



# Automatic Download/Installation of Distribution-Independent Printer Driver Packages

**Till Kamppeter**

OpenPrinting Manager, The Linux Foundation



- **Why auto-download of distro-independent driver packages?**
  - Distributions do not ship all available printer drivers
  - Free drivers from upstream need to be compiled by users -> driver installation too complicated for unexperienced users
  - Manufacturers make packages only for a few major distributions
  - Driver packages often difficult to find on manufacturer's web sites
  - Testing/packaging effort for manufacturers and driver developers too high to ship binary driver packages for all distributions
- **Existing Infrastructure we make use of**
  - **OpenPrinting database** (former linuxprinting.org), central database for printer/driver info
  - **LSB** provides tools and infrastructure to create **distribution-independent binary packages**



- **Solution**

- **Distribution-independent printer driver packages**

- Based on **LSB 4.1** for binary format
- Using **CUPS**, **Ghostscript** (with IJS, CUPS Raster and OpenPrinting Vector interfaces), **Perl**, and **foomatic-rip** which is on every distribution
- **LSB DDK** (Driver Development Kit) helps packaging the drivers correctly
- Make packages part of **OpenPrinting database** (or link them at least from there), so that they can be easily found
- Infrastructure for automatic package lookup, download, installation, and auto update through the internet by printer setup tools and package managers
- **system-config-printer** (Fedora/Red Hat, Ubuntu, Mandriva) already supports automatic download of driver packages (with Jockey)



- **Distribution-independent**
  - One package for Linux, instead of one for Red Hat, one for SuSE, one for Ubuntu, ...
- **Binary packages**
  - User does not need to compile, system is also suitable for closed-source drivers
- **Same installation method for all driver packages**
  - A printer setup tool can easily install them automatically
- **One query location at the OpenPrinting web site**
  - Easy to find for both humans and printer setup tools
  - Granting redistribution permissions of non-free drivers is much easier.



- **Driver query API for printer setup tools**
  - All needed info available: License, supplier, support contact, print quality indices. So the setup tool and the user can easily find the driver suiting best for him.
- **Distributions look up drivers at OpenPrinting**
  - Distributions do not need to support all printer models
  - So drivers newer than the distro are available, for updates and for new printer models
  - Distribution CDs do not get overloaded with printer drivers and PPD files



- **Manufacturers have to take full responsibility on their drivers**
  - Distributions are supposed to download these non-distro packages by default
  - Users would make distros responsible if something goes wrong
  - Manufacturers should sign a legal agreement to take responsibility
- **Cryptographic code in drivers and export restrictions**
  - Use only standardized cryptographic technologies which come already with the OS
  - Host the driver packages on the manufacturer's site and link only from OpenPrinting
    - Repository on manufacturer's site must be indexed for RPM and DEB (for automatic updates)
    - Repository linked from OpenPrinting web site to allow same look-up and download mechanism as for directly hosted drivers
    - Links on OpenPrinting web site have to be kept up-to-date



# What is still needed?

- **Signing**
  - Packages uploaded by manufacturers must be electronically signed
- **Repositories handled like at distros**
  - **main**: Drivers of trusted sources (usually manufacturers) who have signed responsibility agreement go here, only from this repository distributions automatically download and install by default (like “main” in the distros)
  - **contrib**: Upload to here does not require signing the agreement, but to automatically download from here the user has to activate this repository (like “contrib”, “universe”, ...) in the distros



- **Maintainer scripts: Only pre-defined procedures**
  - **pre/post-install/uninstall scripts**
  - To avoid arbitrary system changes by printer driver packages
  - Procedures pre-defined as macros in the LSB DDK
    - Add /opt/<supplier>/... to \$PATH
    - Symlink CUPS backends, filters, filter rules, and PPDs installed in /opt to appropriate system directories
    - Update PPDs of existing queues for this driver
    - Set up, start, and restart driver-specific daemons
    - Restart CUPS
    - Clean up all of the above when uninstalling



- **New features in the LSB**

- **SANE** (Scanner support) for multi-function devices (and also to cover scanners with the LSB DDK)
- **D-Bus**: Allows background processes (CUPS filters or backends) to interact with the user, for example by popping up GUI elements
  - HP's HPLIP does it for fax queues, to get banner pages and fax number
- **U-Dev**: Actions on auto-detecting devices, like permission settings, group ownerships, firmware upload



- **General info**

- <http://www.openprinting.org/>

- **For developers**

- <http://www.linux-foundation.org/en/OpenPrinting/Development>

- <http://www.linux-foundation.org/en/Developers>

- **For driver developers**

- <http://www.linux-foundation.org/en/OpenPrinting/WritingAndPackagingPrinterDrivers>

- **For developers of printer setup tools**

- <http://www.linux-foundation.org/en/OpenPrinting/Database/Query>

- **Available driver packages**

- [http://www.openprinting.org/driver\\_list.cgi](http://www.openprinting.org/driver_list.cgi)

- **How to install driver packages**

- <http://www.linux-foundation.org/en/OpenPrinting/Database/DriverPackages>



- **Components**

- **Renderer/Filter:** Converts device-independent data format from application (PDF/PostScript) into printer-specific format. Your driver is a part of this component. The PDF/Postscript renderer Ghostscript is the other important part.

Driver types:

- CUPS Raster
  - IJS
  - OpenPrinting Vector
  - *Do not create built-in drivers for Ghostscript!!*
- **Low-Level communication with the printer:** The CUPS backends. Usually it is no problem to communicate with a printer using the standard backends. *CUPS backends do not convert incoming data to the printer's language!* You only need to ship custom backend in these cases:
    - Bi-directional features like ink level monitoring, scanning, fax, ...
    - Communication problem due to hardware bug
    - Completely non-standard communication protocol



- **Maintenance and auxiliary tools:** Programs for checking ink levels, cleaning print heads, managing fax numbers and incoming faxes, ... Should have GUI for desktop users, but also text mode for scripts and blind users.
- **Scanner Drivers:** For the scanning part of multi-function devices. Should not be GUI applications but SANE drivers, so that all scanning applications can scan.
- **Available infrastructure and integration**
  - CUPS
    - Standard printing system under Linux
    - Schedules, routes, controls jobs on the network and finally filters and prints them
    - Filters for PDF/PostScript → printer's language conversion
    - Backends for low-level communication
  - Ghostscript
    - PostScript/PDF renderer/rasterizer
    - Built-in printer drivers, CUPS raster output, standard image formats (to pipe into separate driver executables), IJS and OpenPrinting Vector plugin interfaces
    - Called by CUPS filter: pstoraster, pdforaster, foomatic-rip, driver-supplied filter
    - Can be replaced by Poppler in certain cases with PDF input



- foomatic-rip
  - Universal print filter
  - For all printing systems (CUPS, LPD, LP, LPRng, ...)
  - Turns PDF/PostScript to the printer's language
  - Allows arbitrary Ghostscript/filter/wrapper command lines
  - Extra functions: Page overrides, CUPS Raster drivers with non-CUPS spooler
- CUPS-DDK
  - Tools for manipulation of PPD files: Generation from simple, compact .drv files, joining single-language PPDs to one multi-language PPD, ...
  - Universal PCL and ESC/P-2 drivers which can be used as a base for your own driver
  - Integral part of CUPS 1.4.x, so it ships with newer distributions
- SANE
  - Standard scanning environment: Make a SANE driver and users can scan with the software they are used to.
  - Not yet in LSB, but SANE drivers compiled in the LSB build environment seem to work with all distros
  - Planned for being included in LSB 4.1



## ➤ LSB

- Standards to allow creating distribution-independent binary application packages.
- Printing requirements added in LSB 3.2, printing requirements did not change in 4.0.
- Required are the CUPS convenience API, the driver interfaces IJS, OpenPrinting Vector and CUPS raster, Ghostscript, Qt 4.0 printing interfaces, PPDs being searched in `/usr/share/ppd`.
- SANE and full CUPS API still missing, planned for LSB 4.1.

## ➤ OpenPrinting Web Site

- Which printer works with which driver?
- More than 3000 printers
- Can be looked up via browser or by printer setup tools (via web API)
- If you post an entry for your LSB-packaged driver this makes the printer for the users of many distributions “just work”.



- **What to do and what not to do**

- Note that Linux and Windows are well different
  - Normal users and the administrator (“root”)
  - CUPS printing environment, PPD files
  - Drivers are executed on the server and cannot interact with client GUI
  - One package can never overwrite or modify files of another
- Setup of a print queue must work with the distribution's printer setup tool
  - No setup tools coming with driver are required (to do some special steps)
- Printing itself must work with system's software
  - All options of the driver reachable via the PPD file
  - No overwriting of system tools (like “lpr”) with driver GUI
- Provide 1 PPD for each printer model with distinct IEEE-1284 device ID
  - No “LaserStar XXX 100/200/300” PPD if the three printers have different IDs
- Physical PPDs must be present in the package
  - CUPS PPD generating mechanism (/usr/lib/cups/driver, .drv file) not yet in LSB
  - No other on-the-fly-PPD-generation supported by CUPS
  - “lpinfo -m” must list your PPDs



# Printer Driver Design

- Do not require from the user to run command line tools
  - Users rely on standard procedure (install package, set up print queue)
  - User's do not necessarily read README files and Release Notes
  - Fully non-interactive command line programs can be run by pre/post-install scripts of the package
- Do not require the printer to be connected while installing your package
  - Package can get installed independent of the printer, for example during OS installation, by sys admin via SSH from remote machine, ...
  - Auto-download expects at first to install the package, then to set up a queue
- Do not modify the system's configuration, system files, permissions
  - You break other things to get your printer to work
  - These changes get overwritten on system updates
  - Examples: Command line tool “lpr” replaced by GUI breaks scripts, desktop app made running SUID root breaks security
- Avoid the need of specialized kernel modules
  - Instabilities/bugs can crash whole system
  - Kernel modules cannot get LSB-packaged to work on all distributions



- Avoid world-writable files
  - Big security hole
  - Everyone can modify them, and change printing behavior for other users
- Allow flexibility to where to install files
  - File locations for driver shipped by distro are in /usr/..., for drivers shipped by you directly to end user /opt/<supplier/...
  - Do not hard-code file locations into closed-source binaries, use configuration files
- Do not allow normal users to change system-wide defaults
  - If user saves settings, change should only be done for this user, to avoid unexpected results for other users.
- Avoid file conflicts between your driver packages
  - It should be possible to connect any pair of your printers with the same machine
  - No file with a given name and path should exist in more than one package
- Do not create too many too small packages
  - Difficult to maintain and to make available for the user
  - If you provide 100 printers, they do not have more than 10 completely different languages



- Make sure that your printer is uniquely identified by its IEEE-1284 device ID
  - Your custom backends should identify printers by device ID and never by `/dev/usb/lp0`, otherwise there are problems on systems with more than one USB printer
  - Device ID should contain serial number and custom backends should use it to allow more than one printer of the same model on the USB
  - Also do not use USB vendor and product IDs, as there are some completely different printers with the same vendor and product ID.
- **Support the Common Printing Dialog with your PPDs**
  - Assign tags to options so that they appear under these tags (“\*OOptionTag” keyword). More than one tag per option is allowed
  - Make Globalized (multi-language) PPDs as described in CUPS documentation
  - Use CUPS custom options for numerical, string, and password options
  - Use widget hints (“\*OOptionHints”) to get the desired widgets for your options
  - Create quick presets (like “Photo”, “Office Document”, “Quick Draft”, ..., keyword “\*APPrinterPreset”). This sets several options with one click.
  - Assign icons to options and choices



- **LSB DDK is LSB SDK with some additional stuff**
- **Install LSB SDK:**
  - Packages: lsb-build-base, lsb-build-cc, lsb-build-c++, lsb-appchk, lsb-pkgchk, lsb-build-libbat
  - If your package contains auxiliary programs with GUI, install also the appropriate lsb-build-... packages for X and desktop/GUI support
- **Install DDK files**
  - RPM macro set, append it to your ~/.rpmmacros or to /etc/rpmrc
  - LSB package of SANE if you want to make a driver package for multi-function device with scanner
- **Set up your system for building RPMs**
  - Install rpmbuild
  - Set fields in ~/.rpmmacros:

%_topdir	/home/YOURLOGIN/rpm
%_tmppath	/home/YOURLOGIN/tmp
%distribution	LSB
%vendor	YOURORGANIZATION
%packager	YOURNAME <YOURMAILADDR>



# Compiling the Driver to be LSB-Compliant

- All LSB-packages are RPMs, for Debian-based distributions the LSB requires “alien”
- We already convert the RPMs to Debian packages to support automatic updates with apt
- Our RPMs have following differences relative to normal RPMs:
  - Install to /opt/<supplier>. Take care that your packages do not conflict with each other
  - Files which needs to be found in system directories (PPDs, backends) are symlinked by the maintainer scripts (pre-/post-(un)install)
  - The maintainer scripts work in both RPM and Debian packages and with all distributions
  - PPD files are found in /usr/share/ppd/
  - Package contains at least PPD files
- The RPM macro set helps you to get easily RPMS in this form
- Also have a look at the spec files of already existing packages
- Building in LSB Build Environment chroot helps to obtain LSB binaries
- Check binaries for LSB compliance with lsb-appchk



# Compiling the Driver to be LSB-Compliant

- Use a package name only of lower-case letters, numbers, and dashes, no upper-case, nor underscores, to work with both RPM and Debian
- Create your spec file using the macro set
  - Set human-readable strings for PPD nicknames (in PPD header)  
`%define driverstr <Driver Name>`  
`%define supplierstr <Supplier Name>`
  - Let everything go into `/opt/<supplier name>` (in RPM header):  
`%define supplier <supplier name>`  
`%define drivername <driver name>`  
`%install_into_opt`
  - Set executable and man paths so that the executables and man pages in `/opt` get found (in RPM header)  
`%has_bin_executables`  
`%has_sbin_executables`  
`%has_manpages`  
`%set_opt_paths`
  - Use special macros for building and installing the package (sections `%build` and `%install`)  
`%configure`  
`%make`  
`%makeinstall`



# Compiling the Driver to be LSB-Compliant

- Generate PPDs from Foomatic XMLs (after “make”):  
**%build\_foomatic\_ppds**
- Put absolute paths to filter calls in the PPDs and arrange the PPDs according to LSB 3.2 (after “make install”):  
**%adjust\_ppds**
- Maintainer script macros (installation)

**%pre**

**%init\_scriptlet** (source /etc/profile and /etc/profile.d/\*)

**%create\_opt\_dirs** (create directories in /opt)

**%post**

**%init\_scriptlet**

**%set\_opt\_paths** (set \$PATH and \$MANPATH to support the files in /opt)

**%set\_ppd\_links** (set links so that PPDs are found in /usr/share/ppd)

**%set\_cups\_links** (set links to make CUPS backends and mime rules  
getting found)

**%update\_ppds\_fast** (update PPDs of already existing queues when  
updating this package)

**%restart\_cups** (restart the CUPS daemon)



# Compiling the Driver to be LSB-Compliant

- Maintainer script macros (Uninstallation)

```
%preun  
%init_scriptlet  
%not_on_rpm_update (do the following only if we do not update an RPM)  
%remove_opt_paths (Remove changes on $PATH and $MANPATH)  
%end_not_on_rpm_update  
  
%postun  
%init_scriptlet  
%not_on_rpm_update  
%remove_ppd_links (Remove links to /usr/share/ppd)  
%remove_cups_links (Remove links for CUPS backends and mime rules)  
%restart_cups  
%end_not_on_rpm_update
```



# Testing Your Package

- Test the LSB compliance of the executables with **lsbappchk <name of the executable with full path>**  
Link libraries statically if they are not covered by the LSB. Do not forget any executable. They are typically in
  - /opt/<supplier>/bin**
  - /opt/<supplier>/sbin**
  - /opt/<supplier>/lib**
  - /opt/<supplier>/cups/lib/filter**
  - /opt/<supplier>/cups/lib/backend**
- Check Adobe-compliance of PPD files with **cupstestppd <name of the PPD file with full path>**
- Install the driver package. Should work like described in the user instructions
- Try to set up print queues and to print



# Making Your Packages Auto-Downloadable

- To keep responsibility on drivers at the manufacturer/supplier and not at the Linux Foundation he has to proceed as follows:
  - The driver packages are hosted on the manufacturers/suppliers site
  - Manufacturer/supplier has to sign packages and upload the key fingerprint to an <https://> web site with an SSL certificate signed by an official registrar.
  - OpenPrinting hosts a database entry with info about driver properties, supported printers and links to the driver packages and key fingerprints
  - Printer setup tools look up the drivers through OpenPrinting and then download through the links, getting package from manufacturer/supplier. They also verify the signatures.
- The manufacturer/supplier provides the packages as RPM and Debian packages for 32-bit and 64-bit systems
- The repositories have to get indexed for distribution's package managers and automatic updates



# Creating Package Repositories

- You need an 64-bit machine with 64-bit Linux installed (amd64/x86\_64)
- The following packages must be installed: rpm (rpmbuild), dpkg, alien, fakeroot, linux32 (util-linux), ia32-libs, createrepo, apt-utils (apt-ftparchive)
- The packages are all available in Ubuntu Universe, with other distributions names can change or software not available as package
- All packages can be generated on one machine, independent whether it runs a Debian- or RPM-based distribution
- Create RPMs as shown above
- Create RPM repository like:

```
  |-- RPMS
  |  |-- i486
  |  |  |-- yourdriver-1.0.0-1lsb3.2.i486.rpm
  |  |-- x86_64
  |  |  |-- yourdriver-1.0.0-1lsb3.2.x86_64.rpm
  |-- noarch
  |  |-- yourpostscriptprinters-1.2.0-1lsb3.2.noarch.rpm
```

- Index the RPM repository with  
**createrepo RPMS**



# Creating Package Repositories

- Setting up a Debian repository is rather complicated, so use our script “mkreposeb”. Run

**mkreposeb <Binary RPM package file>**

On each binary RPM, and it gets converted into a Debian package (with maintainer scripts) added to the Debian package repository and the repository gets re-indexed.

- Make sure that when your repositories are on your web server, that the directories which contain the \*.rpm and \*.deb files are browsable for the client user, otherwise OpenPrinting cannot resolve your packages through links with wildcards.
- Sign your packages/repositories following our instructions. Register the key and make it available



# Signing Packages/Repositories

- **Generate a GPG key**

- `gpg --gen-key`
- Handle the private key with care, do not forget your password

- **Build a trusted path to the distributions**

- Upload your key fingerprint to an `https://` site with an SSL certificate which has been signed by an official registrar

- **Register the GPG key in the keyserver network**

- `gpg --keyserver certserver.pgp.com --send-key you@example.com`

- **Sign your uploaded packages**

- RPM

- Sign files: `rpm --addsign <package file> ...`

- Debian

- Sign archives: `.deb` files listed in `Packages.gz`, all `Packages.gz` listed in `Release`, sign only the `Release` files, put signature in `Release.gpg`.
- Example: <http://archive.ubuntu.com/ubuntu/dists/lucid/>



# Registering your driver at OpenPrinting

- Create a Foomatic driver XML file for your driver, containing
  - Driver name
  - Short description
  - License
  - License text if non-free
  - Support contacts
  - Performance indices (optional)
  - Comments (optional)
  - List of supported printers
  - Links to driver packages
- Create a Foomatic printer XML file for each supported printer, containing
  - Manufacturer/Model
  - IEEE-1284 Device ID
  - Recommended driver
  - Functionality level
  - Comments (optional)



# Registering your Driver at OpenPrinting

- E-Mail the XML files to us
- We review your XML files and packages
- If all is OK, we import the XML files into the database
- Once imported, your files are auto-downloadable but also visible for who browses the OpenPrinting web site
- At any time, you can replace your packages by newer versions, and they stay auto-downloadable and users who already downloaded get the new version with their distribution's daily update.
- Only do not move your repository, do not rename the packages, and do not forget to re-index and re-sign after each change.



# More Participants needed

- **Driver suppliers**

- Epson (Binary driver packages)
- Ricoh (PPD files)
- Lexmark (PPD files)
- PPD files from manufacturers get always packaged, but most manufacturers did not supply new PPD files for a longer time

- **Linux distributions**

- Ubuntu

- **We need more participants**

- The more distributions participate, the more manufacturers get motivated to package drivers and vice versa.



- The full tutorial text
  - <http://www.openprinting.org/collaborate/workgroups/openprinting/writingandpackagingprinterdrivers>
- How to install distribution-independent driver packages
  - <http://www.openprinting.org/collaborate/workgroups/openprinting/database/driverpackages>
- General info
  - <http://www.openprinting.org/>