Firmware Update Support in the Trusted Services Project

Reusable components for implementing the FWU capability in firmware for Arm A-Profile devices

Julian Hall
Jan 2023

Introduction

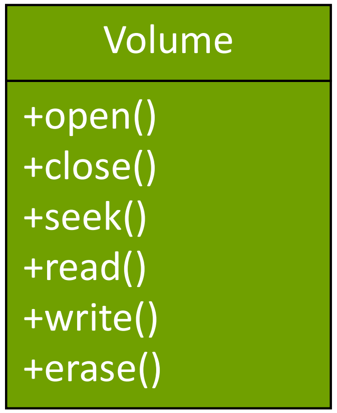
- Firmware update support has recently been added to the Trusted Services project.
- Consists of firmware components for implementing the Update Agent role defined in the Arm FWU-A specification.
- The Update Agent is intended to be used with third-party update managers to provide an end-to-end firmware update solution.
- Support includes:
  - Core Update Agent and FW Store components
  - Installer framework for supporting different image types
  - Service provider and client for remote calling
  - Component and integration level test suites
  - Reference FWU deployment in Secure Partition
  - Alternative deployment in command-line application
  - Library (libfwu) version to make reuse easier by downstream projects

Feature Summary

- Banked A/B robust updates
- Transactional multi-component updates
- Partial update support (update targets specific components)
- Extensible installer framework
- Support for distributed firmware
- Trial and rollback to previous version
- Anti-rollback counter management
- Arm standardized signalling to bootloader via FWU Metadata structure
- Image directory to advertise updatable components
- Compatible with UEFI FMP capsule update model and ESRT

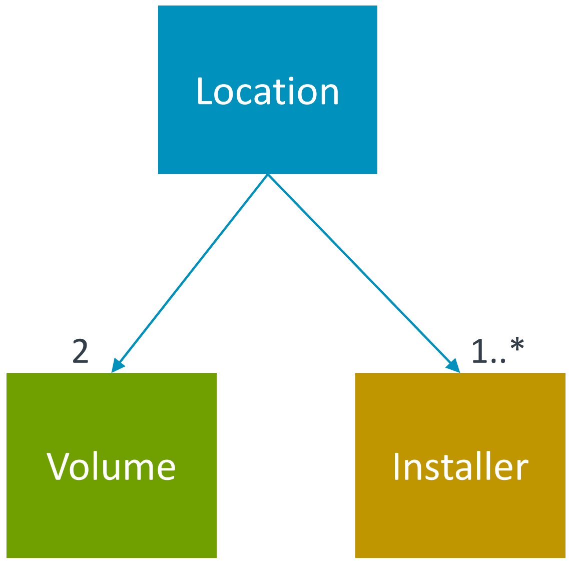
Storage Volumes

- All NV storage accessed by the Update Agent is represented by a set of *volumes*.
- A *volume* presents a unit of storage as a seekable file with standard file IO style operations.
- Volumes are used to access storage for FWU Metadata and firmware images.
- The *volume* reuses the TF-A io_dev driver model with extended erase capabilities.
- Any suitable io_dev driver from TF-A may be reused.
- The common *volume* interface can be used to access different types of storage such as:
  - A raw flash device
  - A flash partition (UEFI formatted flash with MBR/GPT supported)
  - Storage managed by a sub-processor
  - A file in a filesystem
- The TS project includes a Block Volume that is compatible with the Block Store interface.



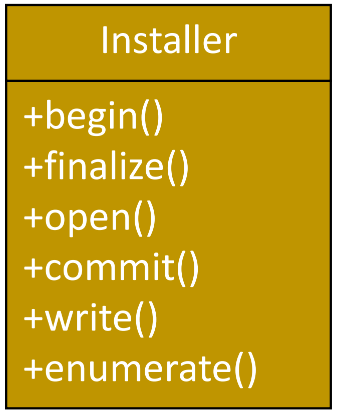
Firmware Locations

- The Update Agent can manage firmware distributed across multiple locations.
- A location binds together:
  - A pair of storage volumes (A + B banks)
  - A set of one or more installers
- There are three different types of installer:
  - **Whole volume installe**r – updates the entire contents of a volume.
  - **Sub-volume installer** – updates components contained within a volume.
  - **Whole volume copy installer** – copies from one volume to another
- A platform configuration defines the set of concrete Volumes and Installers, grouped by location, that are needed to update a device's firmware.

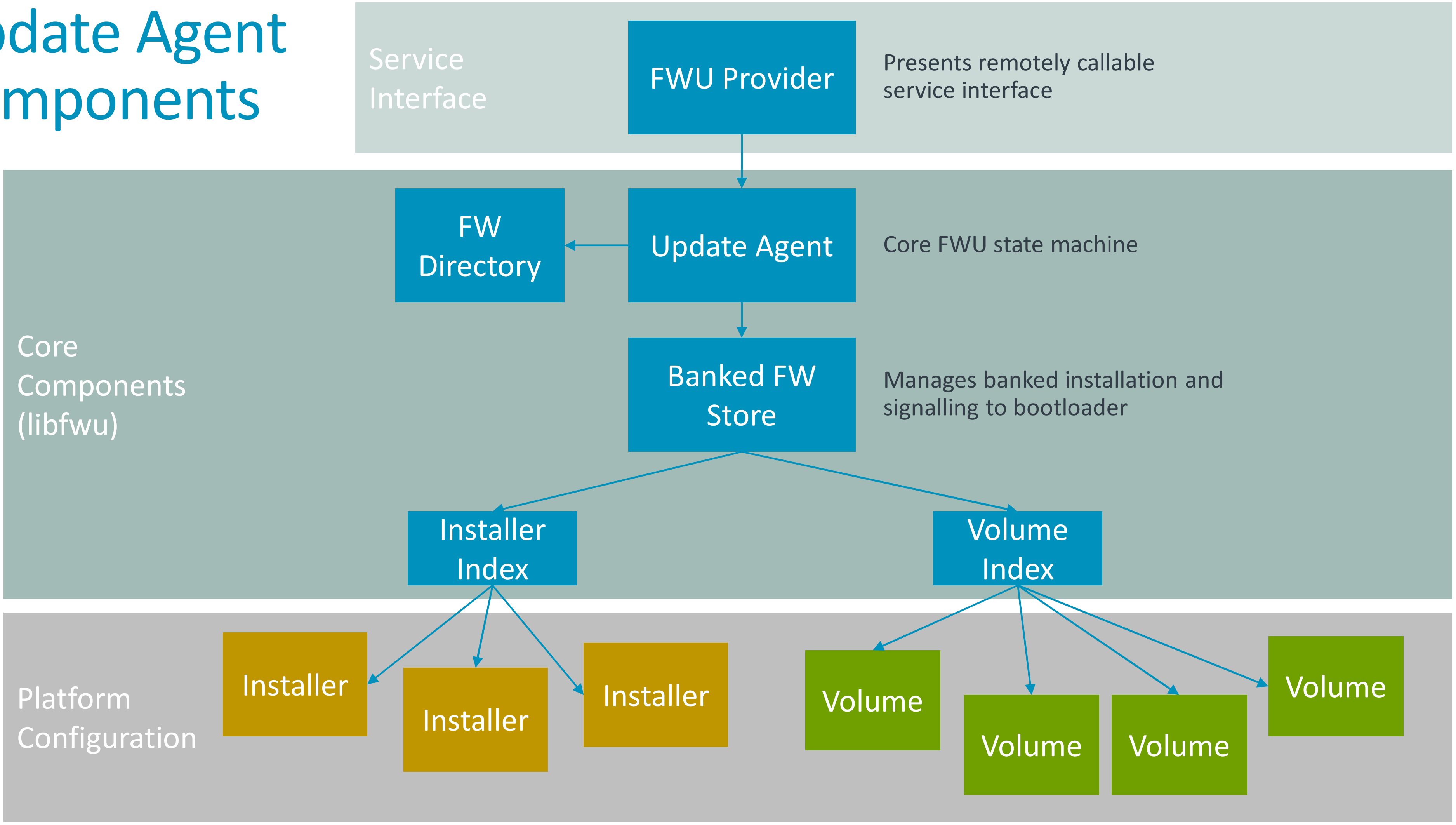


Installers

- An installer takes an input image and installs it into a target volume in some way.
- An installer presents a transactional interface where multiple images may be installed as a set.
- During an update transaction, multiple installers may be used in parallel.
- The common installer interface may be realized by different concrete installers.
- A concrete installer may have knowledge of the format of an image.
- An installer implements an *enumerate* method to return information about the images that it can handle. The contents of the Firmware Directory is formed by aggregating the information returned by each installer's enumerate method.
- Currently, we have the *raw_installer* and *copy_installer* in TS.
  - ***raw_installer*** – a whole volume installer that just copies an image directly into the target volume.
  - ***copy_installer*** – a whole volume copy installer that copies the contents of the currently active volume to the update volume.
- More installers are needed, such as:
  - ***fip_installer*** – a sub-volume installer that installs a subset of images into a FIP formatted volume. This will enable component-oriented updates of images within a FIP.



Update Agent Components

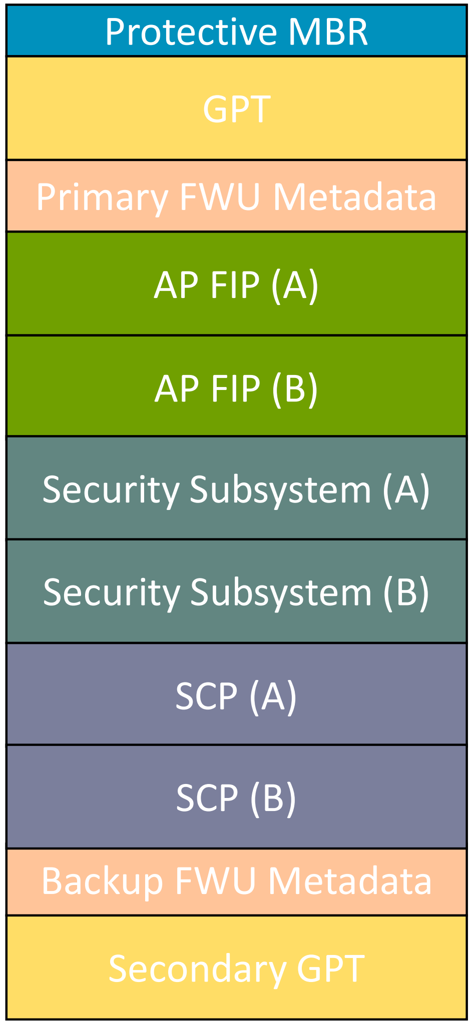


Communicating with the Bootloader

- The Update Agent informs the bootloader about an update using the FWU metadata.
- The FWU metadata structure is held in a NV storage volume.
- FWU metadata is written by the Update Agent and read by an early stage BL during system boot.
- The BL is responsible for passing the following values to the Update Agent during boot:
  - *Boot index* – the index of the bank used by the BL to boot from
  - *FWU metadata version* – the FWU metadata version that the BL understands
- A replica of the FWU metadata is maintained by the Update Agent to defend against corruption due to power failure.

GPT Based Configuration

- For UEFI platforms, use of a GUID Partition Table (GPT) to describe flash layout is common.
- Updatable firmware can be distributed across multiple partitions.
- FWU support includes partition discovery for configuring volumes and installers using information from the partition table.
- Reduces the need for build-time configuration.



Deploying the Update Agent

- To perform firmware updates, the Update Agent needs to access storage volumes that contain firmware and FWU metadata.
- The flash storage hardware architecture of a device strongly influences where the Update Agent can be located.
- FWU components in the TS project are designed for reuse in different execution environments.
- Some deployment options for the Update Agent:
  - In secure partition (devices with dedicated Swd flash)
  - As UEFI MM service
  - In UEFI FMP driver
  - In UEFI app
  - In OS kernel driver

FWU Deployments in Trusted Services

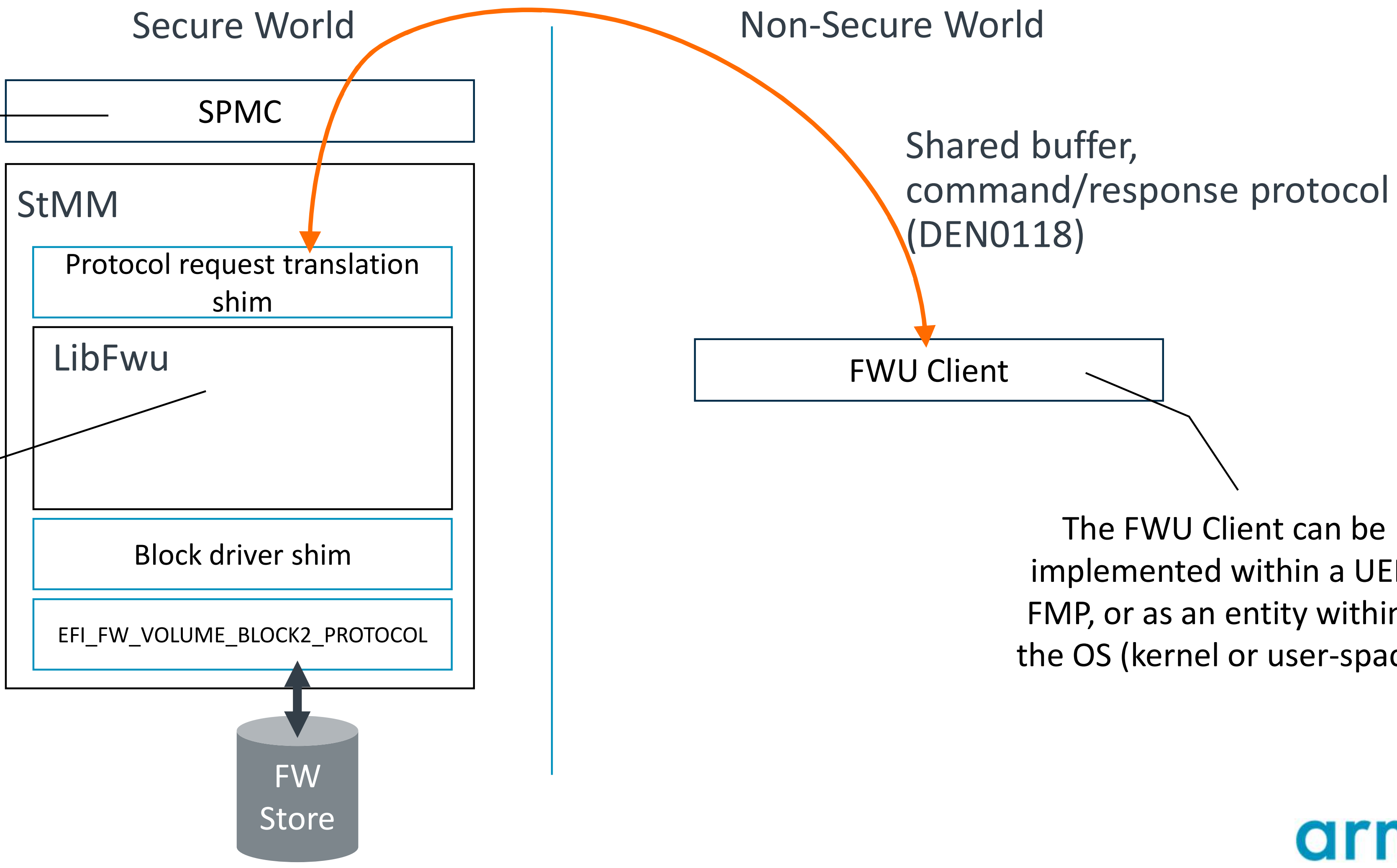
- Currently, three deployments of the Update Agent are maintained in the TS project:
  - **SP deployment** – Update Agent runs within a secure partition where updates are applied to a dedicated Swd flash device. This is the reference deployment that implements the Swd model described in the Arm FWU-A specification.
  - **PC app deployment** – Update Agent runs within a command-line app. Updates are applied to a disk image file, UEFI formatted with MBR/GPT. Useful for checking compatibility with the flash image file generated by the firmware build process.
  - **Libfwu** – a library that incorporates the core Update Agent components to simplify reuse by downstream projects.
- Different integrations of FWU components are also included in the following test deployments:
  - **component-test** – includes component and integration tests of FWU components. Runs in native PC environment.
  - **ts-service-test** – service-level tests. Uses a simulated device that mimics the role of the bootloader.

PC App Deployment

- Update Agent forms core of command-line app.
- Operates on a disk image file.
- Useful for test and debug.
  - Inspect image directory contents
  - Inspect FWU metadata
  - Apply updates to disk image
- Can also be used as a tool to initialise FWU metadata.
- Operates just like an embedded deployment.

```
Update agent started: boot index: 0 metadata ver: 2
fwu_metadata (size 267 bytes) :
               crc_32 : 0x255c880b
               version : 2
               metadata_size : 267
               header_size : 20
               active_index : 0
               previous_active_index : 0
               bank_state : 0xfc 0xff
               fw_store_desc :
                              num_banks : 2
                              num_images : 3
                              img_entry_size : 80
                              bank_entry_size : 24
                              img_entry[0] :
                                             img_type_uuid : 2451cd6e-90fe-4b15-bf10-a69bce2d4486
                                             location_uuid : 00000000-0000-0000-0000-000000000000
                                             img_bank_info[0] :
                                                            img_uuid : 2f597748-56e8-49c8-816f-9af411aecc91
                                                            accepted : 1
                                             img_bank_info[1] :
                                                            img_uuid : 31e0088c-ff47-4447-a49e-f12e5fdb3a17
                                                            accepted : 0
                              img_entry[1] :
                                             img_type_uuid : 691d5ea3-27fe-4104-badd-7539c00a9095
                                             location_uuid : 00000000-0000-0000-0000-000000000000
                                             img_bank_info[0] :
                                                            img_uuid : a08775c3-cdf5-407f-a284-327cbb0dfc4c
                                                            accepted : 1
                                             img_bank_info[1] :
                                                            img_uuid : 6eea03c4-cf29-41b7-b013-98588d48c6ee
                                                            accepted : 0
                              img_entry[2] :
                                             img_type_uuid : c948a156-58cb-4c38-b406-e60bff2223d5
                                             location_uuid : 00000000-0000-0000-0000-000000000000
                                             img_bank_info[0] :
                                                            img_uuid : 234f06c1-54e3-4831-9a95-d55a64dcc658
                                                            accepted : 1
                                             img_bank_info[1] :
                                                            img_uuid : 67ef4ed9-7436-4b32-99eb-9452e9466599
                                                            accepted : 0
OK
```

Possible StMM Deployment



Secure World
Non-Secure World
SPMC
StMM
Protocol request translation shim
LibFwu
Block driver shim
EFI_FW_VOLUME_BLOCK2_PROTOCOL
FW Store
FWU Client
Shared buffer, command/response protocol (DEN0118)

SPMC can be in TF-A (EL3) or S-EL2 (hafnium)

LibFwu source hosted at trustedfirmware.org
https://git.trustedfirmware.org/TS/trusted-services.git/

The FWU Client can be implemented within a UEFI FMP, or as an entity withing the OS (kernel or user-space)

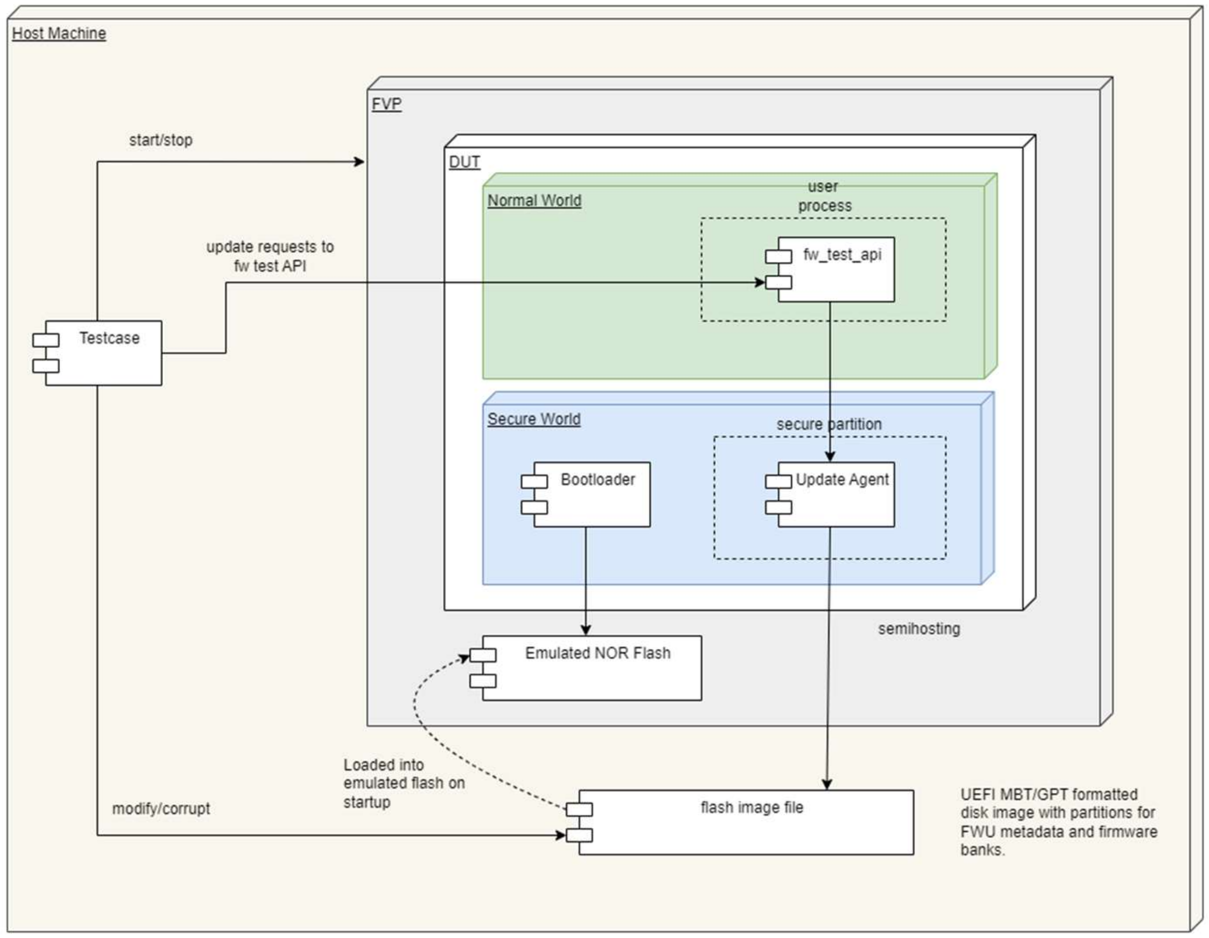
Testing the Update Agent

- Two main types of test:
  1. **Native PC tests** – uses a simulated device to mimic bootloader, device shutdown and reboot behaviour. Tests run within the CppUtest framework and exercise the DUT via the Update Agent public interface. Test cases apply different configurations to test multi-location and partial update behaviour. Tests organised into the following test suites:
     - Image directory tests
     - Normal update scenario tests
     - Invalid client behaviour tests
     - Power failure tests
     - Oversize image tests
     - Version rollback tests
  2. **Host controlled real device tests** – tests where the DUT is a physical or emulated device. Tests run on a host machine and communicate with a broker running on the DUT. The broker communicates with the Update Agent. *Development of this test infrastructure is in progress*.

```
TEST(FwuOversizeImageTests, oversizeInstallMultiLocationEndStaging) - 3 ms
TEST(FwuOversizeImageTests, oversizeInstallEndStaging) - 1 ms
TEST(FwuOversizeImageTests, oversizeInstallCancelStaging) - 1 ms
TEST(FwuOversizeImageTests, maxSizeInstall) - 0 ms
TEST(FwuRollbackTests, bootloaderFallback) - 1 ms
TEST(FwuRollbackTests, selectPreviousAfterActivation) - 1 ms
TEST(FwuRollbackTests, selectPreviousPriorToActivation) - 0 ms
TEST(FwuPowerFailureTests, powerFailureDuringTrial) - 4 ms
TEST(FwuPowerFailureTests, powerFailureDuringStaging) - 3 ms
TEST(FwuUpdateScenarioTests, partialFirmwareUpdateFlow) - 2 ms
TEST(FwuUpdateScenarioTests, wholeFirmwareUpdateFlow) - 1 ms
TEST(FwuInvalidBehaviourTests, invalidOperationsInTrial) - 2 ms
TEST(FwuInvalidBehaviourTests, invalidOperationsInStaging) - 1 ms
TEST(FwuInvalidBehaviourTests, invalidOperationsInRegular) - 0 ms
TEST(FwuImageDirectoryTests, zeroFwLocations) - 0 ms
TEST(FwuImageDirectoryTests, multipleFwLocations) - 1 ms
TEST(FwuImageDirectoryTests, singleFwLocation) - 0 ms
TEST(FwuImageDirectoryTests, streamRecycling) - 1 ms
TEST(FwuImageDirectoryTests, streamedReads) - 0 ms
TEST(FwuCopyInstallerTests, installAndCopy) - 2 ms
TEST(FwuRawInstallerTests, normalInstallFlow) - 0 ms
TEST(FwuMetadataV2Tests, checkImgBankInfoStructure) - 0 ms
TEST(FwuMetadataV2Tests, checkImgEntryStructure) - 0 ms
TEST(FwuMetadataV2Tests, checkFwStoreDescStructure) - 0 ms
TEST(FwuMetadataV2Tests, checkHeaderStructure) - 0 ms
TEST(FwuMetadataManagerTests, checkAndRepairInaccessibleStorage) - 0 ms
TEST(FwuMetadataManagerTests, checkAndRepairAccessibleStorage) - 2 ms

OK (245 tests, 27 ran, 332470 checks, 0 ignored, 218 filtered out, 27 ms)
```

Proposed Test Infrastructure



Code organisation in TS project

- The reference FWU code lives in the Trusted Service project: https://review.trustedfirmware.org/TS/trusted-services
- Some components from TF-A are also used (GPT parser, IO Dev)
- Source code is organised as follows:
  - Public interface message structures (for C/C++):
    - <TSROOT>/protocols/service/fwu/packed-c
  - FWU service components:
    - <TSROOT>/components/service/fwu
- FWU deployments:
  - Service provider in SP:
    - <TSROOT>/deployments/fwu/config/default-sp
  - Update agent command-line app:
    - <TSROOT>/deployments/fwu/config/fwu-app-linux-pc

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה