OpenPrinting

The New Architecture for Printing and Scanning

What GNOME/GTK Developers Need to Know

Till Kamppeter – OpenPrinting

GUADEC, July 21, 2022

Introduction

- **The New Architecture – Pure IPP for Printing and Scanning**
  - CUPS 3.0/CUPS Snap: R. I. P. PPD files
- **Printer Setup Tools**
  - IPP Services, not Queues
- **Print Dialogs**
  - IPP Attributes, Temporary Queues, CPDB
- **Sandboxed/Distribution-Independent Packaging**
  - Snap, Flatpak, AppImage ...

The New Architecture

- For 22 years now, since its 1.0 launch, CUPS uses **principally the same architecture**:
  - **PostScript was standard job format** as printers typically used with UNIX were PostScript
  - **Capabilities of a printer are described by a PPD** (PostScript Printer Description) file
  - **PPD** describes all **user-settable options, resources** (trays, paper sizes, resolution, quality, color, …) in a **static text file**
  - To cover **non-PostScript printers PPD format got extended** (by Michael Sweet) to **specify a filter** to generate printer’s native format
  - Filters use **Ghostscript** (or **Poppler**) to convert PostScript input
  - **Manually created queue** with driver (= PPD + filter)

The New Architecture

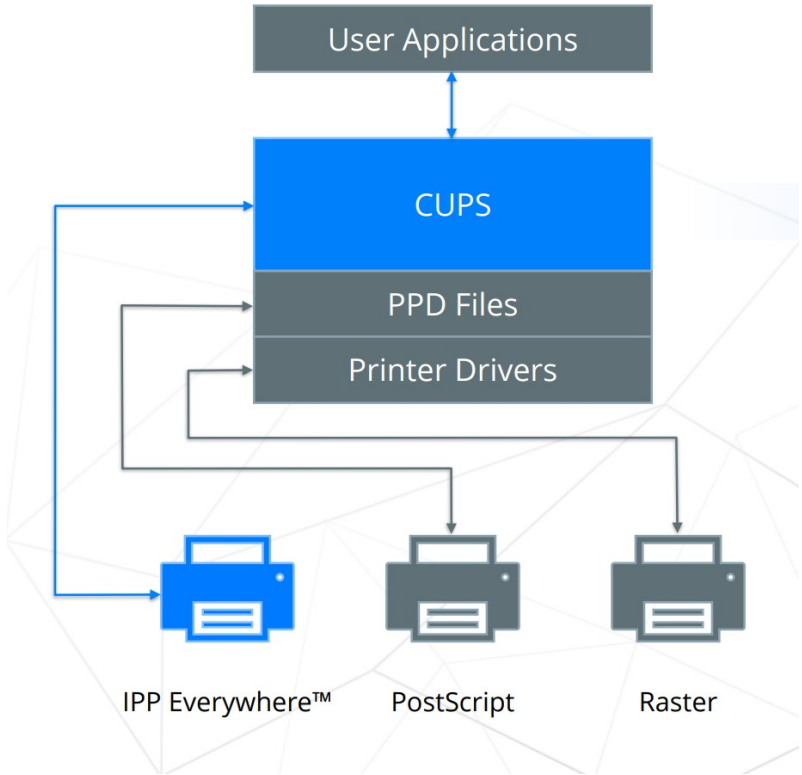
- Why do we want to do away with PPD files?
  - **In 1984 Adobe stopped development on PPDs**, so we started with an obsolete (but useful) format right away
  - In 2006 **we abolished PostScript** as print job format and **replaced it by PDF**
  - PPD files can represent user-settable options only as **enumerated choice** or **boolean**. Ugly workarounds for things like passwords or color adjustment

The New Architecture

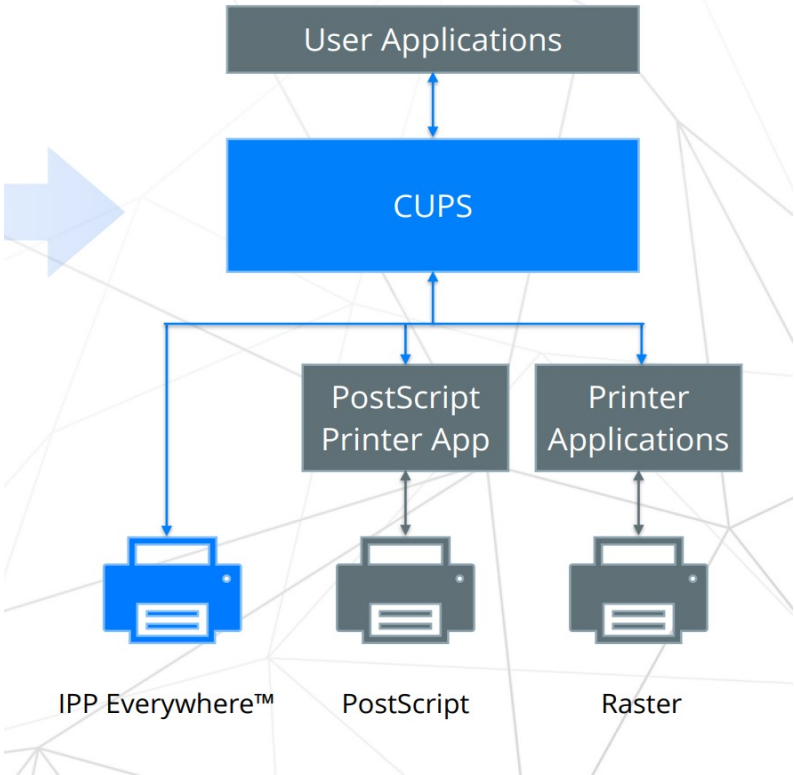
- **PPD-less** CUPS – We are **all-IPP** now
  - **CUPS 3.0.x** will not support PPD files from the ground up
  - The **CUPS Snap** does not allow adding PPDs and filters
  - Now **only driverless IPP printers** (IPP Everywhere, AirPrint, Mopria) are supported
  - **No manually created CUPS queues**: IPP printer discovered, temporary queue automatically created
  - Filtering only for driverless standard formats: PDF, PWG Raster, Apple Raster, PCLm output, **no need to add filters**
  - Legacy/specialty printers which need driver → **Printer Application** emulates IPP printer

The New Architecture

Old CUPS architecture



New CUPS architecture



Printer Setup Tool: How it works currently

- **Printer setup tools**
  - CUPS web admin interface **`http://localhost:631/`**
  - CUPS command line tools: **`lpadmin`**, **`lpinfo`**, **`lpstat`**
  - **`system-config-printer`** - GUI
  - GNOME Control Center – Print module - GUI
  - **`cups-browsed`** – daemon, auto-creating queues
- Tools **control CUPS**, the running **cupsd**
  - List available printers and drivers and create print queues
  - List queues and jobs
  - Modify queues
  - Server settings: Owner/everyone can cancel jobs, debug mode, ...

Printer management in the New Architecture

- We assume any form of the New Architecture
  - The **CUPS Snap** - OR -
  - **CUPS 3.x** or newer
- **All Printers are driverless IPP printers**, native or Printer Applications
- CUPS auto-creates virtual queue for each IPP printer → **No manual queue creation required**
- CUPS fully automatic → **Admin action moves to the IPP printers**
- **Tasks**
  - **List IPP services**
    - Buttons to web admin interfaces, IPP System Service, …
  - **Discover non-driverless printers**
    - Find Printer Applications, local and in Snap Store

Printer Setup Tool: GUI Design

- **Similarities** between old and new
  - **Main Window**
    - Old: List CUPS queues, buttons/pop-up to modify
    - New: List IPP devices, buttons to web IF/IPP System Service
  - **Add-Printer Window**
    - Old: List printer devices and drivers, create CUPS queue
    - New: List non-driverless printers, install Printer Application, open Printer Application’s web interface
- **Multi-function devices, scanners, fax out** (optional)
  - Functions are all IPP services, list functions of same device together
  - Scanners and faxes are managed like printers

Printer Setup Tool:
GNOME Control Center - Printers

- Probably the **most-used printer setup tool**
- **Currently** it
  - lists **permanent CUPS queues**
  - lists **IPP services**, with PPD config, but no web interface access, ...
  - adds new print queues based on **classic driver with PPD**
- **First approach** for New Architecture was **total replacement of the module**
  - GSoC 2020: **IPP System Service** GUI
  - GSoC 2021: **Main panel listing only IPP services** with appropriate control functionality
  - GSoC 2022: Add non-driverless printer with **Printer Application**

Printer Setup Tool: GNOME Control Center - Printers

- **Discussion with upstream developers** (mainly **Marek Kasik**) results in better idea: **Hybrid printer setup tool supporting old and new**:
  - Lists **both IPP services** leading to temporary queues and **permanent CUPS queues**
  - To add **non-driverless printer** it can use **classic driver/PPD or Printer Application**s
  - Non applicable features are simply not shown
  - **Smooth transition**: First new G-C-C “Printers”, then CUPS 3.x (or Snap).
- **Two GSoC contributors** are currently working on it to complete it:
  - **Shivan Mishra**: Main panel, IPP service listings with appropriate control points
  - **Mohit Verma**: Add non-driverless printer with Printer Application

Printer Setup Tool: GNOME Control Center - Printers

- Development discussed and managed in **feature requests** on GitLab:
  - #1877: Improve setting of IPP options
  - #1878: Allow to add new printers via Printer Applications
  - #1879: Do not show setting of drivers for IPP printers
  - #1911: Printers: Make adminurl available for IPP printers
- #1878 is about **adding use of Printer Applications**, both **locally installed** and on the **Internet** (distro packages, Snap Store, OpenPrinting look-up service)
- The other 3 are about adding the **correct control/configuration functionality for IPP services**
- We decided for **not listing scanners** here as users see physical devices, not IPP services

Print Dialogs: Direct adaptation

- Print queues are usually **temporary**, for **discovered IPP services** (IPP printers or Printer Applications)
  - Some print dialogs still use **stone-old CUPS APIs**, not supporting temporary queues, and temporary queues exist for years
  - **GTK dialog** has this fixed
  - But applications with **too old GTK versions** still around
  - **cups-browsed** used as workaround, making all queues permanent, so be careful, some dialogs do well due to cups-browsed
- On CUPS 3.x there are **no PPD files at all**
  - Dialogs should not try to download the printer’s PPD from CUPS. The APIs or URLs will go away with CUPS 3.x
  - Use **modern CUPS convenience APIs** or **IPP** to get capabilities/options

Print Dialogs: The problem

- To control printing, GUI applications use **print dialogs**
- **Many different print dialogs**, usually from the GUI toolkit used (GTK, Qt, ...), but also LibreOffice, Chrome, ...
- Each one has **its own implementation** to connect to CUPS, Print-to-File, and other print technologies
- Print dialog **development does not keep up** with changes, like temporary queues in CUPS, or addition of a new print technology (cloud service, ...)
  - Printing not considered very important
  - Newly introduced print technology not considered worthwhile
  - Developers do not have time
  - Long release cycles of GUI toolkit projects vs. fast pace in printing development

Print Dialogs: The idea → CPDB

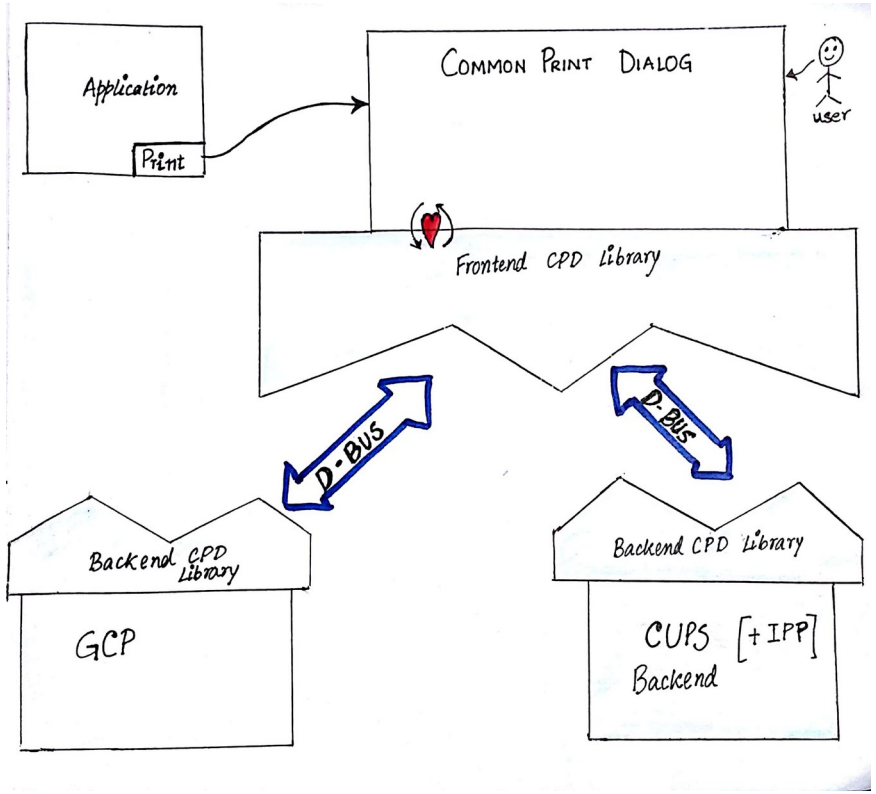
- Long time ago we tried a **Common Print Dialog**, but **failed** due to lack of human resources and/or funding (Flatpak did it finally)
- Later **Aveek Basu** remembered this project and **suggested a revival**, but I was unsure.
- Fixing a CUPS-related bug in the **GTK print dialog** I discovered that it uses **backends** for different print technologies
- All this brought up the idea of **Common Print Dialog Backends** in me:
  - Dialog itself still from the GUI toolkits (GTK, Qt, LibreOffice, ...)
  - **GUI-independent backends** for each print technology (CUPS, Print to file, ...)
  - Connection Dialog ↔ Backend: **D-Bus** (separately sandboxable)
  - Backend and frontend libraries

Print Dialogs: The idea → CPDB

- Backends maintained by **maintainer of print technology**
  - CUPS backend: OpenPrinting
  - GlobalCloud Print backend: GlobalCloud
  - …
- **Print dialog detects installed backends** and shows the printers of the respective print technologies
- **User sees always the same printers** with the same user-settable options in all print dialogs (GTK, Qt, LibreOffice, …)
- Print service provider can **supply backend via Snap Store**
- Maintainer of print technology changes something → He issues backend update and all print dialogs are up-to-date

Print Dialogs: CPDB - The implementation

- I posted this as a project idea in the **Google Summer of Code 2017** ...
- ... and **Nilanjana Lodh** picked it up and implemented it (her original drawing):



Print Dialogs: CPDB - The Implementation

- Libraries are on the **OpenPrinting GitHub**
  - Frontend/Backend libraries: **`cpdb-libs`**
  - CUPS backend: **`cpdb-backend-cups`**
  - Print to file backend: **`cpdb-backend-file`**
- There are also **packages in Ubuntu** (Universe)

Print Dialogs: CPDB - The new problem

- **No one** did a frontend implementation to submit to the GUI toolkit projects GTK and Qt yet.
- CPDB support in dialogs will save us from problems like
  - CUPS added the new **`cupsEnumDests()`** API to **support its temporary queues** many years ago, GTK switched to it this year, **Qt (and perhaps others) did not switch yet** (needs checking).
  - The architecture of CUPS will significantly change with version 3.0 ...
- **GSoC 2022**: Gaurav Guleria works on **improving the CPDB libraries** and adding **support for CPDB to GTK, Qt, and perhaps more print dialogs**

Application Distribution: Snap

- Snap is the **most sophisticated method** for distro-independent packaging
  - Most similar to **smartphone application packaging**
  - **Consistent security concept**, **interfaces** (network, cups, dbus, …) have to be connected fo communicating with system or other Snap
    - “Safe” interfaces (like “cups”) are auto-connected when installing from Snap Store
    - “Dangerous” interfaces (like “cups-control”) need manual connection
  - Only method which allows packaging **system daemons**, like CUPS
  - **Printing support well implemented and integrated**:
    - “cups” interface allows **secure printing**, app cannot mess up CUPS
    - “cups-control” for **printer setup tools**, allows full access to CUPS
  - Disadvantages: **Only one Snap Store**, sometimes **slow start-up**

Application Distribution: Snap

- **Printer Application**: A daemon emulating an IPP printer, replaces classic CUPS drivers
- **Scanner Application**: A daemon emulating an IPP or eSCL scanner, replaces classic SANE drivers
- Printer Application for **multi-function device** fulfills both roles
- **Classic printer or scanner drivers cannot be snapped**, as they are files (PPDs, filter, libraries) to be dropped into defined system directories
- Printer/Scanner Applications **communicate** with CUPS or with scan frontend **via IP**, so they also work in a Snap
- **→ Secure and distribution-independent way for manufacturers to provide drivers** (via the Snap Store)

Application Distribution: Flatpak

- Similar to Snap, applications come with all their libraries, encapsulated
- **Communication to the outside via so-called “portals”**, usually standard GUI dialogs: “Open”, “Save as”, “Print”, … (esp. printing via portal)
  - **Only suitable for interactive GUI applications**
  - Application needs to use standard GUI APIs so that Flatpak extensions of GUI toolkits are used (use D-Bus to portal if running in Flatpak)
  - Invasive patching needed if these APIs and standard toolkits (GTK/Qt) no used
  - **No system daemons**, command line utilities, server software, … can get Flatpaked.
  - **Print security**: Only prints through print dialog, no direct CUPS access
  - → **No CUPS Flatpak, no printer/scanner drivers via Flatpak**

Application Distribution: AppImage

- **Simplest system**: User downloads app as one file and starts it
- The file contains squashfs file system with app and all libraries
- **No security concept**, like AppArmor or similar
- So not recommended to grab from internet and run as root, therefore **not suitable for system daemons**, despite they probably even work
- **Printing should just work** due to lack of security concept

→ Secure printing only with Snap or Flatpak

→ Currently only Snap is suitable for packaging CUPS and printer/scanner drivers

→ Non-GUI user applications only with Snap or AppImage

→ Need alternative system (like Docker) for Flatpak-based distros

Questions / Comments / More Info

- http://www.openprinting.org/: Our site
- https://openprinting.github.io/about-us/: What OpenPrinting is doing
- https://openprinting.github.io/news/: The latest and greates, monthly