# The system-config-printer CUPS administration tool

*Tim Waugh  (twaugh@redhat.com), September 2007*

## *Introduction*

This document aims to give a brief introduction to the CUPS administration tool that was written for Fedora Core 6, system-config-printer, and the current state of this tool's development.

Although it was written for Fedora it is available in Debian and Ubuntu as well.

In its architecture, the tool takes the form of a GTK+ application written in Python.  It uses the libcups application programming interface (API) provided by CUPS, and it is able to do this with the use of Python bindings for libcups.  The package providing these bindings, pycups, was written as part of the effort to create the CUPS administration tool.

The libcups API provided by CUPS allows applications to interact with a CUPS server, and the implementation uses the IPP protocol to do so.  In many cases the CUPS server will be on the local machine but this need not necessarily be the case.  Due to the fact that system-config-printer uses libcups for all interactions with the CUPS server rather than changing configuration files in the local file system, it is well suited to configuring remote, possibly headless, CUPS servers.

This approach, of using the libcups API for all interactions, is also the one used by the CUPS web configuration interface.  Indeed, the idea of system-config-printer is to provide similar functionality to the CUPS web interface but with native GTK+ widgets and greater desktop integration.

## *Inventory of parts*

There are three source packages relevant here.  They are: pycups, system-config-printer, and hal-cups-utils.

The pycups CUPS bindings for Python are distributed separately from the graphical elements of the configuration tool.  This is because they may be useful for all sorts of other projects.  The web page for pycups can be found at http://cyberelk.net/tim/pycups and new releases are announced on freshmeat.net.  It is currently at version 1.9.26.

The graphical administration tool is contained in the system-config-printer source package, the web page for which is at http://cyberelk.net/tim/system-config-printer – again, new releases of this are announced on freshmeat.net.  It is currently at version 0.7.75, with a stable branch 0.7.74.x (which Fedora 8 will use).

Within the system-config-printer source package can be found the administration tool and, in addition, a job management applet (“system-config-printer-applet” for want of a better name).  This applet runs when the desktop session starts, displaying a printer icon in the notification area whenever the user has print jobs queued.  Clicking on this icon shows the queued jobs, and allows them to be cancelled, paused, resumed or (for completed jobs) re-printed.

This job management applet can also run as a normal application without there being an icon in the notification area.  Desktop files are provided both for auto-start (with notification icon) and from-menu launch (without notification icon).

Lastly, the hal-cups-utils source package provides some hooks into the Hardware Abstraction Layer (hal.freedesktop.org) to allow automatic queue set-up for locally connected printers.  This package is not (yet?) distributed separately from the source RPM in Fedora.

## *The pycups CUPS bindings for Python*

Originally the purpose of writing CUPS bindings for Python was in order to implement system-config-printer in Python. Although the configuration tool project was a complete re-design of an existing tool, Python is a very good language for rapid prototyping and it also allowed some small useful sections of the previous tool (such as the SMB printer browsing) to be re-used. The Python bindings have since proved useful as a diagnostic tool – when a user describes a bug it is simple to ask them to run a command line like “python -c 'import cups; ...'” and get them report back the results in order to pin-point the problem.

The idea behind pycups was to get system-config-printer up and running. It was intended to be a fairly thin layer of bindings to the libcups API. In practice there are some convenience functions included so as to keep the configuration tool's code a little simpler, so as a result there is not a one-to-one correlation between the libcups API and that of pycups. The Python doc-strings in pycups are now in epydoc-format, and so “make doc” creates HTML documentation of the API.

Here, then, is an example of pycups being used from an interactive Python session:

```
>>> import cups
>>> cups.setUser('root')
>>> cups.setServer('localhost')
>>> c=cups.Connection()
```

Now `c` represents a CUPS server connection. It is a Python object with methods. Here is how we can get a list of the printers on that server (long lines have been truncated to fit on this page):

```
>>> from pprint import pprint
>>> pprint(c.getPrinters())
{'psc': {'device-uri': 'hp:/usb/PSC_2200_Series?serial=MY2XXXXXCR0G',
         'printer-info': 'HP PSC 2210',
         'printer-is-shared': True,
         'printer-location': "On Sue's right",
         'printer-make-and-model': 'HP PSC 2210 Foomatic/hpijs (recommended)',
         'printer-state': 3,
         'printer-state-message': '',
         'printer-state-reasons': ['none'],
         'printer-type': 36892,
         'printer-uri-supported': 'ipp://localhost:631/printers/psc'},
 'stylus': {'device-uri': 'usb://EPSON/Stylus%20D78',
            'printer-info': 'Epson Stylus D78',
            'printer-is-shared': True,
            'printer-location': 'Between the monitors',
            'printer-make-and-model': 'Epson Stylus D68 Foomatic/gutenprint-ijs-simp…',
            'printer-state': 3,
            'printer-state-message': '',
            'printer-state-reasons': ['none'],
            'printer-type': 167948,
            'printer-uri-supported': 'ipp://localhost:631/printers/stylus'}}
```

We can also get a list of the IPP attributes for a particular printer (this is trimmed quite heavily for size):

```
>>> pprint(c.getPrinterAttributes('stylus'))
{'charset-configured': 'utf-8',
 'charset-supported': ['us-ascii', 'utf-8'],
 'color-supported': True,
 'compression-supported': ['none', 'gzip'],
 'copies-default': 1,
 'copies-supported': (1, 100),
 'device-uri': 'usb://EPSON/Stylus%20D78',
 'document-format-default': 'application/octet-stream',
…
 'job-hold-until-default': 'no-hold',
 'job-hold-until-supported': ['no-hold',
                              'indefinite',
                              'day-time',
                              'evening',
                              'night',
                             'second-shift',
                             'third-shift',
                             'weekend'],
…
 'job-sheets-default': ('none', 'none'),
 'job-sheets-supported': ['none',
                         'classified',
                         'confidential',
                         'mls',
                         'secret',
                         'selinux',
                         'standard',
                         'te',
                         'topsecret',
                         'unclassified'],
…
 'number-up-default': 1,
 'number-up-supported': [1, 2, 4, 6, 9, 16],
…
 'printer-error-policy': 'stop-printer',
 'printer-error-policy-supported': ['abort-job', 'retry-job', 'stop-printer'],
…
 'printer-is-accepting-jobs': True,
 'printer-op-policy': 'default',
 'printer-op-policy-supported': ['default'],
…
 'uri-authentication-supported': ['requesting-user-name'],
 'uri-security-supported': ['none']}
```

Using pycups we can set default options, add queues, manipulate classes, adjust the CUPS server settings (as with the cupsctl command: another libcups application), print test pages – anything that the CUPS web interface can do, more or less.

Another part of the libcups API provides support for handling PPD files. Here is a demonstration of that part of the API being used by pycups:

```
>>> print c.getPPD('psc')
/tmp/46e68bbfce80a
>>> ppd = cups.PPD('/tmp/46e68bbfce80a')
>>> pprint(dir(ppd))
['__class__',
 …
 '__str__',
 'attributes',
 'conflicts',
 'constraints',
 'findAttr',
 'findNextAttr',
 'findOption',
 'localize',
 'markDefaults',
 'markOption',
 'nondefaultsMarked',
 'optionGroups',
 'writeFd']
>>> for g in ppd.optionGroups:
...     print g.name
...     for o in g.options:
...             print " ", o.keyword, ":", o.text
...
General
  PageSize : Page Size
  PageRegion : PageRegion
  PrintoutMode : Printout Mode
  InputSlot : Media Source
  Duplex : Double-Sided Printing
PrintoutMode
  Quality : Resolution, Quality, Ink Type, Media Type
```

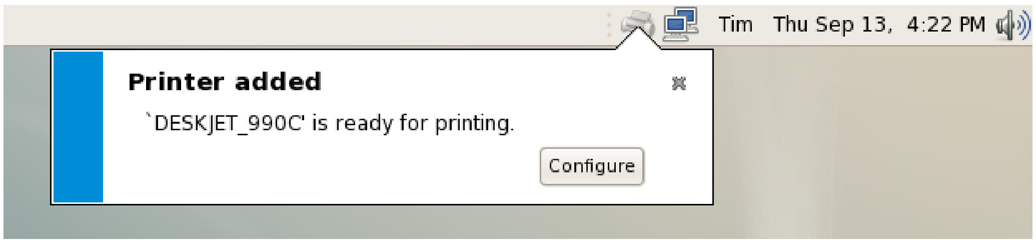
## *The CUPS administration tool*

The best case scenario for a user plugging in a USB printer is that the computer configures it automatically, and there is nothing further to do. In some cases it is possible for us to do that. When the printer is connected, HAL spots the new device and calls into a program in the hal-cups-package called hal_lpadmin.

This program shares some of the system-config-printer code, notably the class for indexing PPD files available in CUPS and matching devices to drivers, and does its best to configure a queue for the printer. The matching is performed using IEEE 1284 Device ID strings: first an exact make and model match is sought, then a fuzzy match, and finally the printer command languages are examined in order to try a generic driver. It will always configure a queue, but in the worst case it will be a text-only queue.

Once the queue is configured, hal_lpadmin makes a D-Bus call on the system bus to announce the new queue's name, and to indicate how good a match the driver is.

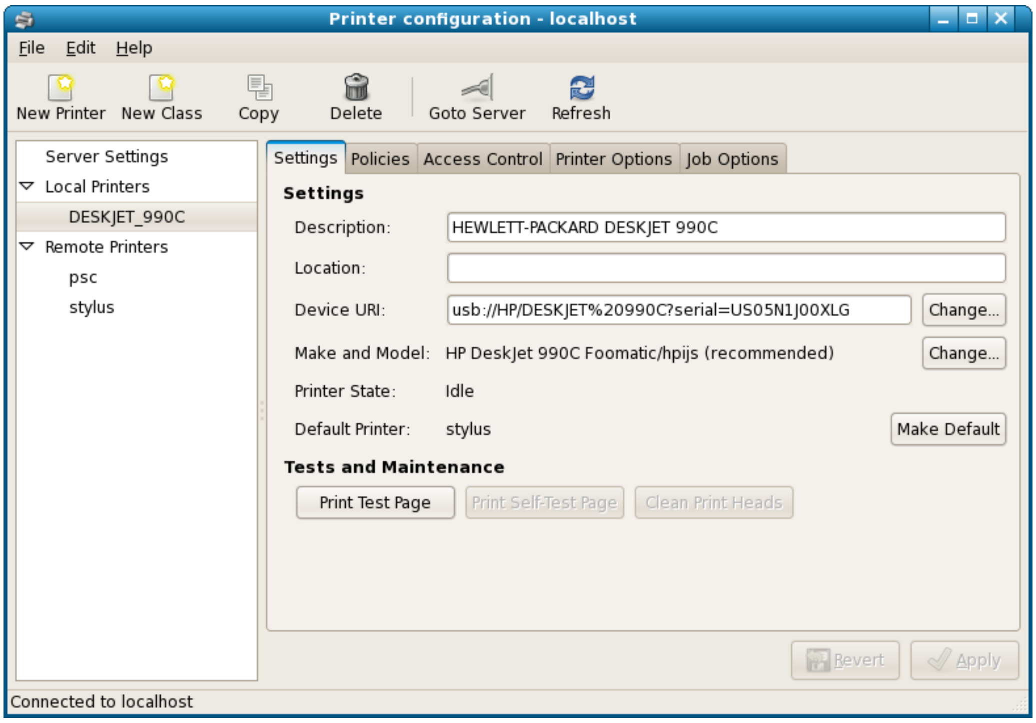
If a user is logged in at the console, the job management applet is waiting for just such a message. When it receives it, a notification bubble is displayed on the screen showing the name of the new queue and offering a chance to perform further configuration if desired.



**Printer added**

`DESKJET_990C' is ready for printing.

Clicking on the "Configure..." button launches system-config-printer, with the relevant printer already selected.


Printer configuration - localhost
File Edit Help
New Printer New Class Copy Delete Goto Server Refresh
Server Settings
Local Printers
DESKJET_990C
Remote Printers
psc
stylus
Settings Policies Access Control Printer Options Job Options
Change...
Change...
Make Default
Print Test Page Print Self-Test Page Clean Print Heads
Revert Apply
**Settings**

Description: HEWLETT-PACKARD DESKJET 990C

Location:

Device URI: usb://HP/DESKJET%20990C?serial=US05N1J00XLG

Make and Model: HP DeskJet 990C Foomatic/hpijs (recommended)

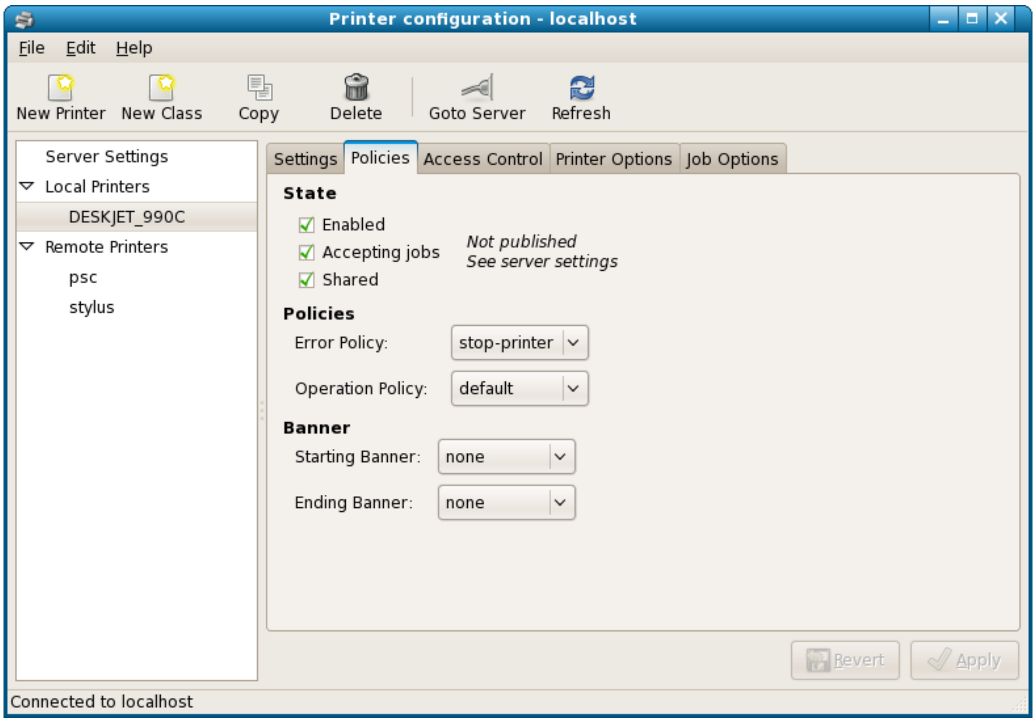
Printer State: Idle

Default Printer: stylus

**Tests and Maintenance**

Connected to localhost

The tabs at the top of the pane show the different properties of the queue that can be modified, including whether it is shared, who may print to it, the PPD options, and default job options.


Printer configuration - localhost
File Edit Help
New Printer New Class Copy Delete Goto Server Refresh
Server Settings
Local Printers
DESKJET_990C
Remote Printers
psc
stylus
Settings Policies Access Control Printer Options Job Options
Revert Apply
Connected to localhost

**State**

✓ Enabled
✓ Accepting jobs
✓ Shared

*Not published*
*See server settings*

**Policies**

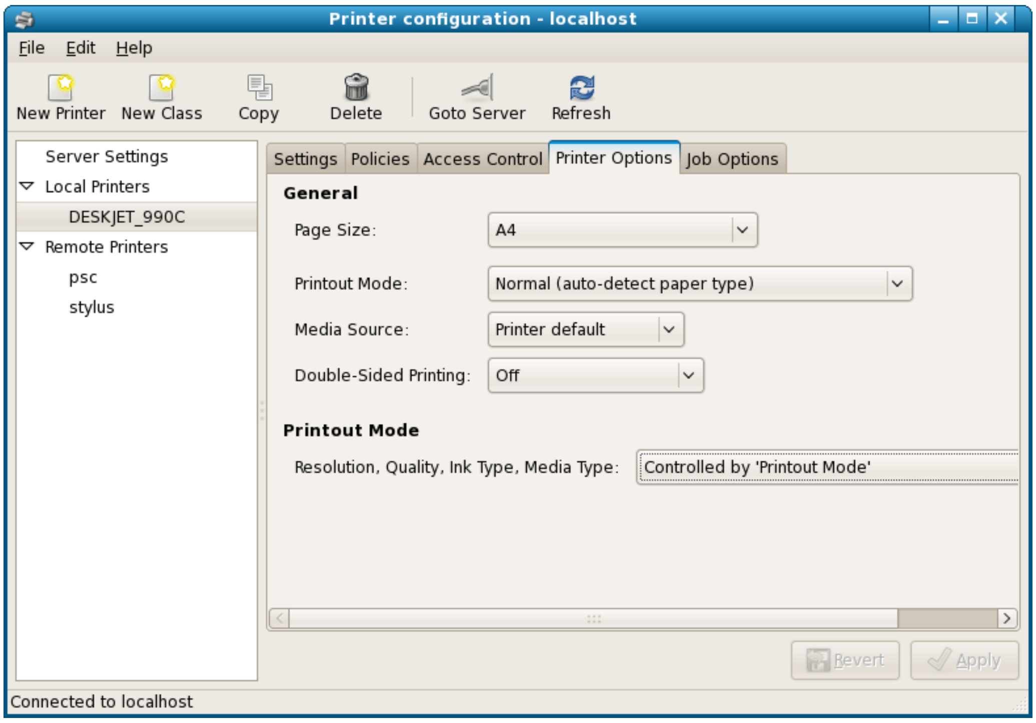
Error Policy: stop-printer

Operation Policy: default

**Banner**

Starting Banner: none

Ending Banner: none


Printer configuration - localhost
File Edit Help
New Printer New Class Copy Delete Goto Server Refresh
Server Settings
Local Printers
DESKJET_990C
Remote Printers
psc
stylus
Settings Policies Access Control Printer Options Job Options
Revert Apply
Connected to localhost

**General**

Page Size: A4

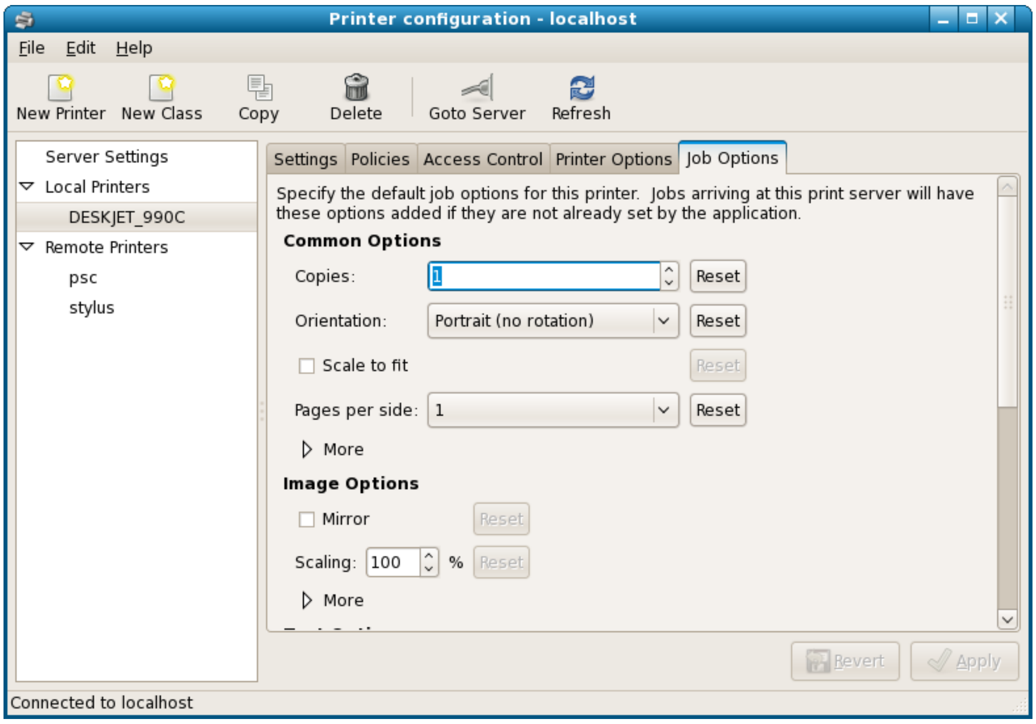
Printout Mode: Normal (auto-detect paper type)

Media Source: Printer default

Double-Sided Printing: Off

**Printout Mode**

Resolution, Quality, Ink Type, Media Type: Controlled by 'Printout Mode'



Specify the default job options for this printer. Jobs arriving at this print server will have these options added if they are not already set by the application.

**Common Options**

Copies: 1 Reset

Orientation: Portrait (no rotation) Reset

☐ Scale to fit Reset

Pages per side: 1 Reset

▷ More

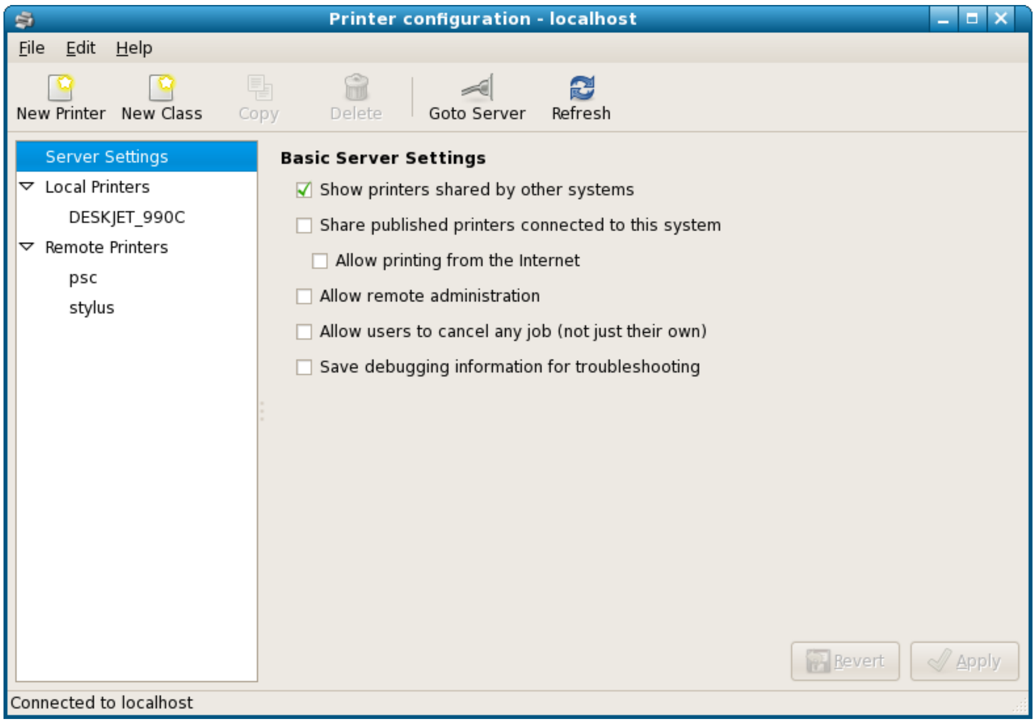
**Image Options**

☐ Mirror Reset

Scaling: 100 % Reset

▷ More

The server settings can be adjusted by selecting “Server Settings” in the list.



**Basic Server Settings**

☑ Show printers shared by other systems

☐ Share published printers connected to this system

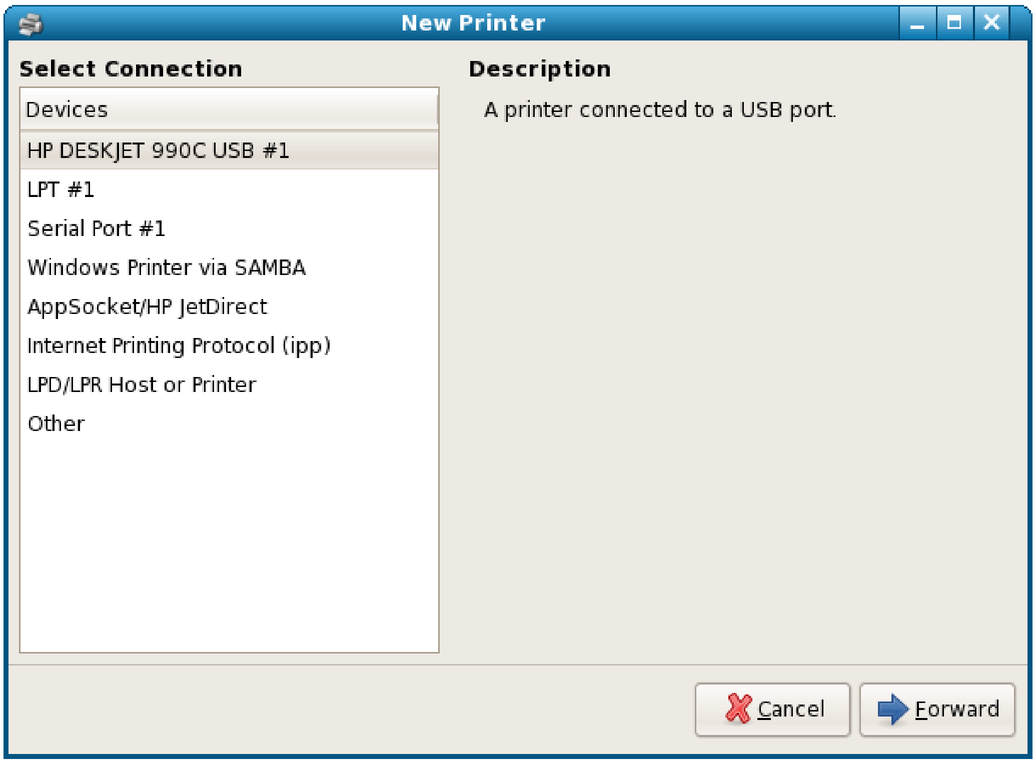
☐ Allow printing from the Internet

☐ Allow remote administration

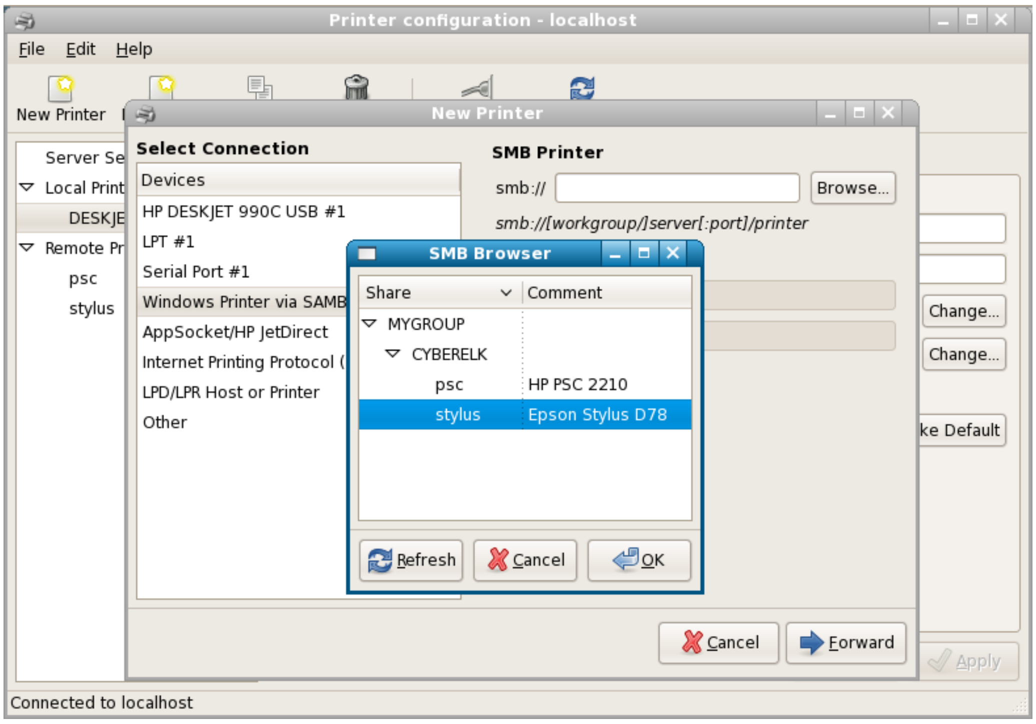
☐ Allow users to cancel any job (not just their own)

☐ Save debugging information for troubleshooting

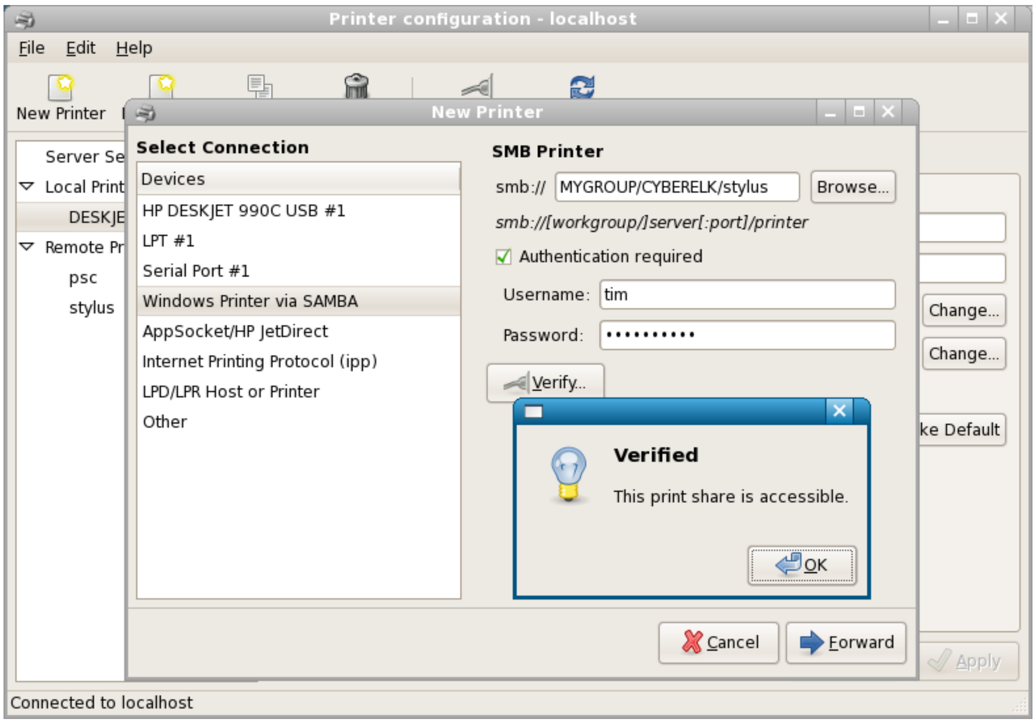
For remote printers it is not so straight-forward. To add a new remote queue the administration tool must be started “manually” from the menu (*System* ▸ *Administration* ▸ *Printing* on Fedora). After clicking the New button a wizard is displayed showing the available printer devices and ports.



To add an SMB printer, for instance, click on “Windows Printer via SAMBA”. The Browse button allows the available SMB printers to be examined.



The SMB share can be verified to check that the authentication details are correct and that the printer is accessible.



Printer configuration - localhost

New Printer

Select Connection

Devices
HP DESKJET 990C USB #1
LPT #1
Serial Port #1
Windows Printer via SAMBA
AppSocket/HP JetDirect
Internet Printing Protocol (ipp)
LPD/LPR Host or Printer
Other

SMB Printer

smb:// MYGROUP/CYBERELK/stylus

*smb://[workgroup/]server[:port]/printer*

✓ Authentication required

Username: tim

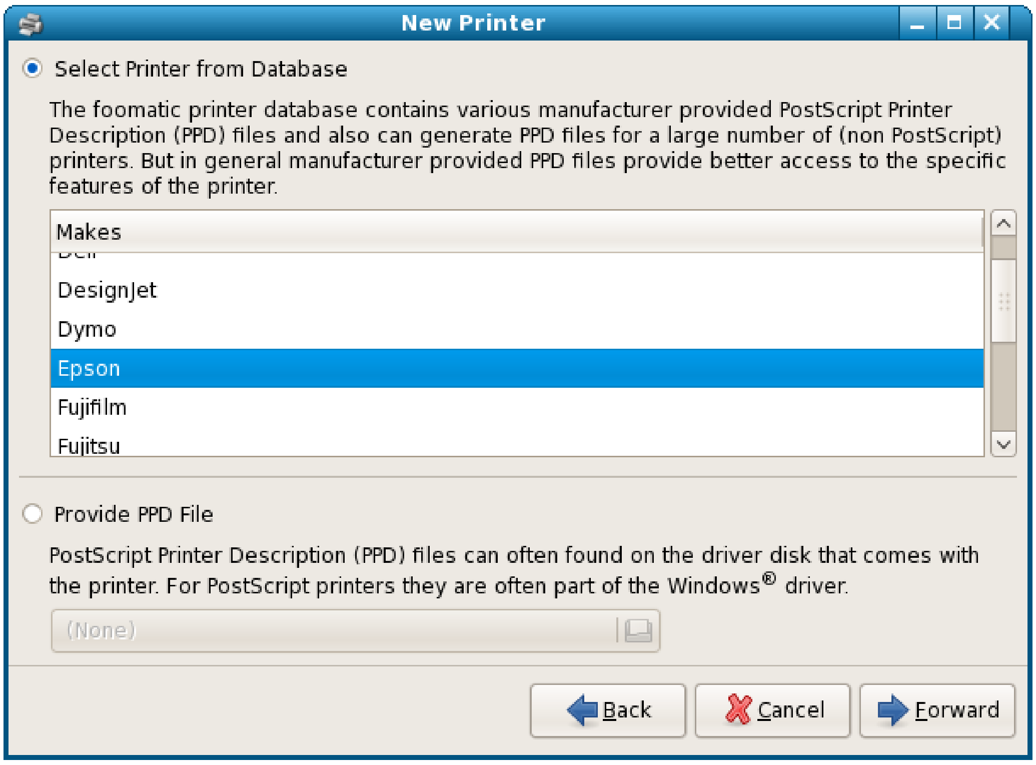
Password: ••••••••••

**Verified**

This print share is accessible.

Connected to localhost

After clicking Forward, the printer manufacturer is selected.  At this point there is the opportunity to provide a PPD file that was supplied with the printer.



New Printer

[✓ Select Printer from Database]

The foomatic printer database contains various manufacturer provided PostScript Printer Description (PPD) files and also can generate PPD files for a large number of (non PostScript) printers. But in general manufacturer provided PPD files provide better access to the specific features of the printer.

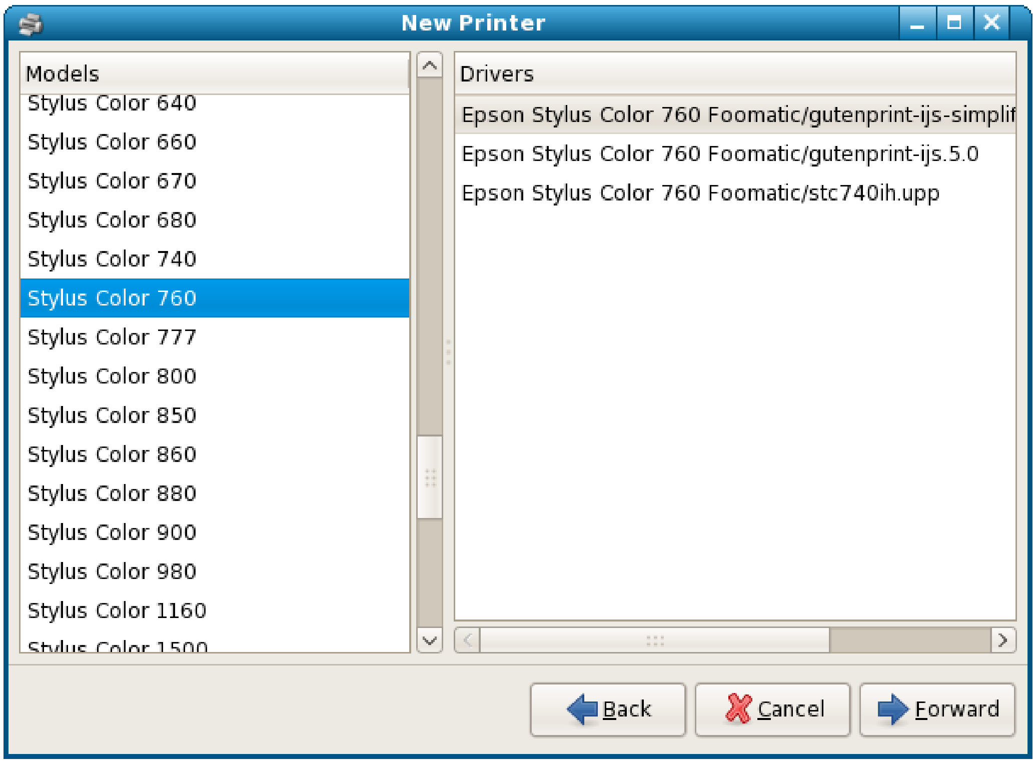
Makes
DesignJet
Dymo
Epson
Fujifilm
Fujitsu

Provide PPD File

PostScript Printer Description (PPD) files can often found on the driver disk that comes with the printer. For PostScript printers they are often part of the Windows® driver.

(None)

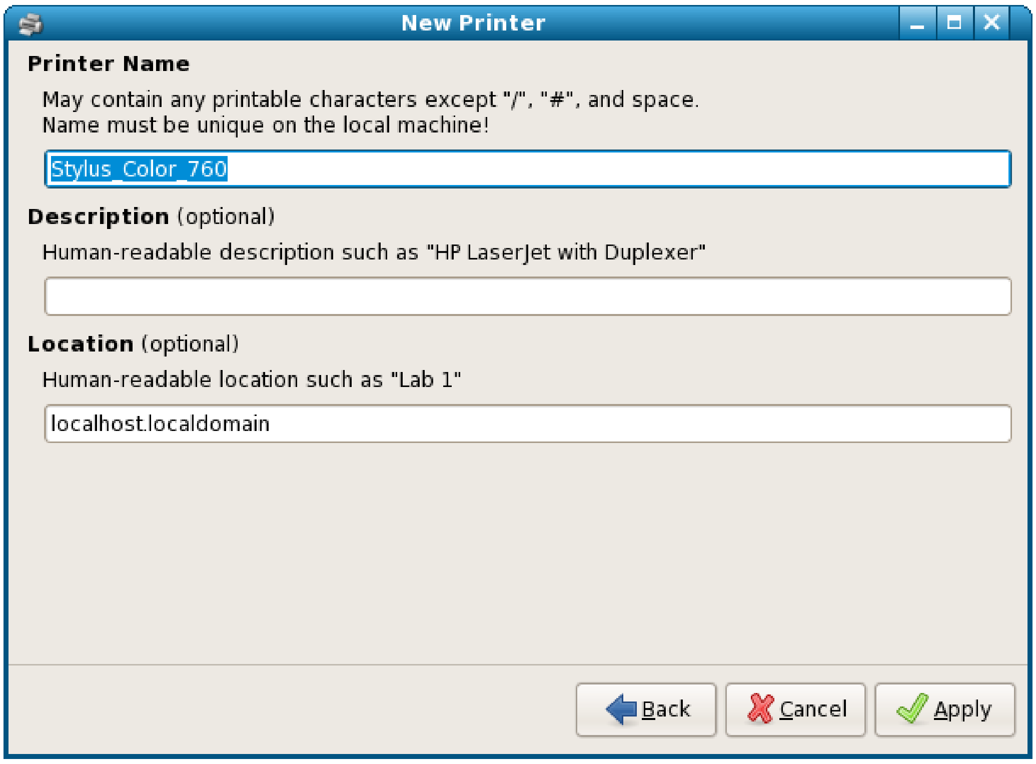
The next page in the wizard allows the printer model to be selected.



Stylus Color 640
Stylus Color 660
Stylus Color 670
Stylus Color 680
Stylus Color 740
Stylus Color 760
Stylus Color 777
Stylus Color 800
Stylus Color 850
Stylus Color 860
Stylus Color 880
Stylus Color 900
Stylus Color 980
Stylus Color 1160
Stylus Color 1500

Epson Stylus Color 760 Foomatic/gutenprint-ijs-simplif
Epson Stylus Color 760 Foomatic/gutenprint-ijs.5.0
Epson Stylus Color 760 Foomatic/stc740ih.upp

Finally, the new printer is given a name, as well as a description and location.



**Printer Name**

May contain any printable characters except "/", "#", and space.
Name must be unique on the local machine!

Stylus_Color_760

**Description** (optional)

Human-readable description such as "HP LaserJet with Duplexer"

**Location** (optional)

Human-readable location such as "Lab 1"

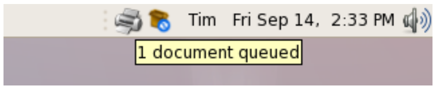
localhost.localdomain

### *The job management applet*

When a desktop session is started the job management applet gets launched. Initially it waits for D-Bus messages before loading GTK+. When it is time for action it loads GTK+ and displays a printer icon in the notification area. This printer icon is shown sensitive when the current user has current jobs in the queue, and insensitive otherwise.

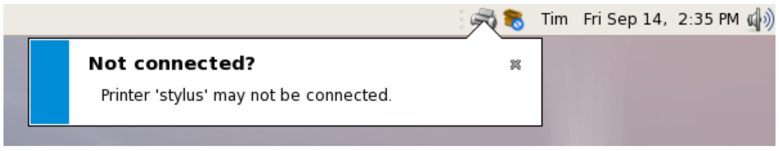
There are two reasons for the icon appearing: either the user has submitted a print job, or the user is at the console and a USB printer has been connected. In this way those not wishing to be affected by anything to do with printers can be sure that the job management applet is not taking up too much in the way of resources.

CUPS provides D-Bus signals on the system bus for important events such as a job being submitted or completed. The applet relies on these signals entirely; it does not poll the server. It is therefore only useful for a local CUPS scheduler. When a D-Bus signal is received, the applet queries the job queue for active jobs owned by the current user; when there are such jobs, the icon is displayed:



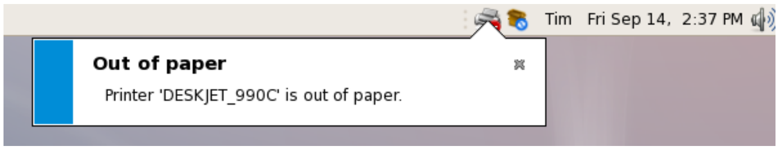
1 document queued

Notification bubbles about printer problems that might affect the current user are attached to this icon.



**Not connected?**

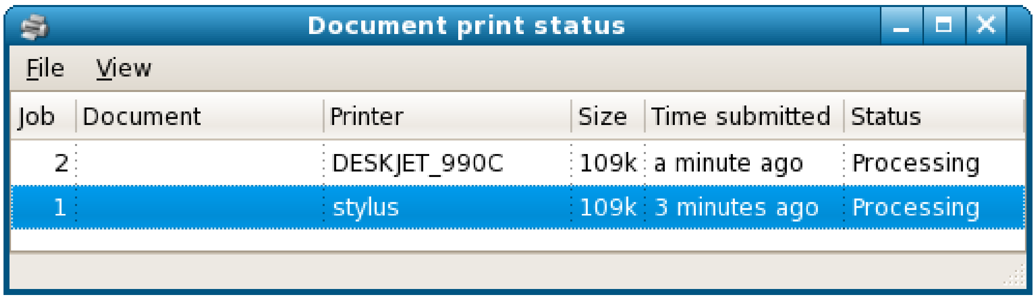
Printer 'stylus' may not be connected.



**Out of paper**

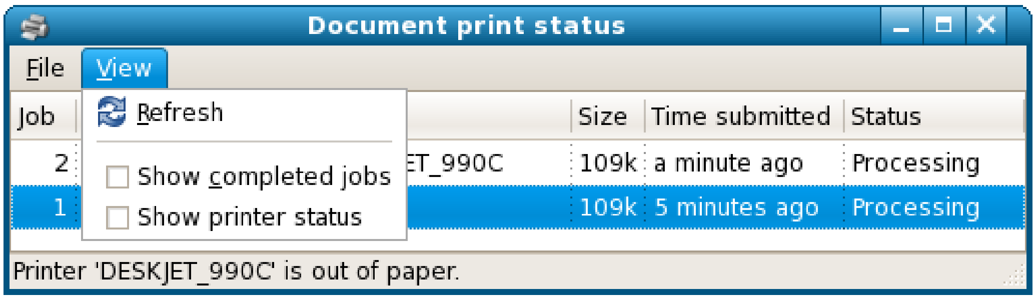
Printer 'DESKJET_990C' is out of paper.

Clicking on the icon displays the job queue:



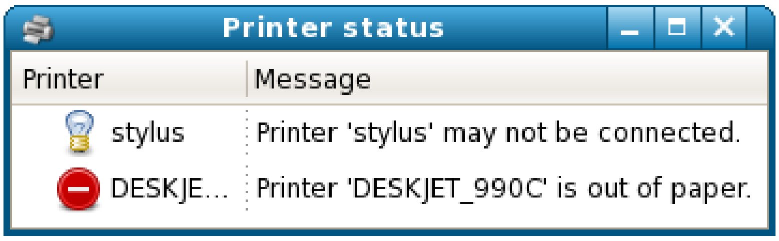
| Job | Document | Printer | Size | Time submitted | Status |
|---|---|---|---|---|---|
| 2 | | DESKJET_990C | 109k | a minute ago | Processing |
| 1 | | stylus | 109k | 3 minutes ago | Processing |

The current printer issues can be viewed by selecting a menu item. This corresponds with the "printer-state-reasons" IPP printer attribute for each printer.



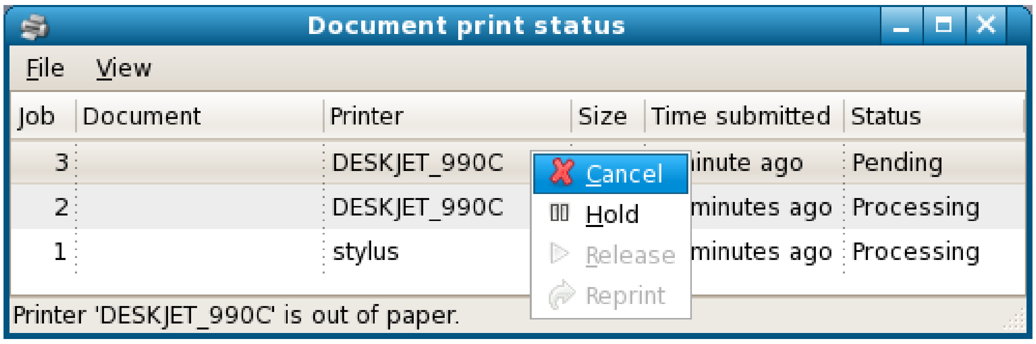
| Job | | Size | Time submitted | Status |
|---|---|---|---|---|
| 2 | ET_990C | 109k | a minute ago | Processing |
| 1 | | 109k | 5 minutes ago | Processing |

Printer 'DESKJET_990C' is out of paper.



| Printer | Message |
|---|---|
| stylus | Printer 'stylus' may not be connected. |
| DESKJE... | Printer 'DESKJET_990C' is out of paper. |

Of course, job management is the main purpose of the application. The job context menu is pictured here:



| Job | Document | Printer | Size | Time submitted | Status |
|---|---|---|---|---|---|
| 3 | | DESKJET_990C | | inute ago | Pending |
| 2 | | DESKJET_990C | | minutes ago | Processing |
| 1 | | stylus | | minutes ago | Processing |

Printer 'DESKJET_990C' is out of paper.

If the job management window is needed when the printer icon is not shown (for example, before a job has been printed), it can be launched from the menu (*Applications* ► *System Tools* ► *Manage Print Jobs* on Fedora).

## *Open issues and future directions*

The development of this application is active. Florian Festi (ffesti@redhat.com) wrote system-config-printer, while I (Tim Waugh) wrote the pycups CUPS bindings for it. I am the maintainer for pycups, system-config-printer, and hal-cups-utils, all of which are hosted in Subversion repositories at fedoraproject.org. Florian, myself, and Till Kamppeter have commit access to system-config-printer, and patches have been submitted by various contributors.

In recent months there have been many large improvements in the code base; however, the nature of software is that it is never finished. A selection of the issues to be addressed are described below.

- The user interface is quite "busy" at first glance. There are a lot of widgets in the window at once and this is not easy to take in when unfamiliar with how the application behaves. It could perhaps be an improvement to show icons representing printers rather than displaying a textual tree view. Getting user interfaces right is tricky. It may be that the icons would look nice but be harder to use than the current state of affairs.
- As the purpose of the application is CUPS administration, extra features in newer versions of CUPS means there is a certain amount of "catching-up" to be done at the moment. One case in point is the "auth-info-required" IPP printer attribute implemented in CUPS 1.3. As I understand it, this feature allows configuration for password-protected printers to have the security-sensitive details stored separately from the rest of the configuration. This is especially useful for SMB printers, where currently the password is contained in the device URI. There is currently no support for this attribute in system-config-printer, nor in pycups.
- Another CUPS 1.3 feature is the ability to fetch a named PPD (as distinct from a PPD in use for a queue). This is useful for the administration tool when it comes to displaying a useful interface to the user who is choosing a printer driver. The new libcups function has been bound in the most recent version of pycups, but a re-worked PPD selection screen in system-config-printer is still in progress.
- When CUPS 1.4 is ready there will be a facility for fetching job data files. This is interesting from the point of view of the job management applet. It raises the possibility of showing job thumbnails, among other things.
- Some work is still needed to make CUPS authentication work "nicely" for those using system-config-printer. In Fedora the application can be started as root (when the system password is supplied), but this is far from ideal. In Ubuntu it runs as the current user, and the first user added to the system is also granted membership of a special group "lpadmin", and the shipped CUPS configuration lets users in this group perform administration tasks.

An ideal situation would be for any user at the console to be able to perform administration tasks. That way, the administration tool could run as the non-root user, and could also add and configure printers. It may be that ConsoleKit would be useful to this end.

- The job management applet is a recent addition to the system-config-printer package, and pycups extended its API coverage to the job management functions in libcups for it. The test page function in the administration tool has very crude controls for cancelling the test page, but there is no proper integration between the two programs.
- Similarly, D-Bus signals are something that the job management applet catches and system-config-printer ignores. It would be very useful for system-config-printer to watch for D-Bus signals about new printers added, state changes, and so on.

## *Conclusion*

One of the reasons the original Red Hat Linux/Fedora printer configuration tool was bad was the fact that it was trying to be “print server neutral”. It was designed when LPRng was shipped with Red Hat Linux, and had CUPS support added on afterwards. It was too generic in its approach to be good at configuring any particular print server.

In choosing to design a tool for configuring CUPS servers, and for the tool to use the libcups API to do so, I believe that pitfall has been avoided. I hope that system-config-printer has chosen its one thing to do, and will do it well.