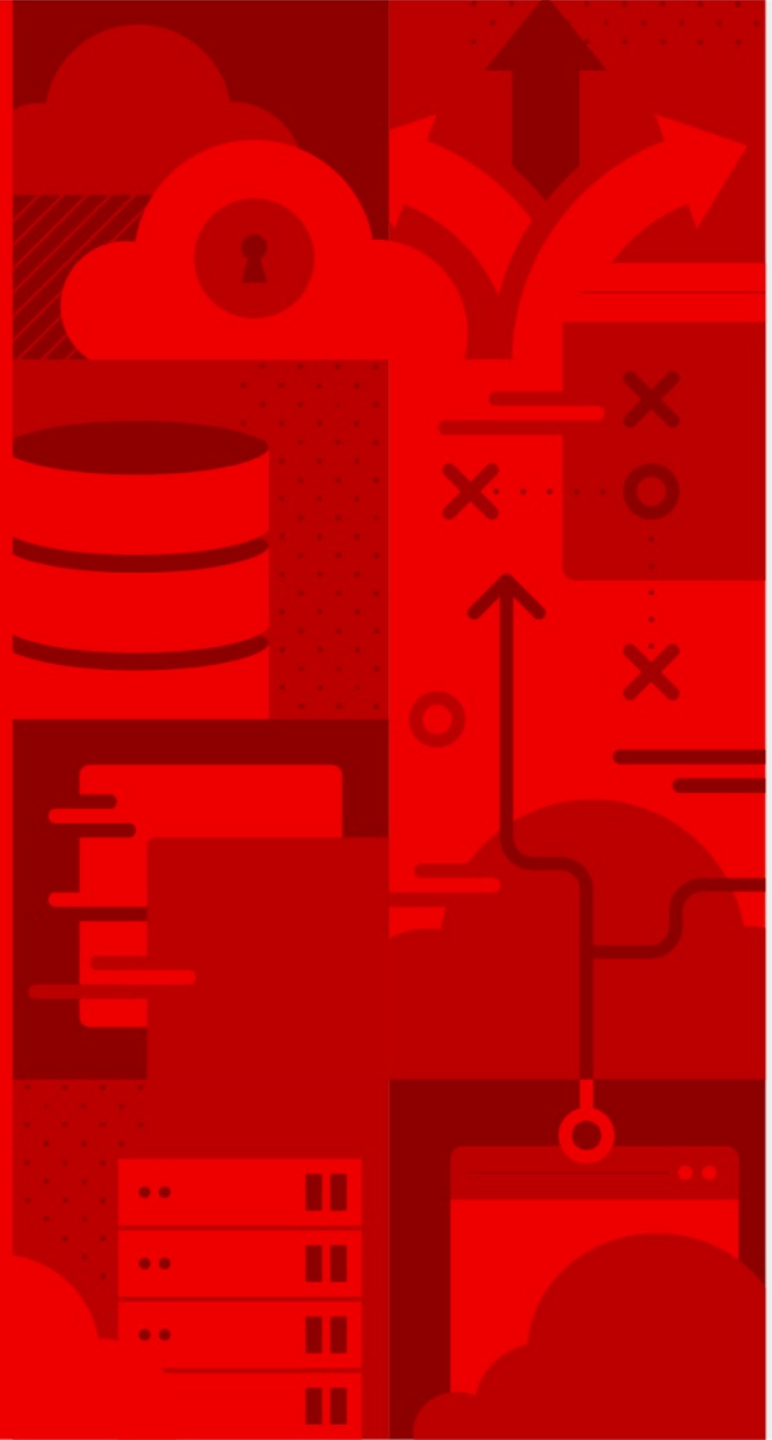


LVFS and fwupd

A high level overview for UNC-FaMAF explaining we distribute firmware updates in Linux.

Richard Hughes
Principal Engineer

Red Hat

Who am I?

I've been doing Open Source work for over 15 years.

I'm responsible for at least 3 of the projects currently installed on your Linux machine.

You already trust me!

Tens of millions of people use my software every single day.

This "talking at University" thing is new to me; please be kind :)

Red Hat

The Problem: Users were not updating firmware

What hardware is installed?

Users don't typically know exactly what hardware they are using.

What updates are available

Users do not visit OEM websites to manually look for firmware updates.

Where do I get them from?

Many OEMs have insecure download links without any file checksums or signatures.

How to apply the update

Vendor tools often required Microsoft Windows, or unsupported Linux versions.

LVFS and fwupd work together

LVFS : Trusted Metadata Source

The hardware vendor uploads firmware to the LVFS where it is verified and signed. Users then download a shared metadata catalogue from a central server.

fwupd : Mechanism

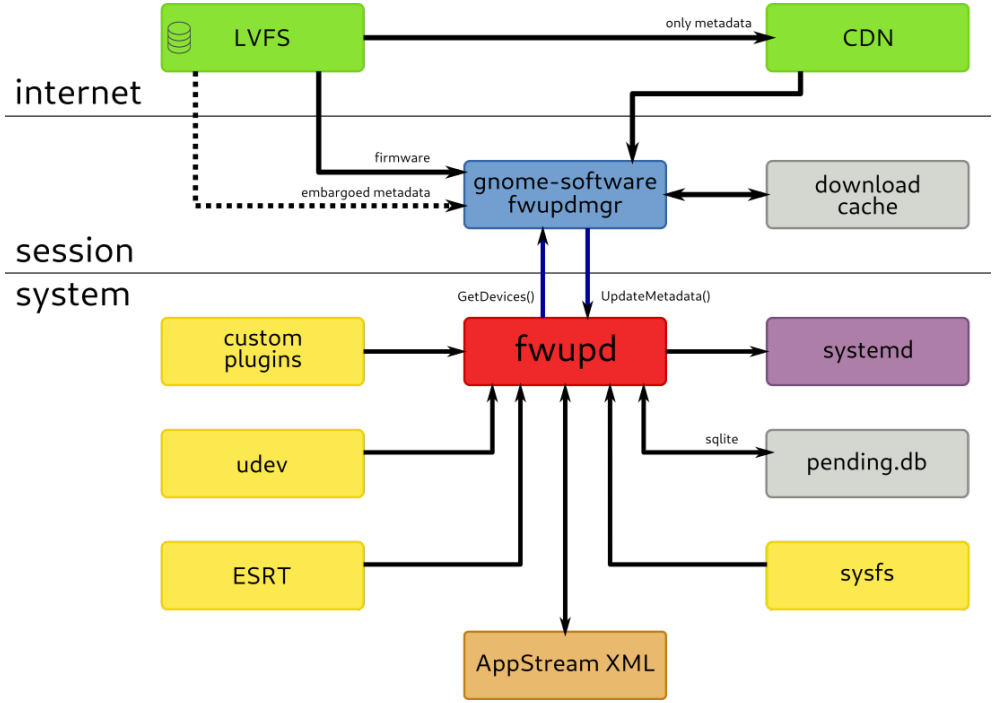
The open source fwupd project deploys the update onto the Linux client machine. Over 32 update protocols are now supported and more are planned.

LVFS : Anonymous Reporting

After updating firmware, fwupd optionally sends success or failure information back to the LVFS to ensure updates are being deployed without problems

Red Hat

Architecture of fwupd



D-Bus is used to interact with fwupd

- Desktop neutral interface with binding for every language
- Optionally downloads metadata from the LVFS
- Enumerate hardware & deploy firmware.

Updates not applied without an agent

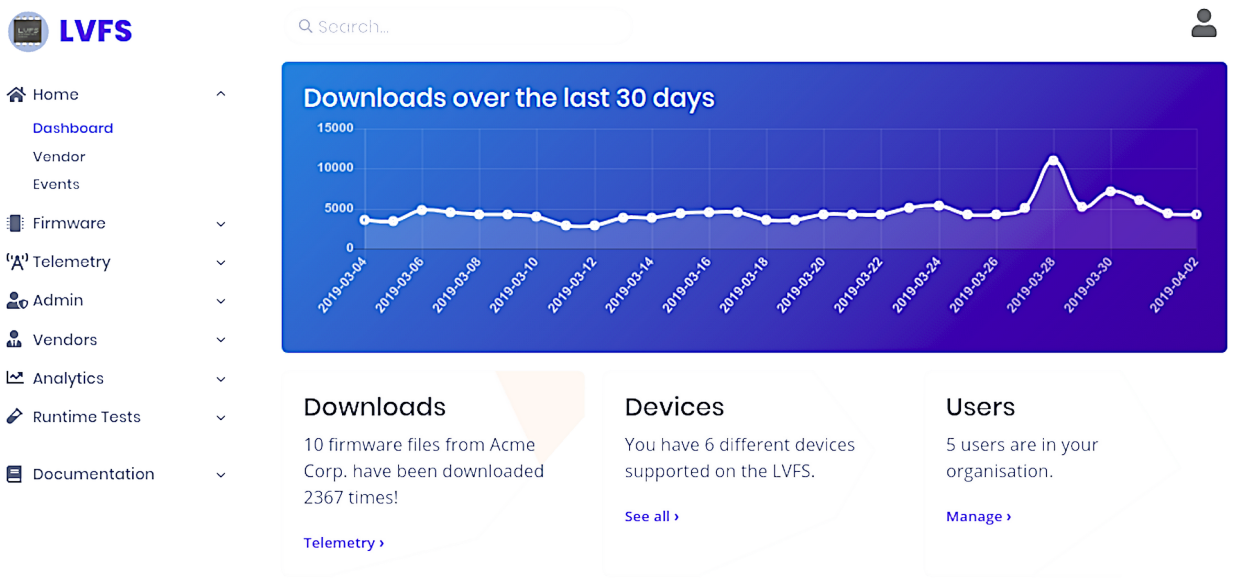
- Full integration with GNOME and KDE, and CLI interface

Scalable architecture designed to continue to grow

- Written in a *lowest common denominator* language: **C**
- Well tested dependancies of **GLib** and **GObject**

Red Hat

Architecture of LVFS



A simple web service that had to be "just good enough"

- Adding functionality only when requried

Privacy-centric by design

- Puts privacy first by matching hardware client side
- Metadata scale out to users via a "dumb" CDN

Mostly centralized firmware distribution

- Can easily be mirrored on a private network
- For demoting failing firmware
- *Really* for statistics

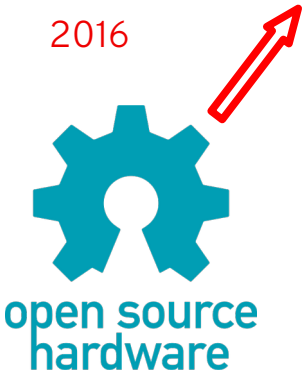
Scalable architecture

- Written in a high level language: **Python**
- Well tested dependancies of **Flask** and **SQLAlchemy**

Red Hat

90 OEMs, ODMs & IHVs **all work together**

2016 — 2017 — 2018 — 2019 — 2020 — 2021



Most of the consumer supply chain

Red Hat

Firmware Analysis : Raising the Bar

Blocklist

Use a simple blocklist to check firmware for problems

Enabled

Values

```
DO NOT TRUST::IBV example certificate being used
DO NOT SHIP::IBV example certificate being used
To Be Defined By O.E.M::IBV example DMI data being used
c97445f45cdef9f0d3e05e1e585fc297235b82b5be8ff3efca67c59852018192::Contains the Dual EC backdoor for the NSA
Do not trust::IBV example certificate being used
```

Modify

Red Hat

Firmware Analysis : Certificates

com.intel.Uefi.Application.InfineonTpmUpdateDxe

Serial Number	113733800528110485149718245815422483014510185 4
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

Red Hat

Firmware update cadence used for purchasing

Red Hat

Private End-to-End Testing

Private

Firmware is only available to your specific user.

Move here

Embargo

Firmware is available to anyone in your vendor group.

Move here

Testing

Firmware is available to thousands of public testers.

Move here

Stable

Firmware is available to millions of public end-users.

History

Timestamp	User	Target
2016-12-28 10:34:11	richard@hughsie.com	stable
2016-12-28 10:34:07	richard@hughsie.com	testing
2016-12-28 10:34:03	richard@hughsie.com	embargo-hughski
2016-12-28 10:34:00	richard@hughsie.com	private

Red Hat

There is no cost to use the LVFS or to contribute to fwupd

The Linux Vendor Firmware Service is managed by the Linux Foundation and core development work is provided by **neutral** Red Hat.

Independent consulting companies provide technical help and training.

Red Hat

OEMs just want an easy life

OEMs are free to choose whatever criteria they like for hardware suppliers, and they are choosing these rules for various business reasons.

Lenovo

All suppliers for Lenovo ThinkPad, ThinkStation and ThinkCentre have to have working fwupd plugins and be able to upload to the LVFS. Failure to meet either criteria causes the “preferred vendor” status to be lost.

Dell

All approved ODMs and ISVs being used by Dell must have firmware that can be updated using fwupd and have updates available on the LVFS.

Google

Firmware must be updatable using fwupd to get the “Works with Chrome” compliance sticker. Google are shipping parts of fwupd in every Chromebook now sold.

ODMs and OEMs include LVFS in contracts

Independent BIOS Vendor

The OBV typically uploads firmware to the LVFS to run tests and to verify that the image works with fwupd. IBVs and ISVs are normally not shown on the LVFS.

Original Device Manufacturer

The ODM can either just upload updates on behalf of the OEM, or the ODM can manage the entire QA process including pushing to testing and stable.

Original Equipment Manufacturer

The OEM is the “user visible” brand the user is familiar with, and is typically the only vendor visible on the LVFS. OEMs can test firmware uploaded by their ODMs.

Keeping two world in sync

Eclypsium

Vendor is sharing metadata with the LVFS.

Show devices

- Lenovo ThinkPad T560/P50s System Update (0.1.29 != 1.31)
- Lenovo ThinkPad X1 Carbon 5th System Update (0.1.48)
- Lenovo ThinkPad X1 Carbon 5th Embedded Controller Update (0.1.20)
- Lenovo ThinkPad X1 Yoga 2nd System Update (0.1.38)
- Lenovo ThinkPad X1 Yoga 2rd Embedded Controller Update (0.1.17)
- Lenovo ThinkPad T470 / ThinkPad 25 System Update (0.1.64)
- Lenovo ThinkPad P71 System Update (0.1.31)
- Lenovo ThinkPad P51 System Update (0.1.52)
- Lenovo ThinkPad T570/P51s System Update (0.1.42)

Red Hat

Every day over 10 million Linux users automatically download firmware update metadata from the LVFS.

Red Hat

The LVFS grows every year, as new vendors join and as more firmware is uploaded

Companies and agencies are free to mirror the LVFS for privacy or scalability reasons and so we don't actually know the real number of downloads.

Every day over 12 million Linux users automatically download firmware update metadata from the LVFS.

29.8M

Firmware files supplied to end users

Since the LVFS started the official server has supplied millions of firmware updates for over 200 different devices.

1.1M

Success reports from end users

Over 99% of firmware was deployed correctly, with 1% of "known failures" identified using a built-in rule engine.

Red Hat

What the vendors are saying...

“

LVFS is strategically important for Dell to be able to provide secure firmware updates in a standards-compliant way.

”

Mario Limonciello
Sr. Principal Software Engineer, Dell

“

Standardizing on LVFS has helped Lenovo seamlessly distribute our firmware updates to our customers

”

Rob Herman
Executive Director, Lenovo

Red Hat

Time for a 2 minute break?

After the break I'm going to show you how to create a real-world plugin. This is what programmers paid by billion-dollar OEMs and ODMs all over the world are doing right now.

These slides are also available here: `https://people.freedesktop.org/~hughsient/temp/UNC-FaMAF.pdf`

Plugin tutorial

Following along on your own computer is optional, but if you want to copy you will need:

- An Ubuntu or Fedora Linux installation, e.g. bare metal or in a VMWare/VirtualBox with git and an editor like gedit installed
- Internet access on the host for source code and additional packages
- About 500MB of spare storage space
- Some patience! Please remember I'm human too :)

Plugin tutorial

For this tutorial we will create a simple fwupd plugin that:

1. Builds a new source file into a shared plugin object
2. Initializes the plugin
3. Enumerates and creates a fake device
4. Accepts some firmware for the fake device (maybe, if we have time)

If something doesn't work or you fall behind DO NOT PANIC. I'll provide some "fast-forward" instructions. A link to the slides will also be available after the session if you want to try this in your own time.

Plugin tutorial : Getting the code

Let's get the fwupd code:

```
$ cd ~
$ git clone https://github.com/fwupd/fwupd.git
$ cd fwupd
$ git checkout 1.6.1
```

```
[hughsie@hughsie-work ~]$ git clone https://github.com/fwupd/fwupd.git
Cloning into 'fwupd'...
remote: Enumerating objects: 50882, done.
remote: Counting objects: 100% (1322/1322), done.
remote: Compressing objects: 100% (653/653), done.
remote: Total 50882 (delta 718), reused 1117 (delta 665), pack-reused 49560
Receiving objects: 100% (50882/50882), 23.15 MiB | 25.51 MiB/s, done.
Resolving deltas: 100% (39119/39119), done.
```

Plugin tutorial : Getting the deps

Let's get the build packages fwupd needs to compile:

```
$ OS=fedora ./contrib/ci/generate_dependencies.py |
xargs sudo dnf install -y
$ OS=ubuntu ./contrib/ci/generate_dependencies.py |
xargs sudo apt install -y
```

```
Dependencies resolved.
================================
 Package
================================
Installing:
 dbus-x11
 gnome-desktop-testing
 libgcrypt-devel
 mingw-w64-tools
 python3-markdown
 python3-typogrify
Installing dependencies:
 python3-smartypants
```

Plugin tutorial : Setting compile options

Let's set up some options which control how fwupd is built:

```
$ mkdir build && cd build
$ meson ../ -Dsystemd_root_prefix=/tmp -Dudevdir=/tmp
--prefix=$HOME/.root -Ddocs=none
```

```
Version: 0.56.2
Source dir: /home/hughsie/fwupd
Build dir: /home/hughsie/fwupd/build
Build type: native build
Project name: fwupd
Project version: 1.6.2
C compiler for the host machine: ccache cc (gcc 11.1.1 "cc (GCC) 11.1.1 20210531 (Red Hat 11.1.1-3)")
C linker for the host machine: cc ld.bfd 2.35.1-41
Host machine cpu family: x86_64
Host machine cpu: x86_64
Program git found: YES (/usr/bin/git)
Compiler for C supports arguments -Waggregate-return: YES
```

Plugin tutorial : Compiling the code

```
$ ninja -v

$ ninja install
```

Installing is very important as we'll find out later!

```
[78/83] ccache cc -Iplugins/wacom-usb/wacom-usb-self-test.p -Iplugins/wacom-usb -I../plugins/wacom-usb
de/glib-2.0 -I/usr/lib64/glib-2.0/include -I/usr/include/sysprof-4 -I/usr/include/libmount -I/usr/inclu
sb-1 -I/usr/include/libusb-1.0 -fdiagnostics-color=always -pipe -D_FILE_OFFSET_BITS=64 -Wall -Winvalid-
ter-statement -Wdiscarded-qualifiers -Wduplicated-branches -Wduplicated-cond -Wempty-body -Wformat=2 -W
 -Winit-self -Wlogical-op -Wmaybe-uninitialized -Wmissing-declarations -Wmissing-format-attribute -Wmis
t-function-type -Wno-address-of-packed-member -Wno-unknown-pragmas -Wno-missing-field-initializers -Wno
-Wpointer-arith -Wredundant-decls -Wreturn-type -Wshadow -Wsign-compare -Wstrict-aliasing -Wstrict-prot
vla -Wwrite-strings -fstack-protector-strong -DFWUPD_COMPILATION -D_DEFAULT_SOURCE -DFWUPD_DISABLE_DEPR
omUsb"' -MD -MQ plugins/wacom-usb/wacom-usb-self-test.p/fu-wac-firmware.c.o -MF plugins/wacom-usb/wacom
plugins/wacom-usb/fu-wac-firmware.c
```

Plugin tutorial : Creating a new plugin

Let's create a new source file that will be our simple **`famaf`** plugin:

```
$ mkdir ../plugins/famaf
$ gedit ../plugins/famaf/fu-plugin-famaf.c
```

Documents | Open | fu-plugin-famaf.c ~/Code/fwupd/plugins/famaf | Save

fu-plugin-famaf.c

```c
#include "config.h"

#include <fwupdplugin.h>

void
fu_plugin_init (FuPlugin *plugin)
{
        fu_plugin_set_build_hash (plugin, FU_BUILD_HASH);
        g_debug ("init");
}

```

Plugin tutorial : Building the new plugin

Let’s create a build definition that actually builds our new source file.

```
$ gedit ../plugins/famaf/meson.build
```

Documents | Open | meson.build ~/Code/fwupd/plugins/famaf | Save

fu-plugin-famaf.c | fu-plugin-famaf.c | meson.build

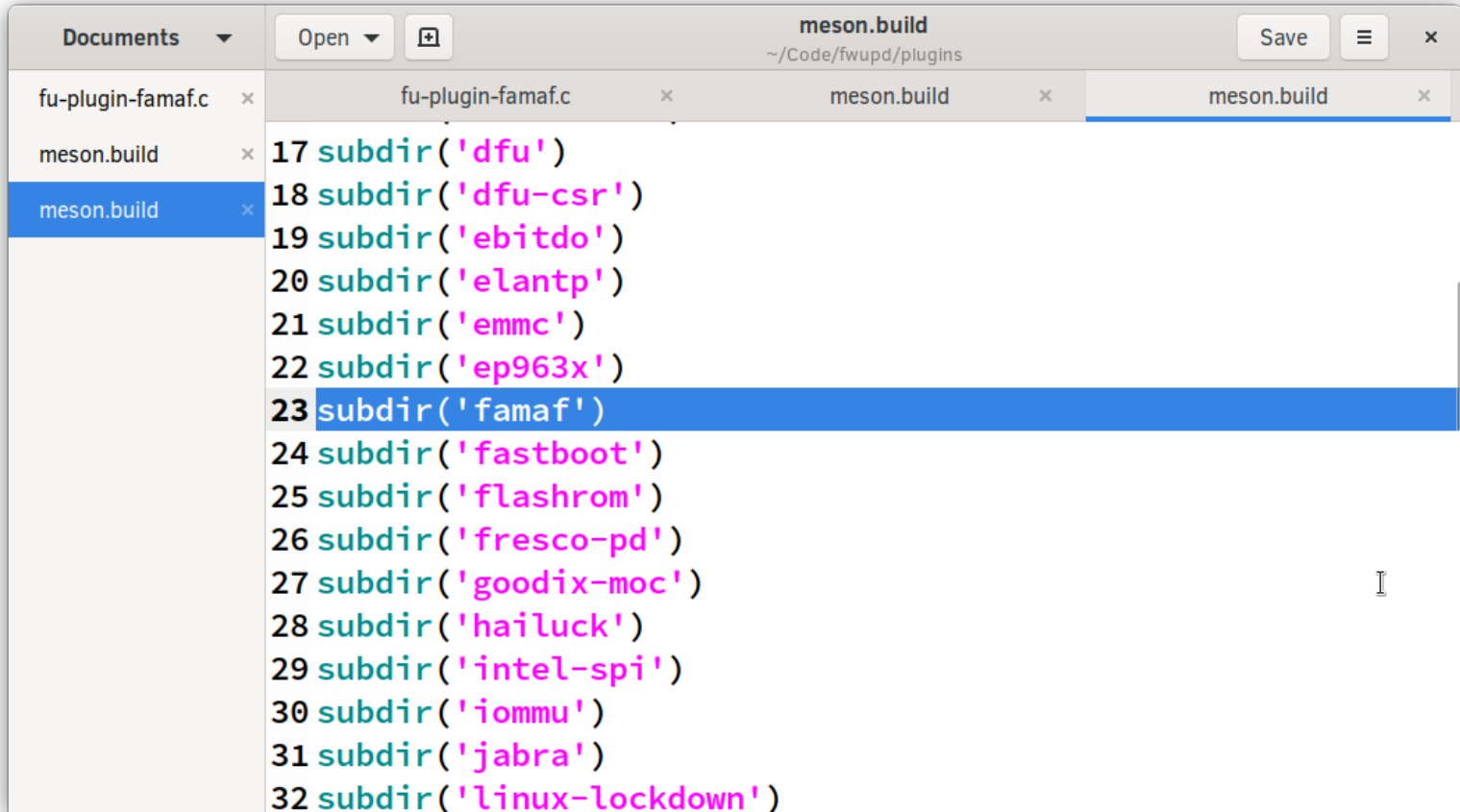
meson.build

```
shared_module('fu_plugin_famaf',
  fu_hash,
  sources : [
    'fu-plugin-famaf.c',
  ],
  include_directories : [
    root_incdir,
    fwupd_incdir,
    fwupdplugin_incdir,
  ],
  install : true,
  install_dir: plugin_dir,
  dependencies : [
    plugin_deps,
  ],
  link_with : [
    fwupd,
    fwupdplugin,
  ],
)
```

Plugin tutorial : Building the new plugin (2)

Now we have to tell the build system we have to use **`plugins/famaf`**

`$ gedit ../plugins/meson.build`



Plugin tutorial : Building the new plugin (3)

Now we can rebuilt the project and install **`libfu_plugin_famaf.so`**

```
$ ninja install
```

If you can't type as fast as I can speak, simply do:

```
$ git reset --hard
$ git checkout wip/famaf/init
```

```
Installing plugins/elantp/elantp-self-test to /home/hughsie/.root/libexec/installed-tests/fwupd
Installing plugins/emmc/libfu_plugin_emmc.so to /home/hughsie/.root/lib64/fwupd-plugins-3
Installing plugins/ep963x/libfu_plugin_ep963x.so to /home/hughsie/.root/lib64/fwupd-plugins-3
Installing plugins/famaf/libfu_plugin_famaf.so to /home/hughsie/.root/lib64/fwupd-plugins-3
Installing plugins/fastboot/libfu_plugin_fastboot.so to /home/hughsie/.root/lib64/fwupd-plugins-3
Installing plugins/fresco-pd/libfu_plugin_fresco_pd.so to /home/hughsie/.root/lib64/fwupd-plugins-3
Installing plugins/goodix-moc/libfu_plugin_goodixmoc.so to /home/hughsie/.root/lib64/fwupd-plugins-3
```

Plugin tutorial : Running the new plugin (3)

The main fwupd binary loads all the plugins and runs the system service. We just want to use a debug binary to run **just** our plugin:

```
$ sudo ./src/fwupdtool --plugins famaf --verbose get-devices
```

Wow!

```
10:46:57:0244 FuHwids          smbios property BaseboardManufacturer=LENOVO
10:46:57:0244 FuHwids          smbios property BaseboardProduct=20EQS64N0C
10:46:57:0244 XbSilo           attempting to load /home/hughsie/.root/var/cache/fwupd/metadata.xmlb
10:46:57:0244 XbSilo           file: 82b83b47-fa92-05e0-a1fb-742ecba6ae09, current:4a8e912b-5dab-336f-c2a9-3f69c4174c97, cached: (null)
10:46:57:0460 FuEngine         581 components now in silo
10:46:57:0468 FuPlugin         init(/home/hughsie/.root/lib64/fwupd-plugins-3/libfu_plugin_famaf.so)
10:46:57:0468 FIXME            init
10:46:57:0468 FuEngine         plugins disabled: synaptics_rmi, linux_tainted, uefi_dbx, jabra, nvme, dell_dock, upower, pci_bcr, dfu,
```

Plugin tutorial : Adding a device

During the “coldplug” phase plugins add devices already connected.

Fast forward with:

```
$ git reset --hard
$ git checkout
wip/famaf/coldplug
```

Documents | Open | fu-plugin-famaf.c ~/Code/fwupd/plugins/famaf | Save

fu-plugin-famaf.c | meson.build

```c

gboolean
fu_plugin_coldplug (FuPlugin *plugin, GError **error)
{
        g_autoptr(FuDevice) dev = fu_device_new ();
        g_debug ("coldplug");
        fu_device_set_name (dev, "Hello World");
        fu_device_set_physical_id (dev, "/dev/usb/foobarbaz");
        fu_device_add_vendor_id (dev, "USB:1234");
        fu_device_add_protocol (dev, "org.uefi.capsule");
        fu_device_add_flag (dev, FWUPD_DEVICE_FLAG_UPDATABLE);
        fu_device_add_instance_id (dev, "USB:VID=1234,PID=4567");
        if (!fu_device_setup (dev, error))
                return FALSE;
        fu_plugin_device_add (plugin, dev);
        return TRUE;
}
```

Plugin tutorial : Adding a device (2)

```
$ ninja install
$ sudo ./src/fwupdtool --plugins famaf --verbose get-devices
```

```
Loading…                    [-                                        ]10:56:39:0387 FuPlugin             coldplug(famaf)
10:56:39:0387 FIXME                 coldplug
10:56:39:0387 FuDevice              using 460b4e8ee60022203328ffaf31d15ea7a2a739ac for /dev/usb/foobarbaz
10:56:39:0387 FuPlugin              emit added from famaf: 460b4e8ee60022203328ffaf31d15ea7a2a739ac
10:56:39:0388 FuDeviceList          ::added 460b4e8ee60022203328ffaf31d15ea7a2a739ac
10:56:39:0388 FuMain                ADDED:
FuDevice:
Hello World
  DeviceId:             460b4e8ee60022203328ffaf31d15ea7a2a739ac
  Guid:                 8ab39228-f746-5835-988e-da8d0803adb9 ← USB:VID=1234,PID=4567
  Plugin:               famaf
  Protocol:             org.uefi.capsule
  Flags:                updatable|registered
  VendorId:             USB:1234
  Created:              2021-06-22
  PhysicalId:           /dev/usb/foobarbaz

10:56:39:0388 FuEngine              using plugins: famaf
```

Plugin tutorial : Writing Firmware

The vfunc **`fu_plugin_update()`** is called with the firmware payload.

Fast forward with:

```
$ git reset --hard
$ git checkout
wip/famaf/update
```

*fu-plugin-famaf.c meson.build

```c
gboolean
fu_plugin_update (FuPlugin *plugin,
                  FuDevice *device,
                  GBytes *blob_fw,
                  FwupdInstallFlags flags,
                  GError **error)
{
        g_debug ("update %s", (char*) g_bytes_get_data (blob_fw, NULL));
        g_set_error (error,
                     G_IO_ERROR,
                     G_IO_ERROR_NOT_SUPPORTED,
                     "this is 1h tutorial!");
        return FALSE;
}
```


Red Hat

Plugin tutorial : Writing Firmware (2)

```
$ ninja install
$ echo -n "LGTM" > firmware.bin
$ sudo ./src/fwupdtool --plugins famaf --verbose
install-blob firmware.bin
```

```
13:09:10:0964 FuPlugin            running superclassed update_detach(famaf)
13:09:10:0965 FuEngine            update -> FuDevice:
Hello World
  DeviceId:             460b4e8ee60022203328ffaf31d15ea7a2a739ac
  Guid:                 8ab39228-f746-5835-988e-da8d0803adb9 ← USB:VID=1234,PID=4567
  Plugin:               famaf
  Protocol:             org.uefi.capsule
  Flags:                updatable|registered
  VendorId:             USB:1234
  Created:              2021-06-23
  Modified:             2021-06-23
  PhysicalId:           /dev/usb/foobarbaz
  Order:                0

13:09:10:0965 FIXME               update LGTM
13:09:10:0965 FuPlugin            running superclassed update_attach(famaf)
13:09:10:0965 FuEngine            cleanup -> FuDevice:
```

Plugin tutorial : Submit upstream

If we added more details, the plugin we just wrote could be submitted upstream as a pull request.

Support for Realtek RTD2142 #3410

Open **tari** wants to merge 4 commits into `fwupd:master` from `tari:realtek-mst`

Conversation 6 | Commits 4 | Checks 10 | Files changed 11

tari commented 9 hours ago • edited — Contributor

Type of pull request:

- [x] New plugin (Please include new plugin checklist)
- [ ] Code fix
- [ ] Feature
- [ ] Documentation

New plugin checklist:

- [x] Fill out `README.md` with update protocol
- [x] Fill out `README.md` with any custom quirks and flags
- [x] Fill out `README.md` with the vendor ID security value
- [x] Implement `FuFirmware->write()` and include at least one fuzzer testcase in `src/fuzzing/firmware` for any custom `FuFirmware` subclass

Plugin tutorial : Complete!

Well done if you're still awake and following along!

There are lots of other things to implement, e.g.

- `prepare()`
- `detach()`
- `attach()`
- `cleanup()`

But this is for another day!

Thanks for listening!

Contact me:

richard@hughsie.com

rhughes@redhat.com

@hughsient

Red Hat