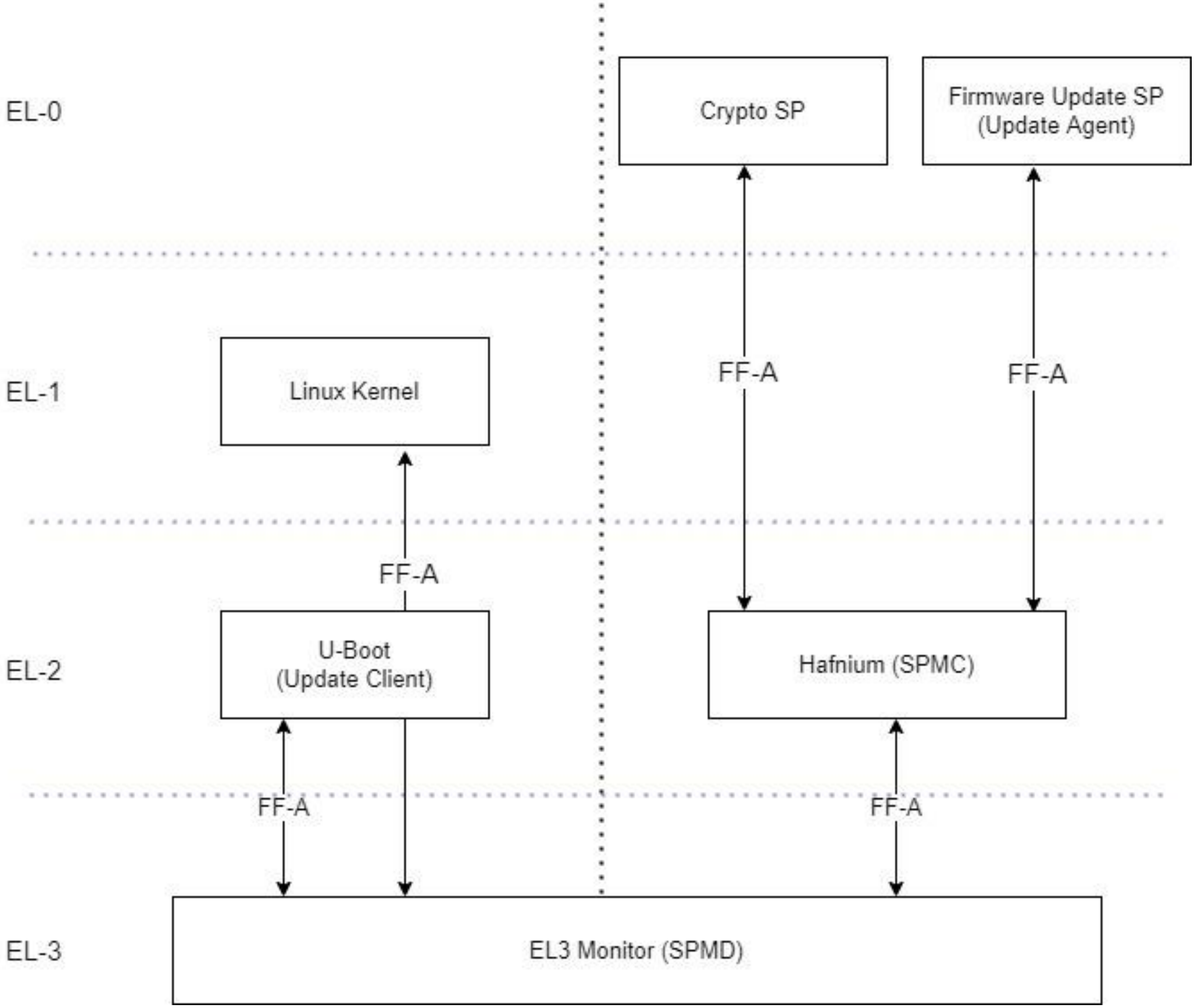
field	offset (bytes)	size (bytes)	Description
img_type_uuid	0h	10h	UUID identifying the image type
location_uuid	10h	10h	the UUID of the storage volume where the image is located
img_bank_info[#banks]	20h	18h.#banks	the properties of images with img_type_uuid in the different FW banks

Table 7: Image properties in a given FW bank version 1

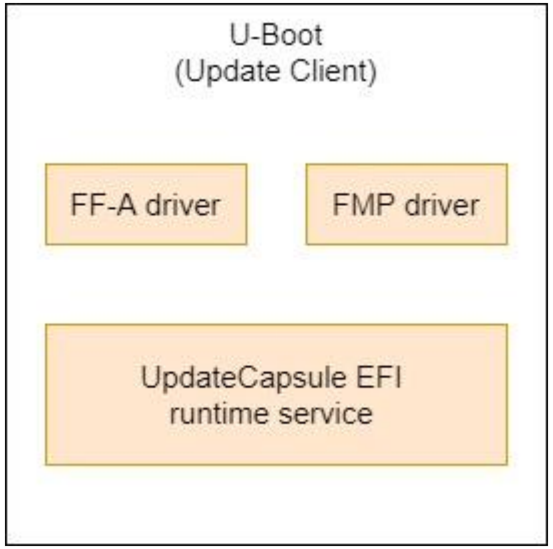


field	offset (bytes)	size (bytes)	Description
img_uuid	0h	10h	the uuid of the image in this bank
accepted	10h	4h	• [0] : bit describing the image acceptance status – 1 means the image is accepted • [31:1] : MBZ
reserved	14h	4h	reserved (MBZ)

TC software stack for Firmware Update

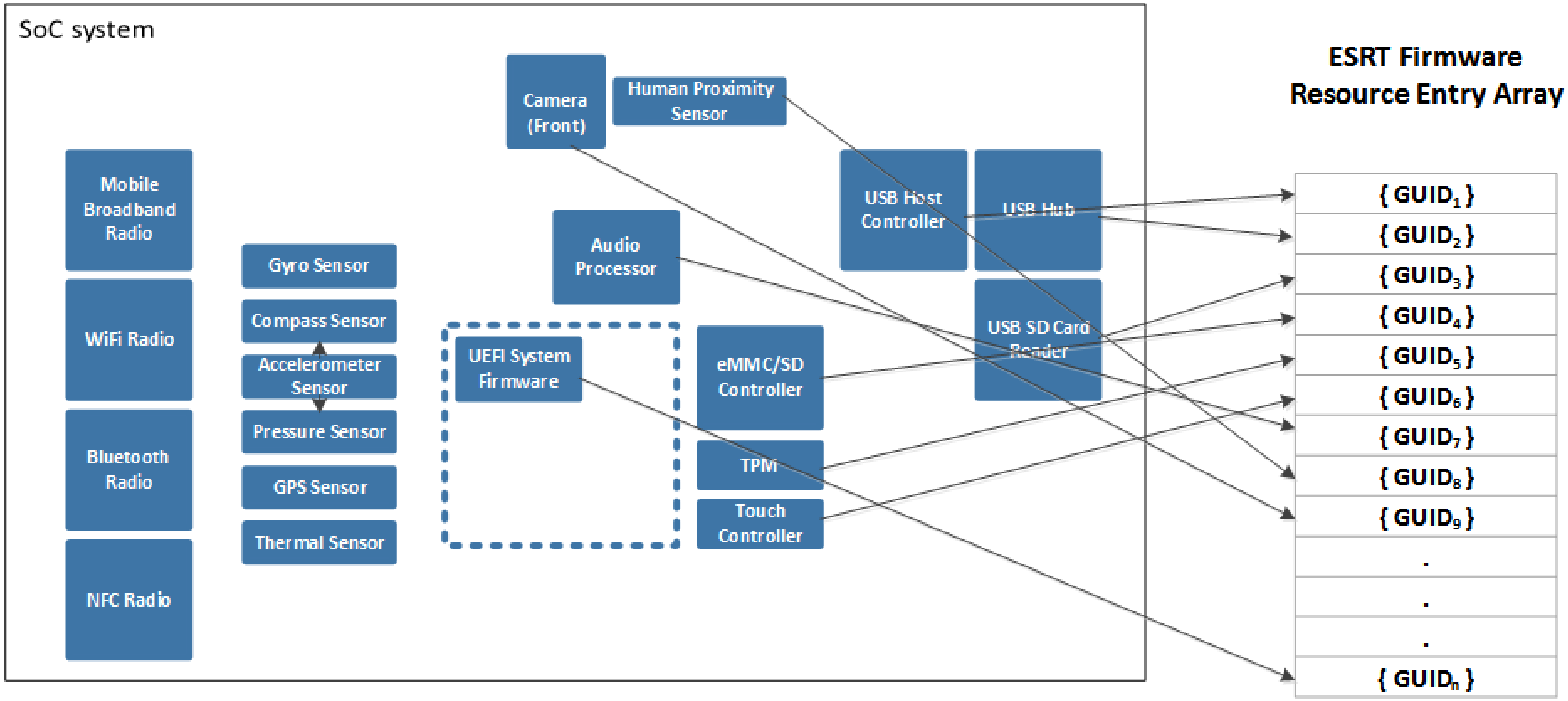


U-Boot



- It is the update client
- Use the UpdateCapsule runtime service of EFI

EFI System Resource Table (ESRT)



SoC system containing *n* firmware resources

Each of the firmware resources maps to an entry in the ESRT Firmware Resource Entry Array. (Note that other data associated with each entry is not shown for simplicity. Refer to ESRT implementation documentation for details.)

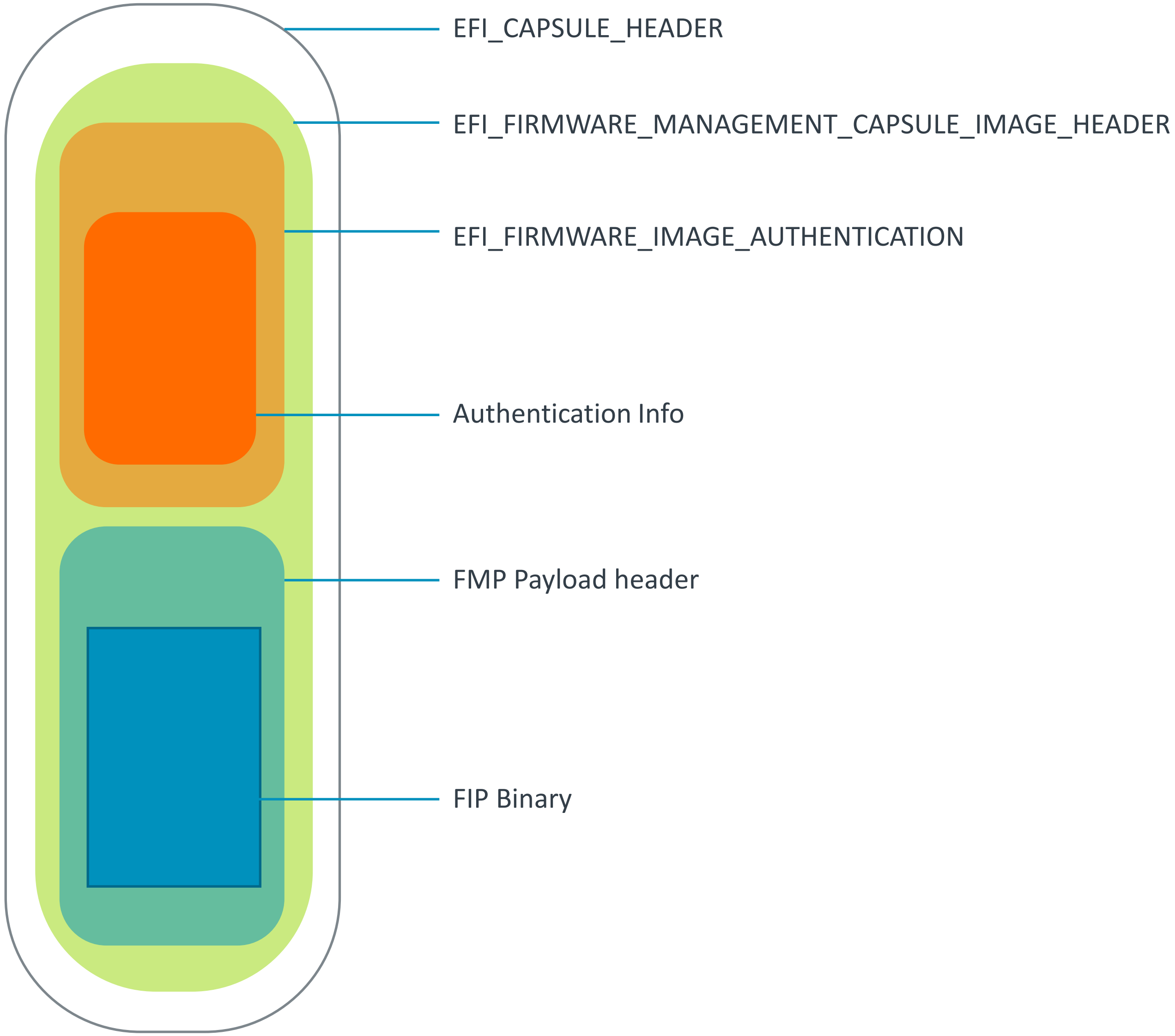
Firmware Management Protocol APIs

- GetImageInfo
- GetImage
- SetImage
- CheckImage
- GetPackageInfo
- SetPackageInfo

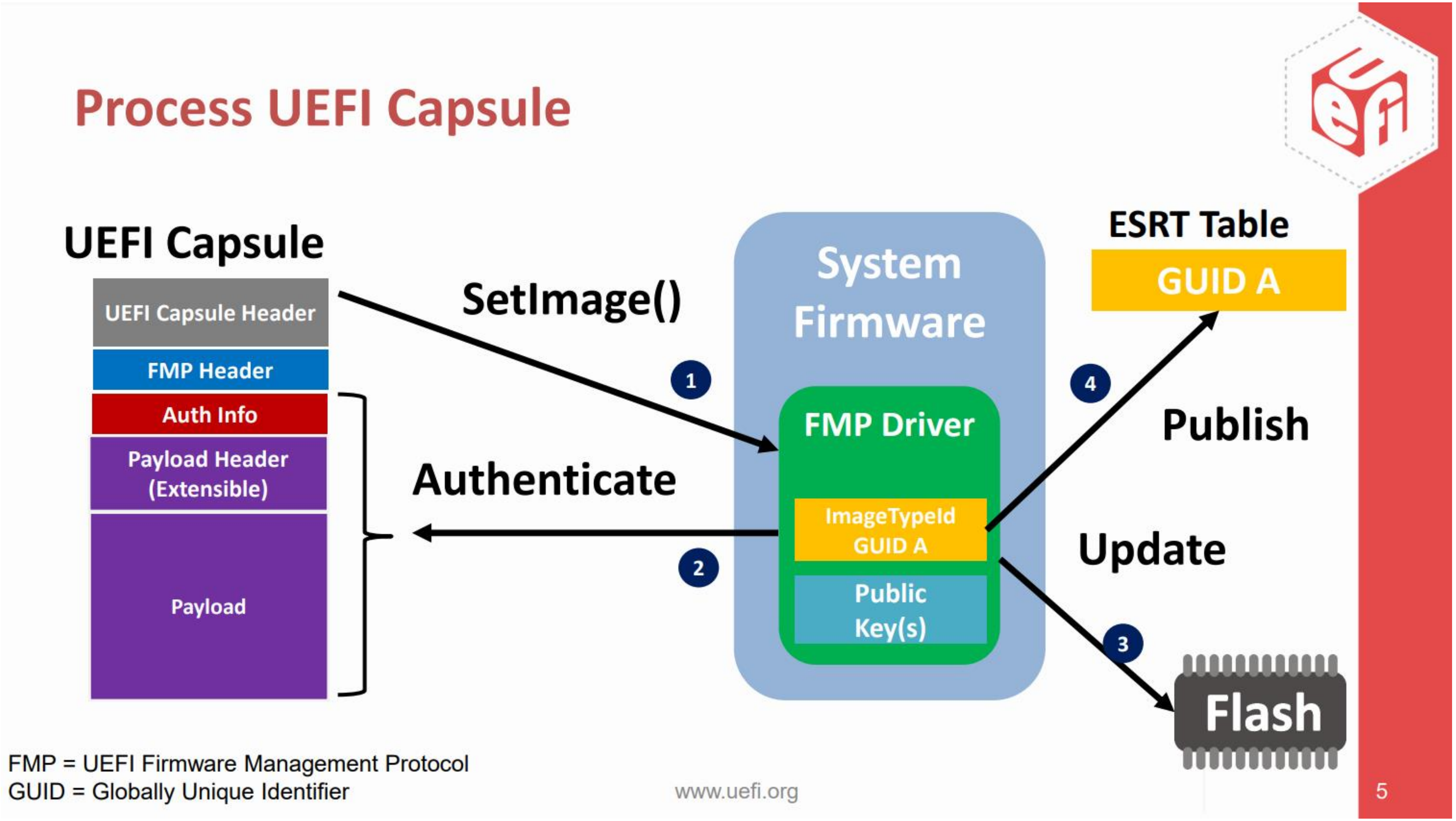
Capsules

- To do capsule update, we have to convert the FIP image into a capsule
- To create capsule, GenerateCapsule from the edk2 project has to be used

```
edk2/BaseTools/BinWrappers/PosixLike/GenerateCa
psule -e -o efi_capsule --fw-version 1 --lsv 0
--guid 0d5c011f-0776-5b38-8e81-36fbdf6743e2 --
verbose --update-image-index 0 --verbose fip-
tc.bin
```


EFI_CAPSULE_HEADER
EFI_FIRMWARE_MANAGEMENT_CAPSULE_IMAGE_HEADER
EFI_FIRMWARE_IMAGE_AUTHENTICATION
Authentication Info
FMP Payload header
FIP Binary

EFI UpdateCapsule



Process UEFI Capsule

UEFI Capsule

UEFI Capsule Header

FMP Header

Auth Info

Payload Header (Extensible)

Payload

SetImage()

1

Authenticate

2

System Firmware

FMP Driver

ImageTypeId GUID A

Public Key(s)

ESRT Table

GUID A

4

Publish

Update

3

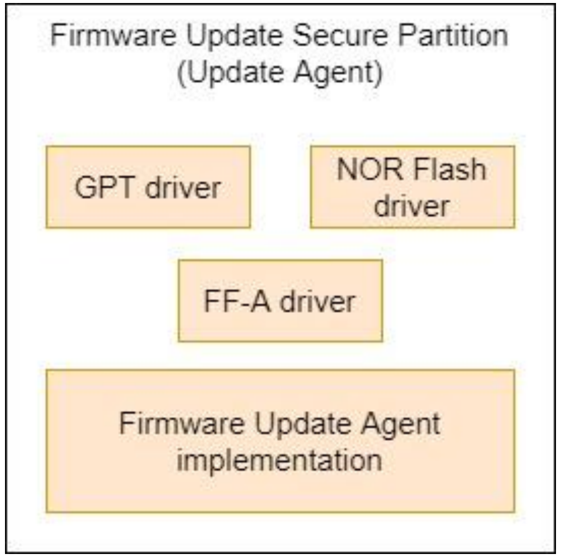
Flash

FMP = UEFI Firmware Management Protocol
GUID = Globally Unique Identifier

www.uefi.org

5

Firmware Update Secure Partition



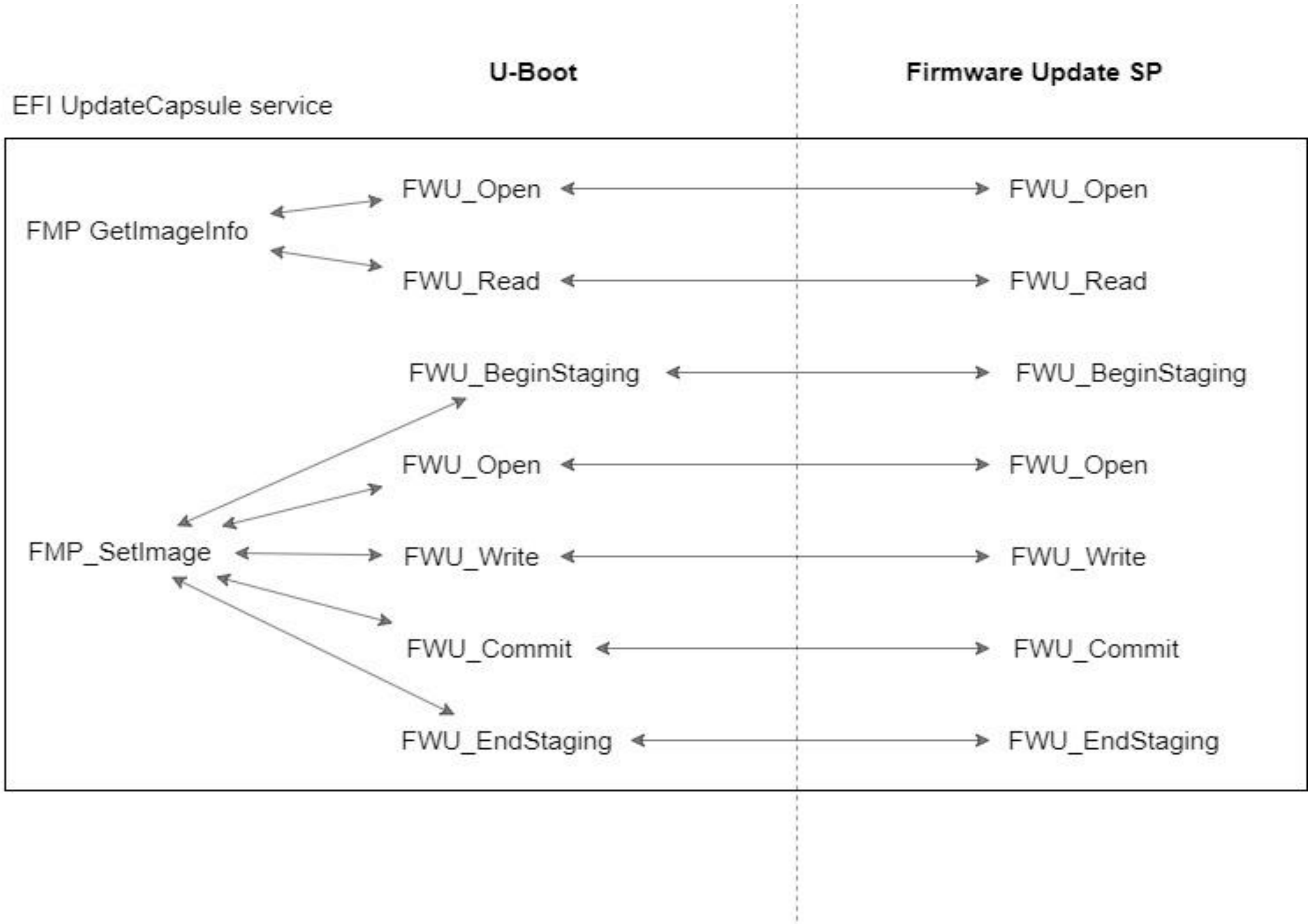
Firmware Update Secure Partition (Update Agent)
GPT driver
NOR Flash driver
FF-A driver
Firmware Update Agent implementation

- It is implemented in the TrustedServices project which is maintained in the https://git.trustedfirmware.org/
- It will run at S-EL0 as a secure partition
- It is the implementation of the arm PSA firmware update specification. It implements all the PSA firmware update APIs.
- NOR flash driver required to read and write to the NOR flash device.
- GPT partition driver required to parse the GPT partition header and to get the partition information from the flash.

Firmware Update APIs

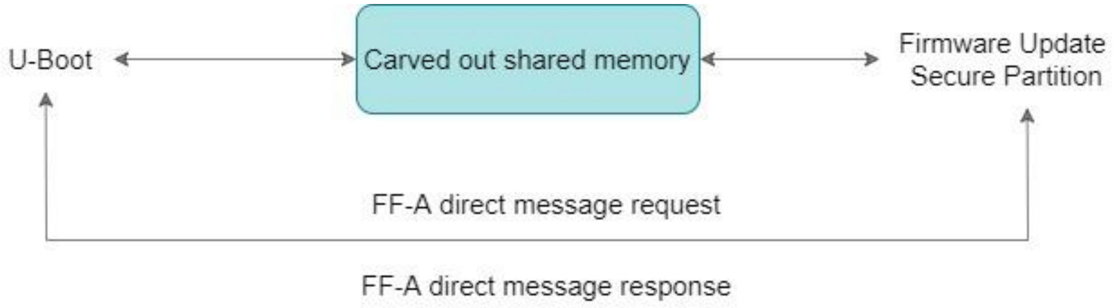
- Discover
- BeginStaging
- EndStaging
- CancelStaging
- Open
- WriteStream
- ReadStream
- Commit
- AcceptImage
- SelectPrevious

Communication

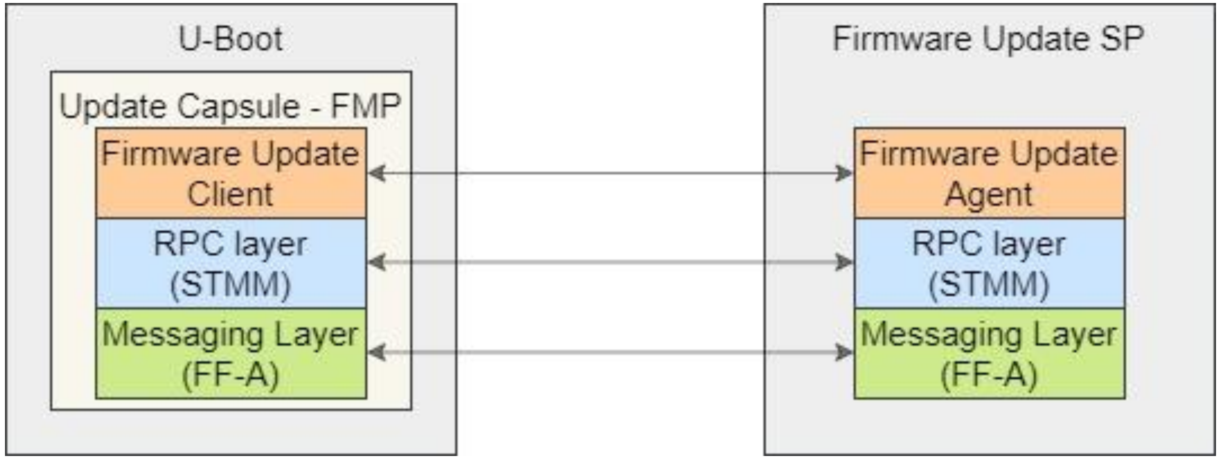


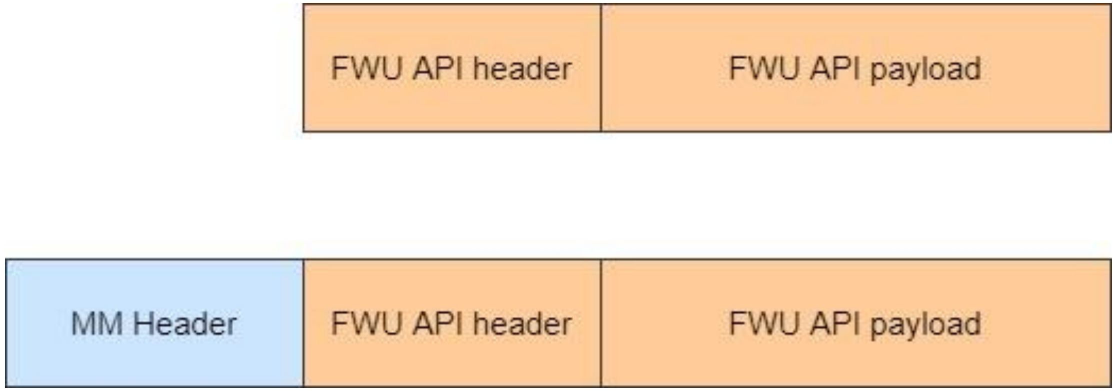
Communication

- U-Boot places the message into the shared memory and sends a ffa_direct_msg_req() to the Firmware Update secure partition
- Firmware Update secure partition receives the ffa_direct_msg_req() and reads the content from the shared memory.
- Firmware Update secure partition performs the action mentioned in the message and puts the reply message in the shared buffer and sends the ffa_direct_msg_resp()

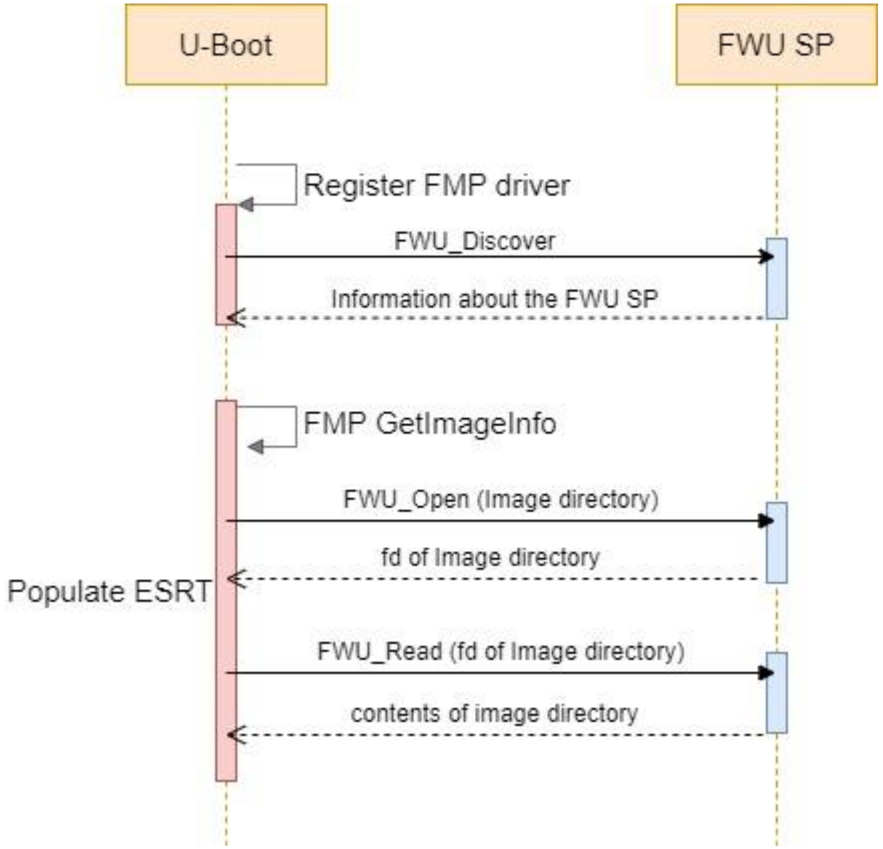

U-Boot
Carved out shared memory
Firmware Update Secure Partition
FF-A direct message request
FF-A direct message response

Communication

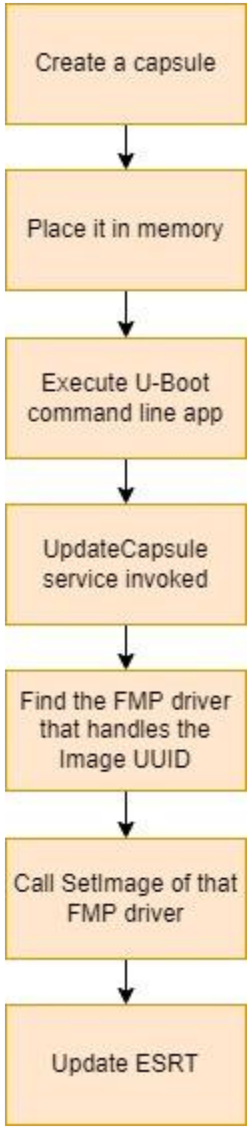




Call Flow – During Boot


U-Boot
FWU SP
Register FMP driver
FWU_Discover
Information about the FWU SP
FMP GetImageInfo
FWU_Open (Image directory)
fd of Image directory
Populate ESRT
FWU_Read (fd of Image directory)
contents of image directory

Call flow – During update



Create a capsule

Place it in memory

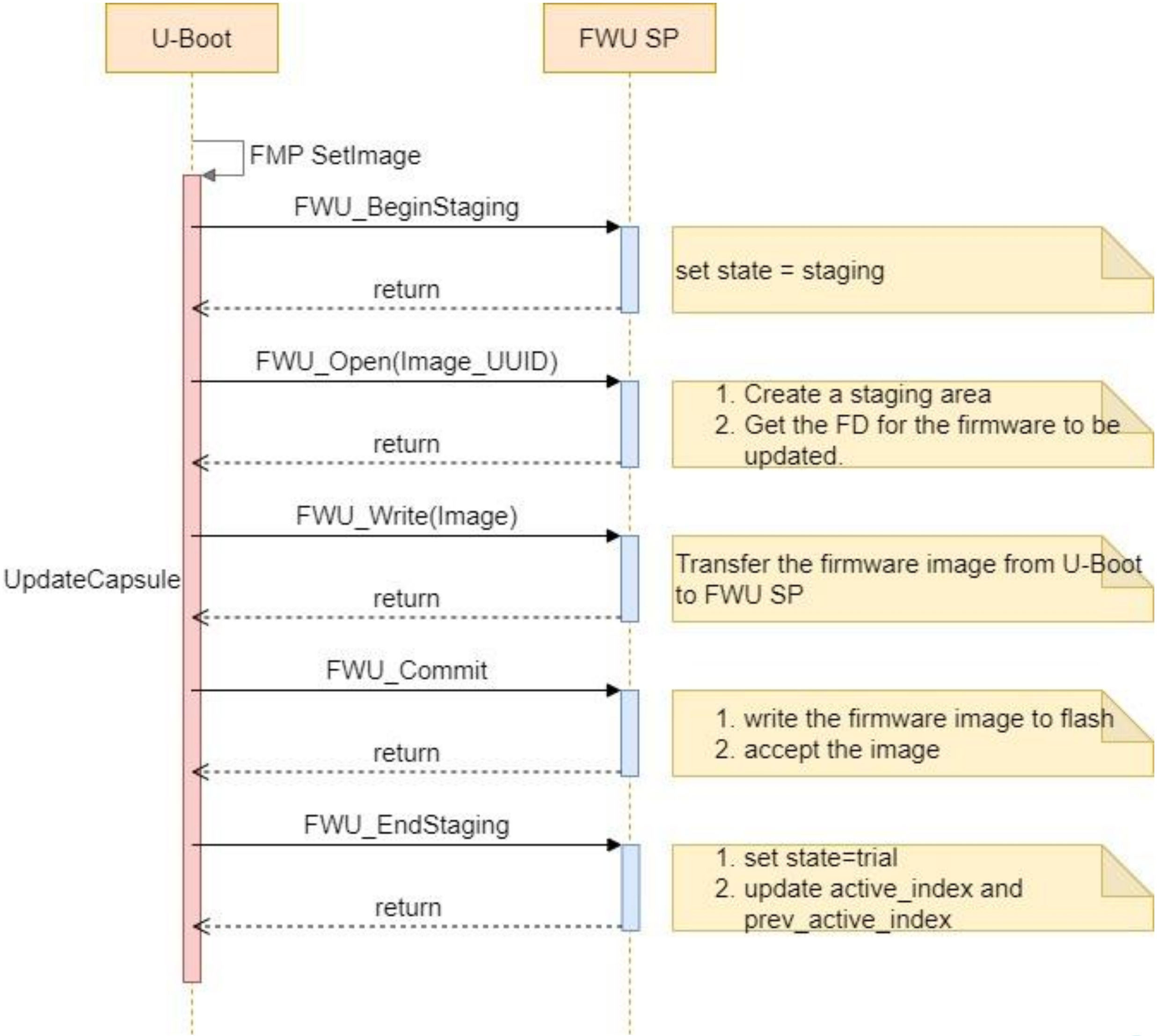
Execute U-Boot command line app

UpdateCapsule service invoked

Find the FMP driver that handles the Image UUID

Call SetImage of that FMP driver

Update ESRT



U-Boot

FWU SP

UpdateCapsule

FMP SetImage

FWU_BeginStaging

return

set state = staging

FWU_Open(Image_UUID)

return

1. Create a staging area
2. Get the FD for the firmware to be updated.

FWU_Write(Image)

return

Transfer the firmware image from U-Boot to FWU SP

FWU_Commit

return

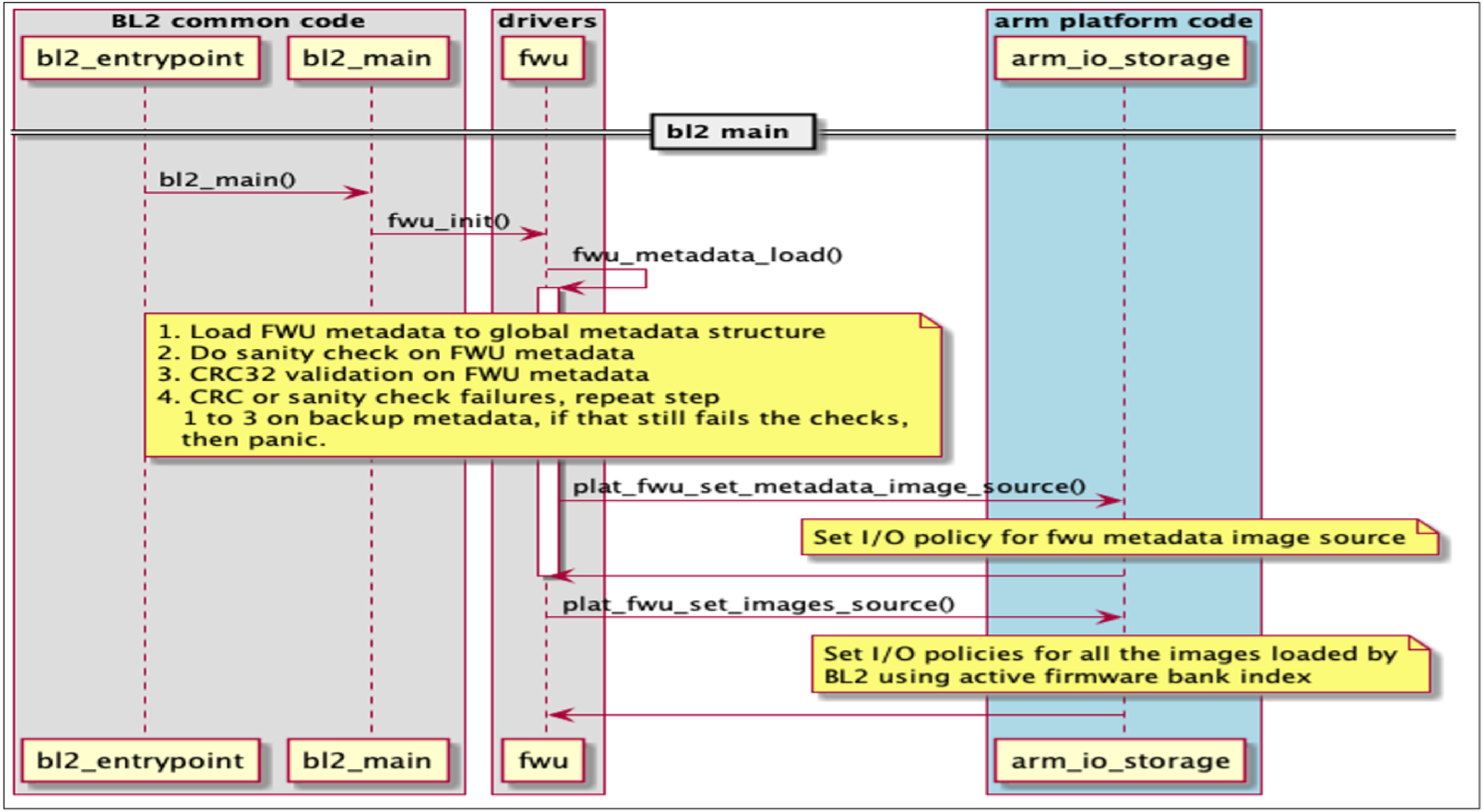
1. write the firmware image to flash
2. accept the image

FWU_EndStaging

return

1. set state=trial
2. update active_index and prev_active_index

Boot Flow 1/2 – BL2 Execution

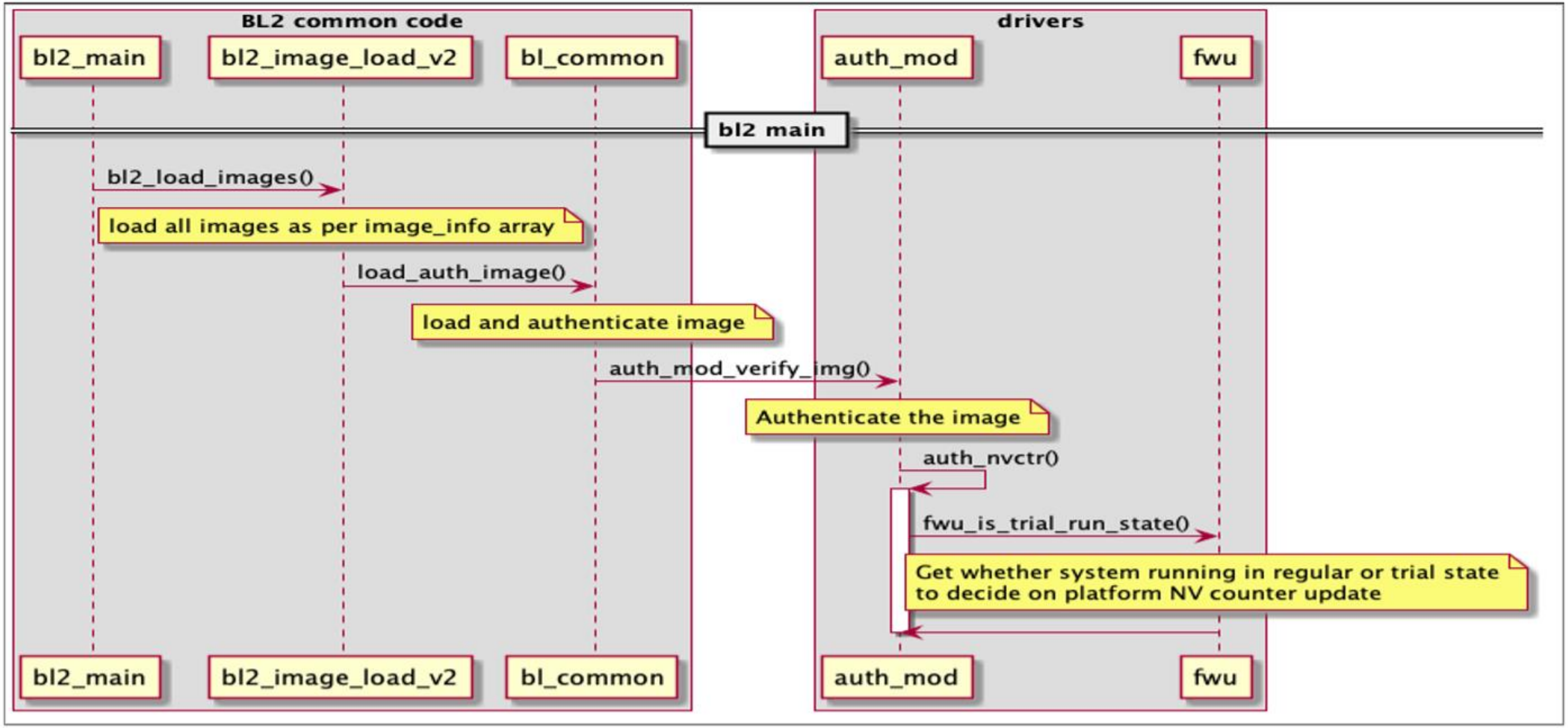


1. Load FWU metadata to global metadata structure
2. Do sanity check on FWU metadata
3. CRC32 validation on FWU metadata
4. CRC or sanity check failures, repeat step 1 to 3 on backup metadata, if that still fails the checks, then panic.

Set I/O policy for fwu metadata image source

Set I/O policies for all the images loaded by BL2 using active firmware bank index

Boot Flow 2/2 – Trial run detection



Features to be implemented

- Actual runtime update via OTA
- UEFI capsule authentication from UEFI spec
- PSA Image authentication from PSA spec
- Anti rollback counter from PSA spec
- No acceptance tests for the firmware - the newly updated firmware is implicitly accepted - from PSA spec
- Selecting previous image/bank from PSA spec
- Recovery mode from PSA spec

Reference

- ARM PSA firmware update spec
  https://developer.arm.com/documentation/den0118/a/?lang=en
- UEFI Spec - https://uefi.org/sites/default/files/resources/UEFI_Spec_2_7.pdf
  - EFI runtime services – Sec 8
  - UpdateCapsule – Sec 8.5.3
  - Firmware Management Protocol – Sec 23
  - ESRT – Sec 23.3
  - GPT partition layout – Sec 5
- Trusted Services Project - https://git.trustedfirmware.org/TS/trusted-services.git/
- Source code and patches will be part of the next Total Compute release.

Thank You

Danke

Gracias

Grazie

谢谢

ありがとう

Asante

Merci

감사합니다

धन्यवाद

Kiitos

شكرًا

ধন্যবাদ

תודה