OpenPrinting®

cups-filters, CUPS Snap, Printer Applications, Driverless Scanning, ...

Till Kamppeter – OpenPrinting

18 May 2022

cups-filters: Introduction

- **cups-filters takes up everything from CUPS which Mac OS X does not need** (CUPS 1.6.x)
  - Started end of 2011 by OpenPrinting, **overtaking most of CUPS' filters**
  - Switched filters over from PostScript-centric to **PDF-centric workflow**
  - **cups-browsed** introduced end of 2012, to introduce **browsing of DNS-SD-advertised remote CUPS queues**, as CUPS dropped its own broadcasting/browsing
  - **11** years of **further development** added things like driverless printing support, clustering, support for Printer Applications, IPP standards, PPD-less...

cups-filters Development: libcupsfilters

- **General**
  - Converted **all CUPS filters** into **filter functions**
  - Filter functions work **without PPDs** (use IPP attrib.)
  - Use **parameters** instead of environment variables
  - All logging into **log function**, no leaks into stderr
- **New filter functions**
  - **`CfFilterPWGToRaster()`**
    - **Apple**/**PWG** Raster → **CUPS**/Apple/PWG Raster
    - Completely **streaming**
    - For **Printer Applications** to **stream** into "**`rasterto…`**" CUPS filters

cups-filters Development: libcupsfilters

- **New filter functions**
  - **`cfFilterUniversal()`**
    - Filter to **convert any format to any other**
    - Internally calls **chains of filter functions**
    - CUPS needs only to call **one filter executable**
  - **`cfFilterExternalCUPS()`**
    - **Calls classic CUPS filters/backends**
    - Call **drivers (also proprietary)** from Printer Apps
    - **Emulates complete CUPS environment**, including env variables, back/side channel, ...
    - Call backends also in **discovery mode**
    - Extensively used by **pappl-retrofit**

cups-filters Development: libcupsfilters

- **Improvements and Fixes**
  - **Auto-selection of color space and depth**
    - Filter gets `print-color-mode` and `print-quality`
    - and **Apple/PWG-Raster/PCLm** printer IPP attrs
    - → Filter **determines needed color space/depth**
    - **But:** PDF jobs do not provide color space info
  - **All driverless formats by Ghostscript**
    - **Added Apple Raster output** to GS 9.56.0
    - Feature requests for **streaming PCLm/raster PDF** and also **gray PCLm** got accepted for GS 9.56.0
    - `cfFilterGhostsript()` outputs **all formats** now
    - → Simplifies filter chains, streaming Raster → PDF

cups-filters Development: libcupsfilters

- **Improvements and Fixes**
  - **Streaming of data through filters**
    - On-demand via “**`filter-streaming-mode`**” option
    - **`cfFilterGhostscript()`**, **`foomatic-rip`**: Assume PostScript input, skip zero-page check
    - **`cfFilterGhostscript()`**: Use PCLm and raster PDF to stream Raster input to PDF
    - **`cfFilterPDFToPDF()`**: Skip QPDF treament, only insert JCL
  - **`CfFilterPDFToPDF()`**, **`cfFilterImageTo...()`**: Fixed page geometry: **`print-scaling`**, **`number-up`**, long-edge-first, **`landscape`**, **`orientation-requested`** ...

cups-filters Development: libppd

- **Auto-selecting best PPD option settings for job IPP attributes**
  - On loading PPD create **PPD option preset** for each
    - combo of **`print-color-mode`** and **`print-quality`**
    - value of **`print-content-optimize`**
  - **Auto-creation algorithm** finding **best settings** for ~10000 PPDs, no pre-building, no hand-editing
  - Used by the **retro-fitting Printer Applications**
  - Gets **best from classic drivers/PPDs with 3 job IPP attributes**, ideal for simplified print dialogs
- **Added “*.drv” PPD compiling from CUPS**
  - Easier retro-fitting of CUPS drivers

cups-filters Development: cups-browsed

- `implicitclass` **backend:** Using **filter functions** via `cfFilterUniversal()`, not external executables
- `implicitclass` **backend: Querying destination printer via IPP** for correct properties, now raster-only printers as destination work

cups-filters Development: cups-browsed

- **Planned**
  - **Avahi browsing/resolving optimization**: Get rid of unneeded, time-consuming resolving
  - Separate cups-browsed from cups-filters, into **own OpenPrinting GitHub project**
  - Separate cups-browsed from CUPS Snap into **own Snap**
  - Turn cups-browsed into a **Printer Application**

cups-filters Development: 1.x → 2.x

- **License change** to **Apache 2.0 + (L)GPL2 exception**, same as CUPS (approved by contributors)
- Cleaned up **naming style to match CUPS:**
  - API functions: “**`cfCamelCase()`**”
  - Library-internal functions: “**`_cfCamelCase()`**”
  - File-local functions: “**`underscore_separated()`**”
- Bumped **soname** to **2**
- First planned release **2.0b1**
- Silenced all compiler warnings
- All logging to **log function**, fixed log leaks to stderr

cups-filters Development: 1.x → 2.x

- **Re-structuring to get rid of PPD support**
  - **Currently:** Filter functions support PPD files for CUPS → **libcupsfilters depends on libppd**
  - **How to solve this?**
    - **First thought: Conditional compiling**
      - Distros want PPD-free libcupsfilters? Or PPD-supporting for Printer Apps as RPM/DEB?
    - **Solution: Re-structuring**
      - Remove PPD support from libcupsfilters → Original filter functions (“**`cfFilter...()`**”) w/o PPD support
      - Wrapper filter functions (“**`ppdFilter...()`**”) in libppd do PPD support and call orig. filter functions
      - CUPS 2.x and retro Printer Apps use libppd filters

cups-filters Development: 1.x → 2.x

- **Optional/Later 2.x release**
  - **“`cfFilterPDFToPDF()`” PDFio-based**
  - **libppd** in **separate** project
  - **cups-browsed** in **separate** project/Snap
  - Turn **cups-browsed** into **Printer Application**
  - Options for the **`./configure`** script for partial builds: No cups-browsed, no libppd/PPD support, no libqpdf, raster-only printing/scanning, ... to **allow Snaps build only the part of cups-filters which they actually need.**

We have agreed on **not to rename cups-filters** and **libcupsfilters**.

CUPS in a Snap

- A **Snap containing CUPS**, cups-filters, cups-browsed, Ghostscript, QPDF → **Complete CUPS printing stack**
- **No support for classic drivers**, as filters and PPDs cannot get dropped into Snap's file system → **Printer Applications**
- Sorting out all the problems with Canonical's Snap gurus on the snapcraft.io forum (see all links in README.md)
- Components **always up-to-date**, independent of release cycles: CUPS 2.4.x, cups-filters 2.x, Ghostscript 9.56.1, QPDF 10.5.0
- **Secure "`cups`" interface** for application Snaps to print
- Available in **Snap Store** "cups": https://snapcraft.io/cups

CUPS in a Snap Properties

- **Three run modes**:
  - **Stand-alone**: Snap’s CUPS is the only CUPS on the system, no classic CUPS present
  - **Proxy**: Classic CUPS present, Snap’s CUPS clones the queues, is firewall for the classic CUPS
  - **Parallel**: Classic CUPS present, Snap’s CUPS runs as second, indpendent CUPS (for development only)
- CUPS **always listens on Snap’s domain socket**, in stand-alone mode also on the standard domain socket and port 631 for unsnapped clients
- To not need to create system users and groups use snapd’s “snap_daemon” for “lp” user and “adm” for “lpadmin” group
- Adapted to Snap environment via cups-files.conf and file permissions, no patches, explicit Snap support built into CUPS upstream code

CUPS in a Snap
Properties

- All **System-V-** and **Berkeley-style command line tools**, also **special tools** cupsfilter, driverless, ippfind, ipptool, ippeveprinter, ippproxy
- **cups-browsed** included, always attaching to the Snap’s CUPS
- The **CUPS Snap on OpenPrinting** is **integral part** of the **Snap eco-system** as it is required for the “cups” snapd interface

CUPS in a Snap
Security concept

- **Snaps are usually completely confined** and can communicate only through **defined interface connections**
- **Everyone can upload Snaps** to the Snap Store **but**
  - On the user’s system only “**safe**” interfaces of downloaded Snaps **connect automatically**
  - “**Dangerous**” interfaces need to get **connected manually** after Snap install (if they do not have auto-connect permission from the Snap Store team)
  - **Unconfined** (“classic”) need **permission** of the Snap Store team
- For **using CUPS from Snaps** there are two interfaces:
  - “**cups**”: For user application Snaps which print (**safe**)
  - “**cups-control**”: Admin access to cupsd (**dangerous**)

CUPS in a Snap
Security concept

- Most **Snap interfaces** are defined only by **AppArmor rules**, but
- “cups” vs. “cups-control” → **Snap Mediation**
  - If cupsd receives an **administrative request** it accepts it only if
    - The client is **no Snap** or a **classically confined Snap**
    - The client connects via “**cups-control**”
- User’s system **usually has classic CUPS**, not CUPS Snap → **No Snap Mediation**, therefore
  - “**cups**” interface only connects to **Snap’s domain socket**
  - Application Snap installation **dependency-installs CUPS Snap**
  - **CUPS Snap** in proxy (**firewall**) mode and mediates requests
  - User stays with **his queues** and (often proprietary) **drivers**

CUPS in a Snap as a distro's CUPS

- **What is needed:**
  - **DONE: Security concept** on the snapd side completed
  - **DONE: All drivers** available on Debian retro-fitted into **Printer Applications** (only Braille embossers missing)
  - **GUI tools:** GNOME Control Center "Printers & Scanners" WIP, **CPDB** for PPD-free print dialogs
  - **Look-up service for Printer Applications** on OpenPrinting web site planned
    - No follow-up on hardware-look-up feature request for Snap Store
    - Could support also other platforms, like Docker
- **Rehearsal for CUPS 3.x in a distro** (no PPD/driver support)

CUPS, Printer Applications, …
Snap only ???

- **Snap is a sophisticated package format**, supports CLI apps, system daemons … Like phone apps … **BUT:**
  - **Slow start-up of desktop apps** (esp. Firefox, Chrome)
  - **Only one Snap Store**
- Investigated **other formats**
  - **Flatpak**
    - Very common format for **desktop apps**
    - System access via **GUI portals** (GNOME, KDE) → **Not suitable for system daemons**
    - **Atomic distros for Flatpak use:** Possibility to add system daemons as OCI container image via Docker or podmap → **Official OpenPrinting images on DockerHub needed**

Printer Applications

- **PAPPL** got standard framework
  - PAPPL provides **everything required in a library**
  - **Driver developer** only has to do the **printer-specific parts**
  - **Tutorial** for manufacturers/driver developers written in GsoD 2020
- **pappl-retrofit**: Printer driver retro-fit library
  - **PPD handling**: Listing, filtering, selecting, options, installable accessories, CUPS extensions, drivers…
  - **Map job IPP attributes to best PPD option settings**
  - Calling external **CUPS driver filters** and **backends**
  - Printer App easy to create, minimum C code required

Printer Applications

- **Current Printer Aplications**:
  - **Retro-fitting all free drivers available in Debian**
    - PostScript Printer Application – ~4000 manufacturer PPDs
    - Ghostscript Printer Application – All the rest
    - HPLIP Printer Application – Proprietary plugin, no scanning
    - Gutenprint Printer Application – Epson, Canon, Dye-Sub, ...
  - **Native Printer Applications**
    - Lprint – Label printers
  - Map **classically installed (also proprietary) drivers** into a Printer Application (not available as Snap)
    - Legacy Printer Application
- For **unmaintained drivers** wrap filters and PPDs into Printer Application via **pappl-retrofit**
- **Native Printer Applications** for **maintained drivers**

Driverless Scanning

- **3 Standards**
  - **IPP Scan**, open PWG standard
  - **eSCL**, proprietary, from HP, specs published by Mopria
  - **WSD**, from Microsoft and W3C
- All are mainly intended for **multi-function printers**
- **eSCL** and **WSD** one **already available** in **AirScan** devices
- **2 SANE drivers** for eSCL: “escl” from Thierry Hucahrd and “airscan” from Alexander Pevzner, both in most distros
- Alexander has added **WSD support** and will add **IPP Scan** if needed/required
- At least **eSCL** also works via **IPP-over-USB** (ipp-usb)

Sandboxed Scanner Drivers

- **Current situation: SANE**
  - Scanner driver (**SANE backend**) is shared library
  - Scanning app (**SANE frontend**) links backends dynamically
  - **To add a driver it needs to be dropped in backend dir** => not good for sandboxed packaging
- **New scanning environment: eSCL/IPP Scan driverless**
  - Scanner drivers in **Scanner Applications, emulating driverless scanner**
  - Scanning app is **eSCL/IPP Scan client**
  - Legacy: App uses only sane-airscan SANE backend, SANE drivers enclosed in legacy Scanner Application
  - **Scan support in PAPPL is WIP**

IPP-over-USB: ipp-usb

- Ipp-usb written in **Go**, as Go has **sophisticated HTTP library** to read out buffer on closed connection
- Ipp-usb **works perfectly**, esp. web admin interface
- **Chrome OS not accepting software in Go** due to high memory footprint → Own approach in Rust
- **ippusbxd** development **discontinued**
- **eSCL scanning** and **IPP Fax Out** work with ipp-usb
- **Note all 7/1/4 USB printers do driverless** (not standard-conforming), e. g. HP Laser series (Wi-Fi works, firmware bug?)
- **ipp-usb Snap** available
  - Uses UDEV-watching script to replace missing UDEV rule support

Printing GUIs
What do we need?

- **Print dialog**: We need to get **CPDB** into GTK and Qt
- **Printer Setup Tool**
  - **Main Window**
    - List all IPP services as reported by DNS-SD, list Printer Applications and their queues in a group, no duplicates for IPv4/IPv6, IPPS, interfaces
    - Buttons for web interface, add new queue, show jobs …
  - **Add Printer Wizzard**
    - List of discovered non-driverless USB/network printers, click button to see list of Printer Applications supporting the printer, installed ones and available in Snap Store (look-up service on OpenPrinting)
    - Buttons to setup printer with selected Printer Application and to install Printer Application from Snap Store

Printing GUIs
GNOME Comtrol Center

- New “**Printers & Scanners**” module for the New Architecture replaces old “Printers” module
- **Three main parts:**
  - **Main screen: List IPP services** (printing, scanning, fax out) by device – Divyasheel, GSoC 2021
  - **IPP System Service configuration dialog** – Lakshay Bandlish, GSoC 2020
  - **Add Printer Dialog**, adding and managing non-driverless printers – GSoC 2022 ???
- Also support by the **Canonical Desktop and Design Teams**

Questions / Comments