# CUPS Plenary

Joint PWG/Open Printing Meeting
Sunnyvale, CA
May 15, 2018

*Michael Sweet, Apple Inc.*

# CUPS Plenary

*Topics*

- Introduction
- CUPS 2.2
- CUPS 2.3
- CUPS Deprecations
- CUPS Future
- Q&A
- Resources

# CUPS Plenary

## *Introduction*

- CUPS is the standards-based, open source printing system developed by Apple Inc. for macOS® and other UNIX®-like operating systems.
- The CUPS web site, source code, and bug database are hosted on Github
  - https://www.cups.org/
  - https://github.com/apple/cups/

# CUPS Plenary

## *CUPS 2.2*

- CUPS 2.2 is the current stable branch with patch releases every 2-3 months:
  - CUPS 2.2.0 was released September 13, 2016
  - CUPS 2.2.4 was released June 30, 2017 with IPP Everywhere and security improvements
  - CUPS 2.2.5 was released October 13, 2017 with IPP Everywhere and reliability improvements
  - CUPS 2.2.7 was released March 23, 2018 with more IPP Everywhere, reliability, and security improvements

# CUPS 2.2

*License*

- CUPS 2.2.x and earlier continue to use the GNU GPL2/LGPL2 with exceptions for software on Apple operating systems (macOS, iOS, tvOS, etc.)
- Future CUPS security fixes will also be explicitly provided under the same license
  - File Github issues for other important bug fixes (larger than a few lines of code) that you need from a newer Apache-licensed CUPS release

# CUPS 2.2

*Developer Resources*

- "New" CUPS Programming Manual
  - https://www.cups.org/doc/cupspm.html
  - https://www.cups.org/doc/cupspm.epub
- Documents CUPS APIs as well as best practices
- Includes lots of examples
- Please provide feedback via Github and/or cups-devel mailing list

# CUPS 2.2

## *CUPS API Changes*

- CUPS 2.2.4: Updated `cupsConnectDest` function to support the `CUPS_DEST_FLAGS_DEVICE` flag
- CUPS 2.2.4: Updated the `cupsGetDests`, `cupsCreateJob`, `cupsPrintFile`, and `cupsPrintFiles` functions to support Bonjour printers
- CUPS 2.2.4: Added the "printer-is-temporary" Printer Status attribute to the destination options

# CUPS 2.2

## *CUPS API Changes, con't*

- CUPS 2.2.4: Added the `cupsAddIntegerOption` and `cupsGetIntegerOption` functions to set and get integer option values:

```
int
cupsAddIntegerOption(const char *name,
    int value, int num_options,
    cups_option_t **options);

int
cupsGetIntegerOption(const char *name,
    int num_options, cups_option_t *options);
```

# CUPS 2.2

## *CUPS API Changes, con't*

- CUPS 2.2.7: Added the `cupsHashString` function to convert binary hashes to a hexadecimal string:

```
const char *
cupsHashString(const unsigned char *hash,
               size_t hashsize, char *buffer,
               size_t bufsize);
```

# CUPS 2.2

## *IPP Everywhere Changes*

- CUPS 2.2.3: Updated to use all "print-quality" and "media-type" values reported by the printer
- CUPS 2.2.3: Fixed mapping of "finishings" values
- CUPS 2.2.4: Updated to match PWG Raster capability values using case-insensitive comparisons
- CUPS 2.2.5: Improved mapping of printer resolutions to "print-quality" values
- CUPS 2.2.5: Updated to include the HTTP Date: header in IPP requests

# CUPS 2.2

*IPP Everywhere Changes, con't*

- CUPS 2.2.7: Added IPP "job-password" and "job-password-repertoire-configured" support
  - The default repertoire is 'iana_us-ascii_digits'
- CUPS 2.2.7: Added PWG Raster wide/deep color support
  - 16-bit per color and Adobe RGB
- CUPS 2.2.7: Updated to detect 'document-format-error' and 'document-unprintable' reasons after submitting a print job so that complex PDFs that fail are reprinted automatically using the PWG Raster/Apple Raster format instead

# CUPS 2.2

## *Scheduler Changes*

- CUPS 2.2.5 and 2.2.7: Added various Avahi-related crash fixes
- CUPS 2.2.7: Added substitution of default values for invalid values in "relaxed" conformance mode
  - Typically for non-UTF-8 strings being passed as UTF-8, e.g., "requesting-user-name" = 'iso-8859-1 user name'
  - Prevents a lot of DBUS-related crashes

# CUPS 2.2

## *Security Changes*

- CUPS 2.2.4: Added support for "`SSLOptions DenyCBC`" and "`SSLOptions DenyTLS1.0`" in client.conf and cupsd.conf
- CUPS 2.2.5: Fixed "`ServerTokens None`" support in cupsd.conf (privacy bug)
- CUPS 2.2.7: Added support for "`SSLOptions MinTLS`" and "`SSLOptions MaxTLS`" options in client.conf and cupsd.conf
- CUPS 2.2.7: Added TLS negotiation timeouts for all types of HTTP connections

# CUPS 2.2

## *Security Changes, con't*

- CUPS 2.2.7: Added support for multiple authentication schemes to allow support for other authentication schemes in the future like OAuth 2.0, MutualAuth, etc.:
  - Comma-delimited in a single HTTP WWW-Authenticate: header, or
  - Multiple HTTP WWW-Authenticate: headers concatenated as required by the HTTP specifications
- CUPS 2.2.7: Updated HTTP Basic and Digest support to the current RFCs
  - Digest is still only supported client-side

# CUPS Plenary

## *CUPS 2.3*

- CUPS 2.3 is the next feature release:
  - CUPS 2.3.0 tentatively scheduled for June/July 2018
  - Additional 2.3.x updates planned through the end of 2019
- Primary Focus of CUPS 2.3:
  - License Change
  - IPP Everywhere
  - Print Accounting
  - Scheduler

# CUPS 2.3

*License Change*

- CUPS 2.3 and later will be distributed under the terms of the Apache License Version 2.0
  - Eliminates compatibility issues with projects that use GPL3, LGPL3, AGPL3, and the Apache License Version 2.0
- Still working out the details to allow GPL2/LGPL2-only software to link to the CUPS libraries...

# CUPS 2.3

## *IPP Everywhere*

- Localization of attributes and values, including printer-specific values from a printer's .strings files
- IPP Job Presets support
- IPP "finishing-template" support
- Closing of any remaining CUPS API "holes" preventing applications from using IPP Everywhere instead of printer drivers
- Bug fixes

# CUPS 2.3

## *Print Accounting*

- The scheduler now tracks the total number of media sheets and only logs the count once a job completes
  - The previous mix of progress (from filters) and total (from printer) values could yield incorrect accounting results
  - Also simplifies accounting software that uses the page_log file - now just a single line for each job that is printed

# CUPS 2.3

## *Scheduler*

- Now generate a per-printer .strings file for client-side localization
- Bonjour (sharing) host name can now be set
- Now support the "printer-id (integer(1:65535))" Printer Status attribute
- Scripted CGI programs are supported differently:
  - Now rely on execute bit and #! header
  - No more hardcoded script interpreters

# CUPS 2.3

## *CUPS API*

- Improved media selection support, including a new function:

```
int
cupsAddDestMediaOptions(http_t *http,
    cups_dest_t *dest, cups_dinfo_t *dinfo,
    unsigned flags, cups_size_t *size,
    int num_options, cups_option_t **options);
```

- New option encoding function:

```
ipp_attribute_t *
cupsEncodeOption(ipp_t *ipp,
    ipp_tag_t group_tag, const char *name,
    const char *value);
```

# CUPS 2.3

*CUPS API, con't*

- `cupsCopyDestConflicts` now supports collection attribute ("media-col", "finishings-col", etc.) constraints
- HTTP header values can now be longer than the old static limit (`HTTP_MAX_VALUE`)
- The '`-D_IPP_PRIVATE_STRUCTURES=1`' cheat no longer works when including the `<cups/ipp.h>` header
  - The `ipp_t` and `ipp_attribute_t` structures are now fully private (moved to private header)
  - Use the accessor functions (`ippGetXxx`/`ippSetXxx`) which were added in CUPS 1.6 *eight* years ago

# CUPS 2.3

*CUPS API, con't*

- The '-D_PPD_DEPRECATED=""' cheat for the `<cups/ppd.h>` header file no longer works
  - There is no longer a way to disable compile-time warnings when using the PPD functions
  - The CUPS destination APIs are the replacement for all PPD functionality and have been since CUPS 1.4 which was released *ten* years ago

# CUPS Plenary

## *CUPS Deprecations*

- We periodically deprecate functionality that either is no longer necessary or will prevent us from improving CUPS
- When we deprecate something:
  - We announce the deprecation as far in advance as possible
  - We display a warning that the functionality is going away in a future release of CUPS
  - We help developers and users migrate to any replacement functionality, if applicable
- Deprecation is a necessary step prior to removal from CUPS
- *Deprecated items are still functional until removed*

# CUPS Plenary

## *CUPS Deprecations, con't*

- After a transition period, deprecated items are removed from CUPS
  - Deprecated CUPS APIs are never fully removed from shared libraries (non-functional stubs remain) to preserve binary compatibility
- We've had some hard exceptions over the years:
  - Security issues forced us to do a hard transition of some cupsd.conf directives to cups-files.conf
  - Security issues forced us to drop interface script support
  - Performance and architectural issues forced us to drop CUPS browsing before Avahi was fully supported/deployed

# CUPS Deprecations

## *CUPS 2.2.7: Deprecate Raw Print Queues*

- *Raw queues will continue to work in CUPS 2.2.x/2.3.x*
- Why deprecate them?
  - Raw queues pointing to shared printers cause problems for sandboxed applications on macOS and applications using AppArmor/SELinux on Linux (no direct network access)
  - Raw queues pointing to label printers, etc. require applications to provide printer-specific UI and print data, the opposite of what CUPS is about
  - Raw queues do not work with file: device queues, which people still occasionally use with special-purpose printers and software

# CUPS Deprecations

## *CUPS 2.3: Deprecate Printer Drivers*

- *Printer drivers and the PPD APIs will continue to work in CUPS 2.3.x*
- PPD files were deprecated in CUPS 1.4 (ten years ago) but we didn't have a replacement strategy for printer drivers at that time
- IPP Everywhere (and related standards) provide the replacement for most printer drivers
  - Strategy for other printers and drivers is to use Printer Applications
- We hope to remove printer driver support in the CUPS feature release following 2.3.x

# CUPS Deprecations

## *CUPS 2.3: Deprecate Printer Drivers, con't*

- Why deprecate printer drivers?
  - At least 98% of all printers sold since 2010 support IPP, Apple/PWG Raster, and JPEG; many (about half) support PDF
    - Holdouts are industrial label printers and certain vertical market printers
  - PPDs and drivers have been holding us back from offering a better user experience (ready media, localization, full range of printer options/values), improved document processing, and improved accounting
  - PPDs and drivers are a security and distribution nightmare

# CUPS Deprecations

## *CUPS 2.3: Deprecate Printer Drivers, con't*

- A Printer Application:
  - Is a standalone replacement for printer drivers
  - Provides an IPP printer endpoint
    - The CUPS library contains everything needed to implement a lightweight IPP Everywhere Printer that can be used by any client running CUPS 1.4 or later
- Supports setup, configuration (e.g., presets for custom media/ink sets), operator functions, and status information UI

# CUPS Deprecations

## *CUPS 2.3: Deprecate Printer Drivers, con't*

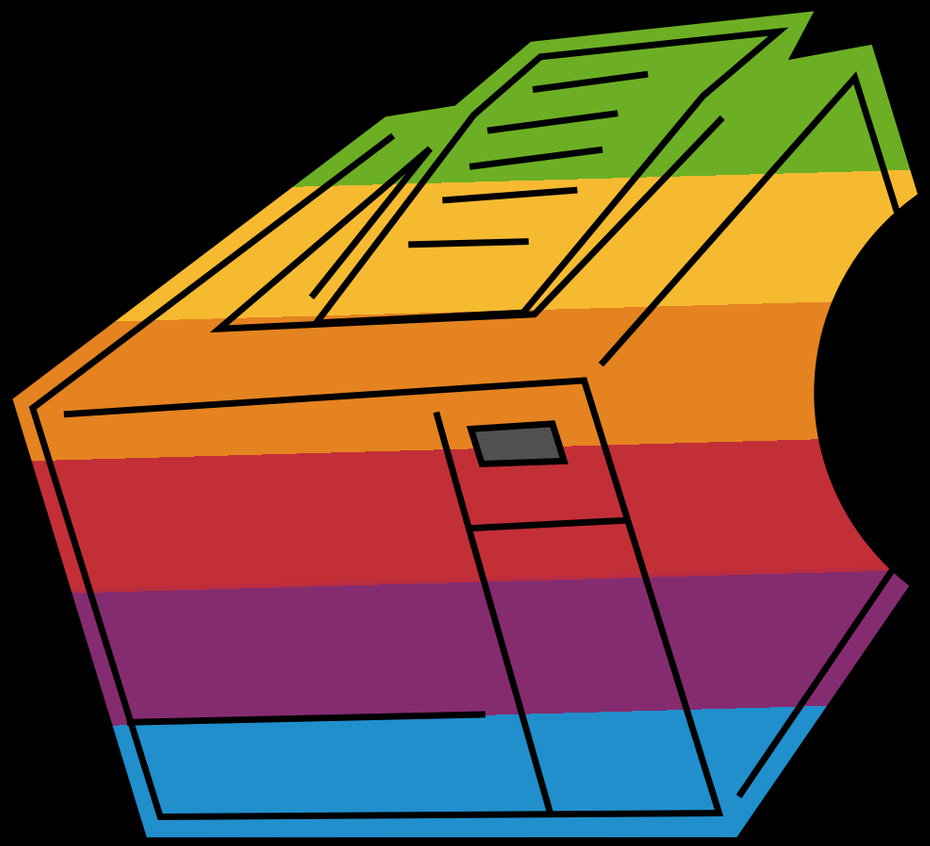
- A Printer Application:
  - Can accept high-level (PDF/JPEG) or PWG Raster print files that are generated and sent to it by the print client
  - Can be distributed via the Mac App Store, Homebrew project, Linux distribution packages, snapcraft, AppImage, Docker, etc.
- Can support any number of printers
- Should provide a more efficient and better functioning solution that we have today with PPD-based printer drivers

# CUPS Future

# CUPS Plenary

## *CUPS Future*

- CUPS needs to change to adapt to a more mobile computing world that does different kinds of printing in different environments
- Major focus areas:
  - Power Usage
  - Mobility and Scalability
  - Discovery
  - Security
  - Simplification



# Print different.

# CUPS Future

## *Power Usage*

- Current CUPS does a great deal to minimize power usage, but there is room for improvement:
  - Job Processing - optimize common filter paths with standalone filters rather than stringing several filters together, which also provides better printing performance with reduced data copying and fewer user/kernel-space transitions
  - Idle Wait - exiting when idle is not the most power efficient design, so we should instead respond to memory/resource pressure (available on macOS at least)

# CUPS Future

*Mobility and Scaling*

- Mobile computing has a dramatic effect on networking
  - Multiple networks/interfaces/protocols/addresses
  - Printer visibility can change in moments
- Need to look beyond the BSD sockets API, including things like the PostSockets API proposed in the IETF:
  - https://tools.ietf.org/html/draft-trammell-taps-post-sockets

# CUPS Future

*Mobility and Scaling, con't*

- The full CUPS printing stack currently doesn't scale down well:
  - Still depends on a variety of (large) filter and driver packages
  - CUPS itself includes a lot of functionality that is not needed by most mobile/desktop users such as the web interface and printer sharing
- It would make sense to separate out the server/sharing bits and driver support to provide a smaller, more efficient stack when sharing is not needed

# CUPS Future

*Discovery*

- DNS-SD/Bonjour is the primary mechanism today but doesn't scale well on typical multi-segment networks and only works for "public" printers
  - Some work happening in the IETF DNS-SD workgroup to address this
  - https://tools.ietf.org/wg/dnssd/
- LDAP has also been on the roadmap for a while
  - Allows filtering available printers using an authenticated user identity
  - https://tools.ietf.org/html/rfc7612

# CUPS Future

*Discovery, con't*

- Also want to do proper support for static/distributed "printer profiles"
- Printer profiles:
  - Can point to one or more servers and/or printers
  - Can be distributed in many ways
  - Can be system-wide or per-user
  - Don't depend on DNS-SD, LDAP, or other infrastructure

# CUPS Future

## *Security*

- Known Issues
  - Printer Drivers
  - Scheduler Running as Root
  - Web Interface
- OAuth 2.0

# Security

## *Printer Drivers*

- Often make assumptions about the print user being logged in at the console/single-user environment
- Sandboxed on macOS, some protection provided by AppArmor and SELinux
  - Still typically have (enough) network and file system access to (at least) pose a disclosure risk
- No CPU, disk, or memory limits imposed
- Historically most drivers are well behaved, but a few buggy ones have caused performance issues in the past

# Security

## *Scheduler Running as Root*

- Currently the scheduler (cupsd) must run as root to:
  - Bind to port 631 and /var/run/cupsd
  - Run as other users for Kerberized printing
  - Support using a privileged source port for LPD printing
- Solutions:
  - launchd, systemd, and upstart can all handle binding to port 631 or a domain socket for cupsd
  - launchd, systemd, and upstart can all handle starting per-user instances of cupsd
  - Stop supporting printers that need LPD

# Security

## *Web Interface*

- Many browser based attacks identified over the years
  - When the same host and port combination are used for the web interface and IPP endpoint, the browser thinks they are the same "origin" making it difficult to fully protect against browser-based attacks
  - CUPS includes many mitigations, but we have disabled the web interface by default to limit out-of-the-box exposure
- Difficult user experience issues
  - No way to manage different user logins in the same session or "logout" using current HTTP Basic or Kerberos authentication

# Security

## *OAuth 2.0*

- Possible solution for web interface security issues
- Possible replacement for Kerberos
  - though we need to determine whether SMB/SPOOLSS and OAuth 2.0 co-exist today - seems so for Windows 10 but not for older Windows releases, Samba, or SMB on macOS
- Doesn't require root access
- Also used for common Cloud printing solutions
- Many open source solutions available, including my own:
  - https://michaelrsweet.github.io/moauth

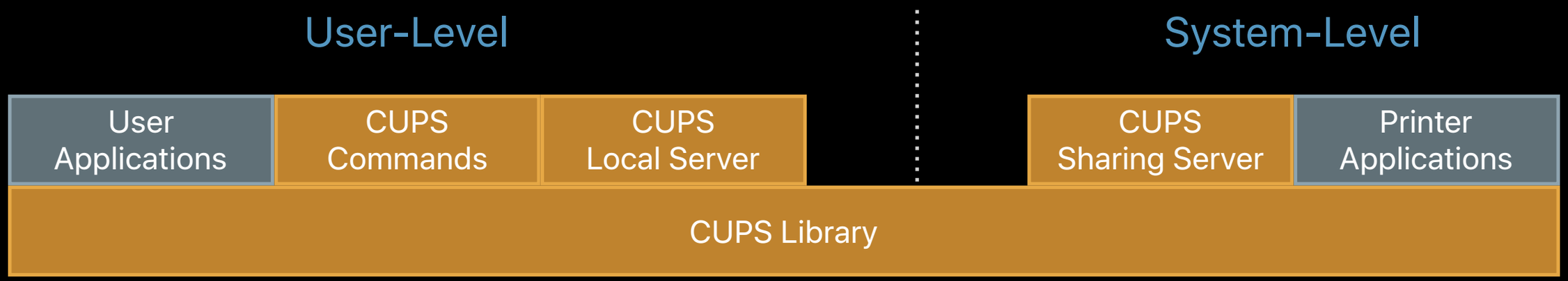
# CUPS Future

*Simplification*

- The current architecture has many moving parts and a lot of legacy interfaces, protocols, and file formats to support
  - Client-centric and server-centric features are mixed together - not optimal for either usage
  - The current printer driver architecture enforces a "lowest common denominator" approach for printing features
- Clearly, we need to make major changes to the printing system

# Simplification

*Modular Printing System*



# Modular Printing System

*CUPS Libraries*

- libcups, libcupsimage, and associated headers
- `cupsRaster` functions included in both libcups and libcupsimage
- libcupscgi, libcupsmime, and libcupsppdc libraries no longer supplied
  - Components that used them dropped in new modular printing system
- Just as in current CUPS releases, the same libraries are used for both user and system level processes

# Modular Printing System

*User Applications*

- *Developed and distributed separately from CUPS*
- Use CUPS library directly, or indirectly through GUI toolkits to provide printing functionality
- Aside from any legacy PPD-based UI, really no change from today

# Modular Printing System

## *CUPS Commands*

- cancel, lp, lpmove, lpoptions, lpq, lpr, lprm, and lpstat commands
- Communicate with the CUPS Local Server by default
  - But can be pointed directly at a CUPS Sharing Server
- Guidance would be to package them separately for platforms/environments that lack a command-line interface

# Modular Printing System

*CUPS Local Server*

- Runs in the user session:
  - Accessed through user-specific domain socket, run on demand
  - Has access to credentials needed for printing
- Provides basic print conversion or pass-through of plain text, JPEG, and PDF files
- No persistent print queues
  - Bonjour + LDAP + configuration profiles populate list of printers, temporary queues created on demand

# Modular Printing System

## *CUPS Sharing Server*

- System service that runs as "lp" or some other system role account
- Bonjour + LDAP support for sharing to the client/local spoolers
- Authentication, authorization, and access control using PAM and/or OAuth 2.0
- Reliable accounting, quotas/limits, print policies (duplex only, B&W only, etc.)
- Multiple job/document queuing on high speed/release printers

# Modular Printing System

*CUPS Sharing Server, con't*

- Shared Infrastructures Extensions support for release/ indirect printing
- Administrative commands
  - cupsaccept, cupsdisable, cupsenable, cupsreject, lpadmin, and lpc
- Web interface

# Modular Printing System

## *Printer Applications*

- *Developed and distributed separately from CUPS*
- Can run at user and/or system level (as needed)
- Recommended design pattern is to create a dedicated service or application with tight integration, e.g., ippserver + Gutenprint combination
  - An advantage of this approach is that drivers like Gutenprint will no longer be shackled by the limitations of the CUPS driver interface and can actually provide configuration and status UI that makes sense for that software

# Modular Printing System

*Printer Applications, con't*

- Current ippserver implementation shows how to provide generic "drivers" for legacy printers
  - HP PCL and PostScript printers can often be queried for supported media and trays...
- Another possible backwards-compatibility implementation would be a service based on current CUPS that implements the filter chain and provides JPEG and PDF to PostScript/Raster conversion
  - LD_PRELOAD could be used to insert working deprecated functions (`ppdOpen`, etc.)
  - ... but still have the same limitations of CUPS 2.x

# CUPS Plenary

*Q&A*

Questions?

# CUPS Plenary

*Resources*

- CUPS Web Site
  - https://www.cups.org/
- CUPS Repository
  - https://github.com/apple/cups
- CUPS Programming Manual
  - https://www.cups.org/doc/cupspm.html
  - https://www.cups.org/doc/cupspm.epub