CAPSULE UPDATE & LVFS IMPROVING SYSTEM FIRMWARE UPDATES

Presented by Brian Richardson, Intel Corporation
Materials prepared by Vincent Zimmer (Intel), Mike Kinney (Intel) and Richard Hughes (LVFS Maintainer)
FOSDEM (Feb 2020)
Topics

The Update Problem
Using UEFI Capsules for Firmware Update
Firmware Management Protocol
Modularization
Distribution using LVFS
Summary & Call to Action
THE UPDATE PROBLEM
PROBLEM STATEMENT

- Low-attach rate for firmware updates on end-user systems
- Firmware process is traditionally designed for experts, not users
- Creates an environment where released updates are never applied
CHALLENGES FOR LINUX

• OEM update process typically targets users of Microsoft Windows

• Running an update utility at Linux runtime has technical complexities

• Creates an environment where released updates are never applied
Current Solution Space

- **Standardized Delivery Format**
  - OS-independent payload (Capsule)

- **Infrastructure for Update Delivery**
  - Consistent protocols and data formats
  - OS-based staging infrastructure

- **Leverage Modular Firmware Infrastructure**
  - Drive innovation through expandability & flexibility
USING UEFI CAPSULES FOR Firmware UPDATE
Using UEFI Capsules for Firmware Update
(Unified Extensible Firmware Interface)

1. Generate UEFI Capsule
2. Publish UEFI Capsule
3. Distribute UEFI Capsule
4. Process UEFI Capsule

Linux* Vendor Firmware Service (LVFS)
https://fwupd.org

* Other names and brands may be claimed as property of others
Process UEFI Capsule

ESRT = EFI System Resource Table
FMP = Firmware Management Protocol
GUID = Globally Unique Identifier
UEFI Capsule Processing using UEFI PI

- Power on
  - [.. Platform initialization ..]
- Normal Boot
  - ResetSystem()
- Coalesce UEFI Capsules
- Process UEFI Capsules
- Final OS Boot Loader
- Save UEFI Capsule
- Receive UEFI Capsules (LVFS) based on matching ESRT
- Load UEFI Capsule
- Call UpdateCapsule()
- Normal Boot
  - ResetSystem()
- Normal Boot
  - ResetSystem()
- Boot on FLASH Update
  - ResetSystem()
- Normal Boot

- Security (SEC)
- Pre EFI Initialization (PEI)
- Driver Execution Environment (DXE)
- Boot Dev Select (BDS)
- Transient System Load (TSL)
- Run Time (RT)
- After Life (AL)

- Run Time
  - [.. OS boot ..]
- Shutdown
Firmware Update Indicators

**UEFI Graphics Console**
EFI_GRAPHICS_OUTPUT_PROTOCOL

- System Logo
- User Experience (UX) Capsule
- Bitmap Message

**UEFI Text Console**
EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL

- Update Progress - 100%
- Update Progress - 100%
- Update Progress - 100%
- Update Progress - 32%

Customize with a new DisplayUpdateProgressLib instance
The Modular Philosophy

Make firmware component integration easy during Manufacturing.

Make firmware update easy using Capsules.

Networking Module

MnpDxe
IpDxe
UdpDxe
TcpDxe
HttpDxe

FvPreMemory
FVPREMEMORY CORE
FSP
FVUEFIBOOTCORE
FvUefiBoot
FVUEFISecure Boot
FV GOP
FvSecurity
FvAdvanced
NV_STORAGE

FV CORE
Silicon
OEM

ESRT

GUID

Capsule

MnpDxe
IpDxe
UdpDxe
TcpDxe
HttpDxe
Intel Open Platform – Minimum Platform + Intel® FSP

- UEFI is built with the PC supply chain in mind.
  - Open & closed modules co-exist in a system.
  - Minimum Platform increases overall share of open source UEFI firmware code available.
  - Increases open source firmware for community engagement, development & testing.
- UEFI component-based design gives OEMs choices:
  - Wide array of peripherals and components:
    - CPU, GPU, I/O Controllers (USB, Disk, etc.)
  - Silicon vendors can provide pluggable UEFI components that adhere to specifications.
FmpDxe Module Overview

FMP DXE Module
Configured through PCDs
Produces UEFI Firmware Management Protocol

FmpAuthenticationLib
BaseCryptLib
OpensslLib

FmpPayloadHeaderLib

CapsuleUpdatePolicyLib

FmpDeviceLib

Generic
Device Vendor
Platform Vendor

PCD = Platform Configuration Database
DISTRIBUTION USING LVFS
Distribution Using LVFS

Two Major Components

fwupd - Mechanism

- 100% free software (LGPLv2+)
- Used by users, typically with a GUI

lvfs-website – Data Source

- 100% free software (GPLv2+)
- Used by vendors: OEMs and ODMs
LVFS – “It’s Just a Website”

Designed for vendor secrecy (permissions system)
LVFS can be mirrored using PULP

Material contributed by rhughes@redhat.com
Vendor Support for LVFS
Layers of Security & UEFI Capsule Verification

- Firmware
  - GPG/PKCS7
- MetaInfo XML
- Cabinet Archive
- fwupd.org SSL
- AppStream XML
- GPG/PKCS7
- Windows INF
- Windows CAT
- README

UEFI Capsule

Check the UEFI capsule header and file structure

- GUID: 5ffdbca0d-f340-441c-a083-8439ecb0aee10
- HeaderSize: 0x1000
- Flags: 0x70000
- CapsuleImageSize: 0xabad

com.intel.UEFI.Application.InfineonTpmUpdateDxe

- Serial Number: 1137338005281104851497182458154224830145101854
- Description: C=US, ST=Washington, L=Redmond, O=Microsoft Corporation
- Not Before: 2016-11-17 22:05:37
- Not After: 2018-02-17 22:05:37
- Plugin: PE Check

Material contributed by rhughes@redhat.com
Firmware Analysis (LVFS Server Side)

Version 1.10.1:
- Uploaded: 2019-03-18 09:16:12
- State: stable
- Urgency: critical
- License: proprietary
- Filename: Signed_1152921504627948718.cab
- Description: This stable release fixes the following issues:
  - Fixed an issue with Secure Boot Option ROM Signature Verification.

Some new functionality has also been added:
- Added TPM PPI Bypass for Clear Command support.
- Added BIOS Password Feature: Master Password Lockout.

Security:
- Added to the LVFS by Dell
- Firmware has no attestation checksums
- Update is cryptographically signed
- Firmware can be verified after flashing
- Virus checked using ClamAV

Plugin: CHIPSEC
Size: 271.0KiB
Entropy: 5.76
GUID: 4953f72b-606d-41f5-956d-0ac77426bb68
SHA1: 6f27a53d07642b82464c49c9682319688516f38b1
SHA256: de64393ebff49bf461b5d8c325b14f0ad3dfc9aadd929bf11459e39843a25be6

Material contributed by rhughes@redhat.com
Way too much LVFS info for one presentation!

Looking to the Future

• Dashboard, albeit with caveats
• Get adoption from a few remaining vendors
• More tests, possibly using external companies

Per Richard... “Question Everything! (except asking what vendors are testing in secret!)”

• https://www.fwupd.org/
• https://github.com/fwupd/lvfs-website

Material contributed by rhughes@redhat.com
SUMMARY & CALL TO ACTION
Summary

EDK II supports UEFI Capsule Infrastructure for Firmware Update

- Simplifies FMP support for system firmware and integrated devices.
- Multiple authentication keys with flexible key storage options.
- System update pre-check (Power/battery, thermal, and system).
- Improved UX with progress indicators during update.
- Built-in support for test key detection & watchdog timer.
- Simplified ESRT driver using FMP instances

Open Source Developers can Generate Signed UEFI Capsules

Infrastructure Simplifies Distribution and Adoption of Firmware Updates
Call to Action

Platform Designers & OEMs
• Use Signed Capsules to distribute firmware updates
  • Guidance: NIST 800-147/800-147B
• Post updates to LVFS & Microsoft Windows Update
• Require device vendors create capsules for their components
• Platforms should implement a firmware recovery solution (NIST 800-193)

Developers
• Engage with open source communities supporting modern update solutions (examples: LVFS, EDK II) to ensure compatibility with future products
More Information

▪ Firmware threat model information:
  – https://edk2-docs.gitbooks.io/edk-ii-secure-coding-guide/content/appendix_threat_model_for_edk_ii/asset_flash_content.html
  – https://edk2-docs.gitbooks.io/understanding-the-uefi-secure-boot-chain/content/secure_boot_chain_in_uefi/boot_chain__putting_it_all_together/signed-capsule-update.html

▪ LVFS: https://fwupd.org/


▪ UEFI Specifications: https://uefi.org/specifications


▪ Additional Resources:
UEFI Platform Initialization (PI) Architecture

Firmware Phases

- **Pre EFI Initialization (PEI)**
  - CPU Init
  - Chipset Init
  - Board Init
  - Device, Bus, or Service Driver
  - EFI Driver Dispatcher
  - Intrinsic Services

- **Boot Environment (DXE)**
  - EFI Driver Dispatcher
  - Boot Manager
  - OS-Absent App
  - Transient OS Environment
  - Transient OS Boot Loader
  - Final OS Boot Loader
  - Final OS Environment

- **Final OS Environment**

- **Run Time (RT)**

- **After Life (AL)**

---

**Power on**

- Platform initialization

- OS boot

- Shutdown
## EDK II UEFI Capsule Features

EFI Development Kit II ([https://www.tianocore.org](https://www.tianocore.org))

<table>
<thead>
<tr>
<th>Feature</th>
<th>UDK2017 / UDK2018</th>
<th>edk2-stable201808</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate UEFI Capsule</td>
<td>Integrated EDK II Build</td>
<td>Standalone Python* Script</td>
</tr>
<tr>
<td>Update Granularity</td>
<td>Focused on Monolithic</td>
<td>Designed to support Multiple Components</td>
</tr>
<tr>
<td>Authentication</td>
<td>PKCS7 Single Key</td>
<td>PKCS7 Multiple Keys</td>
</tr>
<tr>
<td>Pre Check</td>
<td>N/A</td>
<td>Power/Battery, Thermal, System</td>
</tr>
<tr>
<td>Update Indicator</td>
<td>Requires platform code</td>
<td>Built-in with Consistent UX and Progress Bar</td>
</tr>
<tr>
<td>Firmware Management Protocol</td>
<td>Requires full implementation</td>
<td>Produced by FmpDxe module customized using configuration data and small libraries.</td>
</tr>
<tr>
<td>Test Key Detection</td>
<td>Requires platform code</td>
<td>Built-in</td>
</tr>
<tr>
<td>Watchdog</td>
<td>Requires platform code</td>
<td>Built-in</td>
</tr>
<tr>
<td>ESRT Driver</td>
<td>Legacy + FMP</td>
<td>Smaller/Simpler FMP only version</td>
</tr>
</tbody>
</table>

* Other names and brands may be claimed as property of others
ESRT GUIDs and Keys
Multiple Components

System Firmware

FMP Driver
ImageTypeId
GUID A

FMP Driver
ImageTypeId
GUID B

FMP Driver
ImageTypeId
GUID C

FMP Driver
ImageTypeId
GUID D

Vital Product Data (VPD)
Public Key(s)

Public Key(s)

Public Key(s)

Public Key(s)

ESRT Table
GUID A
GUID B
GUID C
GUID D
ESRT GUIDs and Keys

3rd Party FMP Driver

3rd Party UEFI Capsules must be resigned with System Key
ESRT GUIDs and Keys

3\textsuperscript{rd} Party FMP Driver

System allows UEFI Capsules from 3rd Party to be installed
Add FMP to Existing Device Driver

System Firmware

Device Driver
- FMP Library
  - ImageTypeId GUID A
  - Public Key(s)

FMP Driver
- ImageTypeId GUID B
  - Public Key(s)

FMP Driver
- ImageTypeId GUID C
  - Public Key(s)

ESRT Table
- GUID A
- GUID B
- GUID C
## FmpDxe Module Configuration

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE_GUID</td>
<td>ESRT GUID Value</td>
</tr>
<tr>
<td>PcdFmpDeviceImageIdName</td>
<td>FMP Image Descriptor - Unicode string</td>
</tr>
<tr>
<td>PcdFmpDeviceBuildTimeLowestSupportedVersion</td>
<td>Build time FMP/ESRT default value</td>
</tr>
<tr>
<td>PcdFmpDeviceLockEventGuid</td>
<td>Event GUID to lock FW storage device. Default is End of DXE.</td>
</tr>
<tr>
<td>PcdFmpDeviceProgressWatchdogTimeInSeconds</td>
<td>Watchdog armed on each progress update</td>
</tr>
<tr>
<td>PcdFmpDeviceProgressColor</td>
<td>24-bit Progress Bar Color (0x00rrggbb)</td>
</tr>
<tr>
<td>PcdFmpDevicePkcs7CertBufferXdr</td>
<td>One or more PKCS7 Certs in XDR format. Encode with <code>BaseTools/Scripts/BinToPcd</code></td>
</tr>
<tr>
<td>PcdFmpDeviceTestKeySha256Digest</td>
<td>Set to <code>{0}</code> to disable test key detection</td>
</tr>
</tbody>
</table>

XDR = External Data Representation using Variable-Length Opaque Data format from RFC 4506
# CapsuleUpdatePolicyLib APIs

## Platform Specific Library

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CheckSystemPower()</td>
<td>Is system power/battery ok for FW update?</td>
</tr>
<tr>
<td>CheckSystemThermal()</td>
<td>Is system temperature ok for FW update?</td>
</tr>
<tr>
<td>CheckSystemEnvironment()</td>
<td>Is the system environment ok for FW update?</td>
</tr>
<tr>
<td>IsLowestSupportedVersionCheckRequired()</td>
<td>Skip lowest supported version check? (e.g. Service Mode)</td>
</tr>
<tr>
<td>IsLockFmpDeviceAtLockEventGuidRequired()</td>
<td>Skip firmware storage device lock action? (e.g. Manufacturing Mode)</td>
</tr>
</tbody>
</table>
## FmpDeviceLib APIs - Device Specific Library

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegisterFmpInstaller()</td>
<td>Future expansion for add-in controllers.</td>
</tr>
<tr>
<td>FmpDeviceGetSize()</td>
<td>Size of <em>currently stored FW image</em>.</td>
</tr>
<tr>
<td>FmpDeviceGetImageTypeIdGuidPtr()</td>
<td>ESRT/FMP GUID. Overrides FILE_GUID value.</td>
</tr>
<tr>
<td>FmpDeviceGetAttributes()</td>
<td>FMP Attributes Supported/Settings.</td>
</tr>
<tr>
<td>FmpDeviceGetLowestSupportedVersion()</td>
<td>LSV from <em>currently stored FW image</em>.</td>
</tr>
<tr>
<td>FmpDeviceGetVersionString()</td>
<td>Unicode version string from <em>currently stored FW image</em>.</td>
</tr>
<tr>
<td>FmpDeviceGetVersion()</td>
<td>32-bit version value from <em>currently stored FW image</em>.</td>
</tr>
<tr>
<td>FmpDeviceGetImage()</td>
<td>Retrieve copy of <em>currently stored FW image</em>.</td>
</tr>
<tr>
<td>FmpDeviceCheckImage()</td>
<td>Check if a new FW image is valid for this device.</td>
</tr>
<tr>
<td>FmpDeviceSetImage()</td>
<td>Update FW storage with a new FW image.</td>
</tr>
<tr>
<td>FmpDeviceLock()</td>
<td>Lock FW storage to prevent any further changes.</td>
</tr>
</tbody>
</table>
Legal Disclaimer

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request. No product or component can be absolutely secure.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel, the Intel logo, Intel Firmware Support Package (Intel FSP), Intel Server Platform Services (Intel SPS), Intel Slim Bootloader, and Intel Trusted Execution Technology (Intel TXT) are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others

© 2020 Intel Corporation.