The KDEPrint Handbook

Version 1.00.04−b

Kurt Pfeifle, Danka Deutschland GmbH

The KDEPrint Handbook

Kurt Pfeifle

<kpfeifle@danka.de>

Developer: Michael Goffioul

Reviewer: Lauri Watts

Revision 1.00.04−b

This handbook describes KDEPrint. KDEPrint is not a standalone program. It is the new printing framework for KDE 2.2. KDEPrint is an intermediate layer between KDE (or other) applications and the selected (and installed) print subsystem of your OS (operating system).

Table of Contents

Next
Introduction

Prev Next

Chapter 1. Introduction

This handbook describes KDEPrint. KDEPrint is not a standalone program. It is the new printing framework for KDE 2.2. KDEPrint is an intermediate layer between KDE (or other) applications and the selected (and installed) print subsystem of your OS (operating system).

It should be noted that both the developer of this application, and the author of this document are most familiar with CUPS as a printing system. At the time of writing, CUPS is the best supported printing subsystem, and it is the best documented.

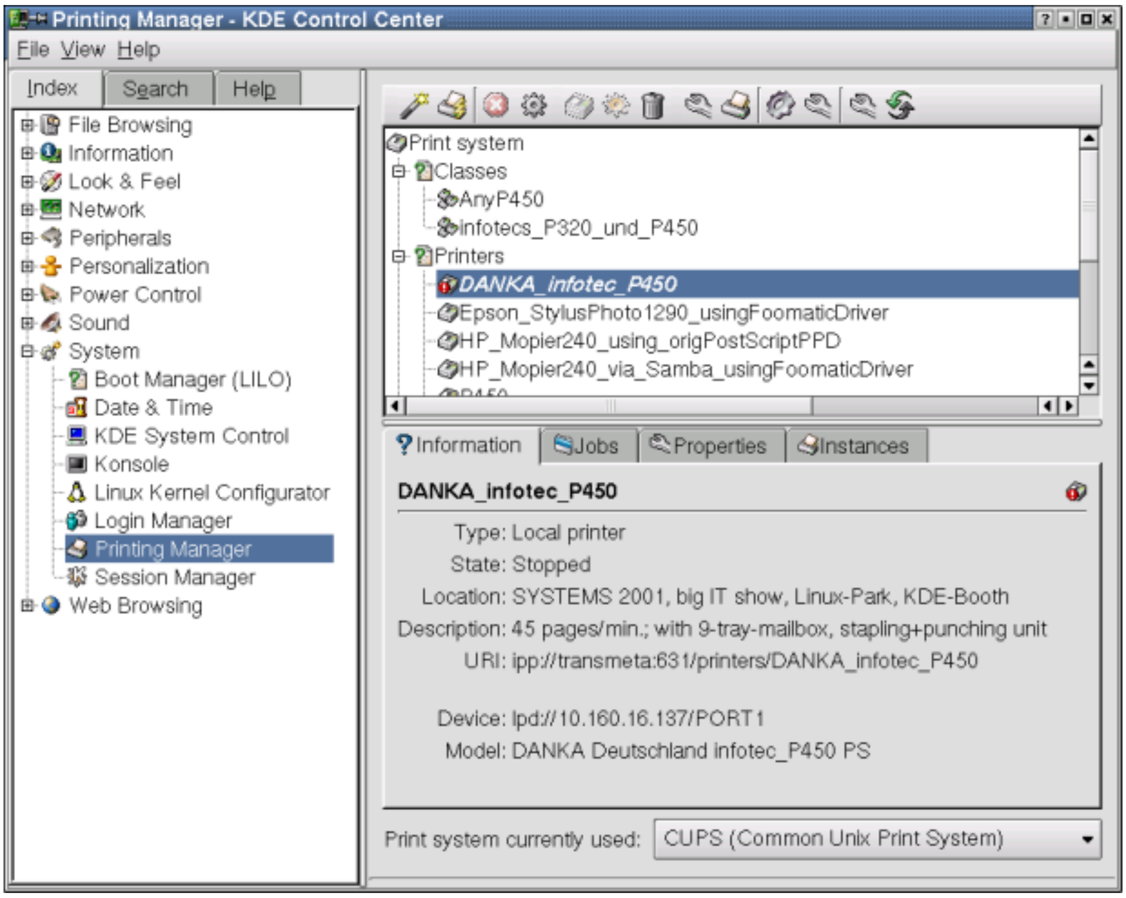
This handbook is work in progress, and later versions of the KDEPrint software and editions of this handbook will support and explore more closely other printing systems. Also, because KDEPrint is evolving so rapidly, the Handbook might not be in sync with the software version on your computer.

In the meantime, even if your printing subsystem is not yet well covered, you are encouraged to explore the Printing Manager module in KControl, and you will find its operation to hopefully be fairly self evident, no matter what printing subsystem you use.

Lauri Watts, KDE documentation team

To configure your printing subsystem from KControl

To configure your printing subsystem from KControl, go to System+Printing Manager and select your subsystem. Or you can let KDEPrint guess it...



DANKA_infotec_P450

Type: Local printer
State: Stopped
Location: SYSTEMS 2001, big IT show, Linux-Park, KDE-Booth
Description: 45 pages/min.; with 9-tray-mailbox, stapling+punching unit
URI: ipp://transmeta:631/printers/DANKA_infotec_P450

Device: lpd://10.160.16.137/PORT1
Model: DANKA Deutschland infotec_P450 PS

Print system currently used: CUPS (Common Unix Print System)

Configuration of printing subsystem from KControl

Prev	Home	Next
The KDEPrint Handbook	Up	Technical Overview

Chapter 2. Technical Overview

This chapter aims to give a technical overview of KDEPrint which is comprehensible for non−programmers.

KDEPrint is a new and revolutionary tool to give easy access to printing services for both KDE users and KDE developers.

A Brief Description of KDEPrint

You can access the functions of KDEPrint in different ways: through the Printing Manger in the KControl, through the **kprinter** command or through the dialog that pops up if you want to print.

What it is *not*

KDEPrint is *not* a replacement of the printing subsystem itself. KDEPrint therefor does *not* provide for the spooling, and it does *not* do the basic processing of PostScript® or other print data.

What it *is*

KDEPrint is an intermediate layer between the spooling and the data processing print subsystem (as installed), and the application that seeks to print. KDEPrint provides a common interface, for KDE developers and KDE users, to different supported print subsystems. At the same time, it is customizable, and highly configurable.

KDEPrint is easy to use for KDE developers and end−users. Developers can port their applications with minimal changes to use KDEPrint instead of the old Qt" print system . Users can easily choose and configure their print subsystem.

For a reference to new KDE users: Qt" is the basic library and graphical toolkit, which is used by all KDE applications; Qt" is developed by TrollTech, a Norwegian software company).

KDEPrint −− Different Usage for Different People

KDEPrint has different faces for different people.

What users and administrators can do with KDEPrint

KDEPrint allows users and/or administrators, depending on their rights, to access printing subsystems (CUPS, LPD, RLPR, LPRng, PDQ etc.) through a KDE graphical user interface (GUI). Using KDEPrint, they can print, administer jobs, printers and the printing daemon, all in a comfortable manner.

Experienced users will like the possibility to plug any working filter for the print data between the output of their application and the input into the chosen print subsystem. Some examples for this already ship with plain vanilla KDEPrint. Read on.

What KDE developers can do with it...

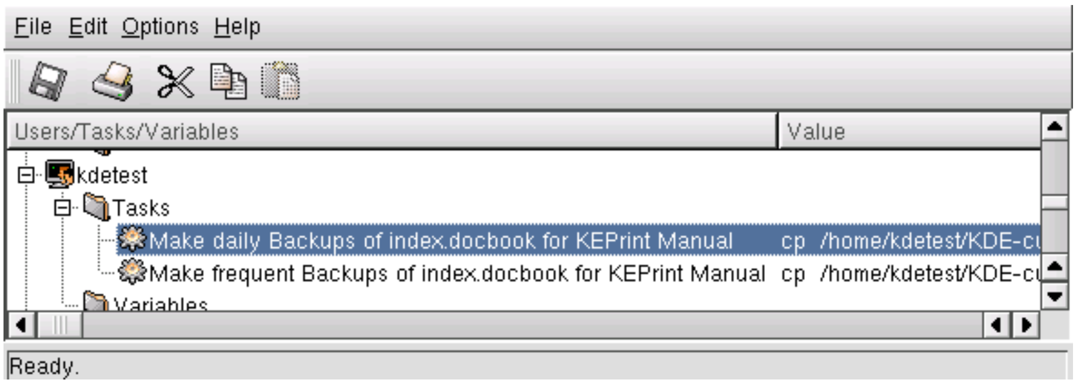
If a KDE developer needs printing access for his application, he does not code the printing functions from scratch. Before KDE 2.2 this service was provided by the `QPrinter` class, a library function of the Qt" Toolkit. The `QPrinter` class relied on the out−moded Line Printer Daemon (LPD). The KDEPrint library bases itself firmly on the more modern Common UNIX® Printing System (CUPS), while at the same time keeping backward compatibility to LPD and other legacy or less elaborate print systems, and also leaving the door open to any new development that might occur.

For KDE developers to use the new KDEPrint class in their applications, they need just minimal changes in their code: for every call of `QPrinter`, they just need to change this to `KPrinter`. Replacing one (!) letter in a few spots, and automatically they are done; their application then can use all the features of the new KDEPrint library.

More ambitious developers, or ones with special needs can do more: despite KDEPrint's feature−rich framework, they are still able to customize the print dialog of their application by letting an additional Tab appear, where their extensions to the standard KDEPrint will feel right at home.

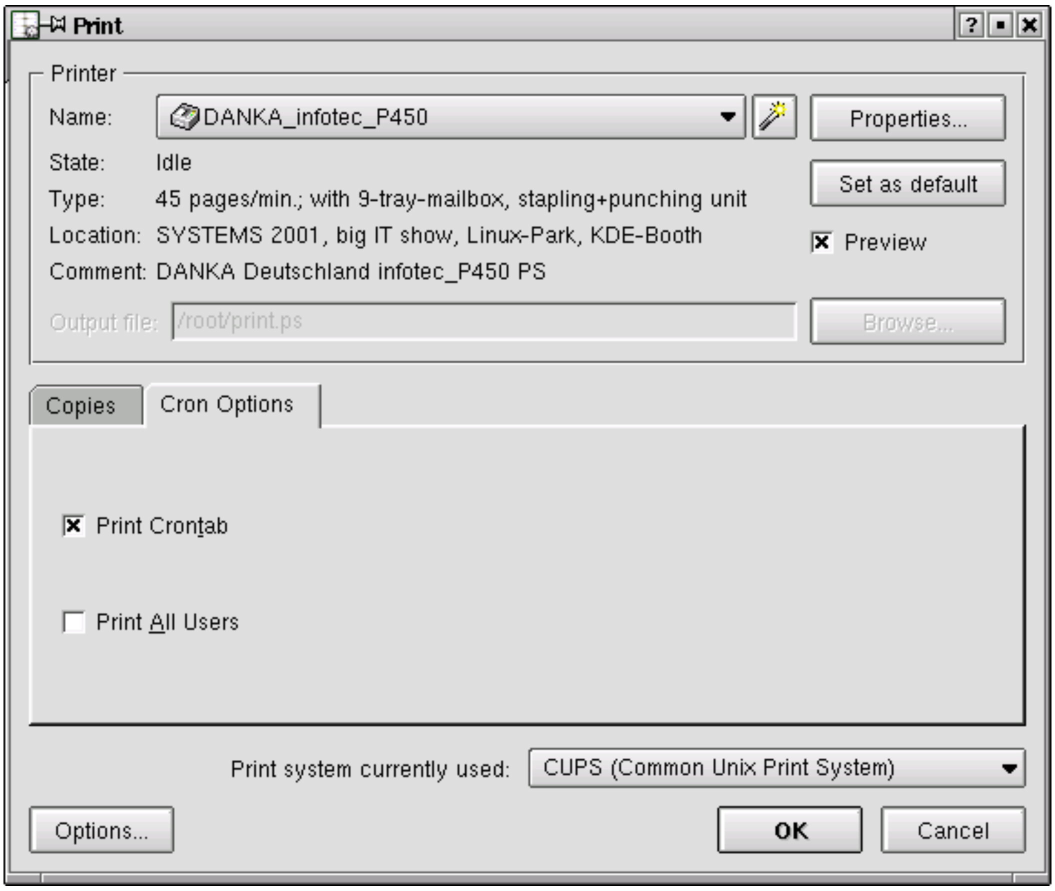
This last mentioned feature is not being used widely inside KDE so far, as developers are not yet fully aware of KDEPrint's power. Expect more of this in the near future. One example I discovered is the Kcron application. It lets you edit the crontab through a GUI. The developers have implemented a printing feature that lets you (or `root`) choose if you want to print the whole of crontab (for all users) or just the part that is marked. You can see the effects on KDEPrint in the following screenshots.

This shot shows a sample from the Kcron utility.



Make daily Backups of index.docbook for KEPrint Manual cp /home/kdetest/KDE-cu
Make frequent Backups of index.docbook for KEPrint Manual cp /home/kdetest/KDE-cu

The dialog to configure Kcron's printing options: the additional tab titled Cron Options is from inside Kcron, not KDEPrint; it is a special extension added by the Kcron developers for printing purposes, not originating from, but executed by the KDEPrint. Developers of other applications are free to implement their own goodies, if they feel need for it.



Printer

Name: DANKA_infotec_P450

State: Idle

Type: 45 pages/min.; with 9-tray-mailbox, stapling+punching unit

Location: SYSTEMS 2001, big IT show, Linux-Park, KDE-Booth

Comment: DANKA Deutschland infotec_P450 PS

Output file: /root/print.ps

[✓ Preview]

[✓ Print Crontab]

[ ] Print All Users

Print system currently used: CUPS (Common Unix Print System)

Kcron's addition to the KDEPrint dialog.

What KDEPrint offers to everybody...

KDEPrint's easy−to−use interface towards all supported print subsystems of course does not eliminate basic traditional weaknesses of some of those systems. But it smoothes some rough edges. Different users may use different printing systems on the same box. A user is free to even switch on the fly, from the print dialog, the print subsystem to be used for the next job. (This is possible if different systems are installed in a way that they don't get in each other's way.)

Most PC users are used to LPD printing. LPD provides only basic printing functions, is very inflexible and does not utilize the many options of more modern print systems like CUPS. While also working remotely over any distance (like every TCP/IP based protocol), LPD lacks bi−directional communication, authentification, access control and encryption support.

KDEPrint can use CUPS to support:

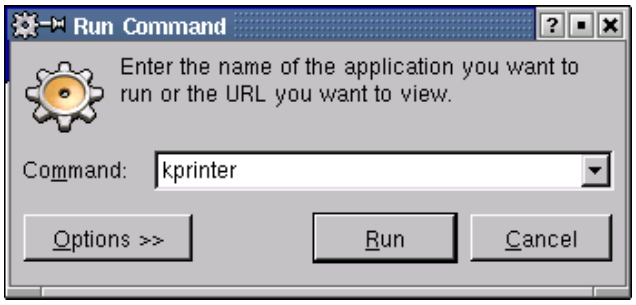
- Querying the LAN for available printers,
- Basic, Digest, and Certificate Authentication,
- Access Control based on IP addresses, net addresses, netmasks, host− and domain names,
- and 128−Bit TLS or SSL3 encryption of print data, to prevent eavesdropping, or at least make it much more difficult.

This makes KDEPrint a much more robust and reliable solution than using the venerable LPD.

How to access KDEPrint

You get access to KDEPrint or parts of it in four different ways:

- through your applications: if you call the printing dialogue (either File+Print...) or the button with the little printer icon on it; this opens the printing dialogue.
- through the typed command **kprinter** in a terminal or a Konsole window or from the Run Command... mini−CLI window: this also opens the printing dialogue.
- from the button, starting KControl, and then go to System+Printing Manager. This opens the KDEPrint administration as part of the KDE Control Center and lets switch to other parts of the KControl
- from a commandline (Konsole or mini−CLI) type `kcmshell printmgr`. This opens just the KDEPrint part of KControl to do your settings

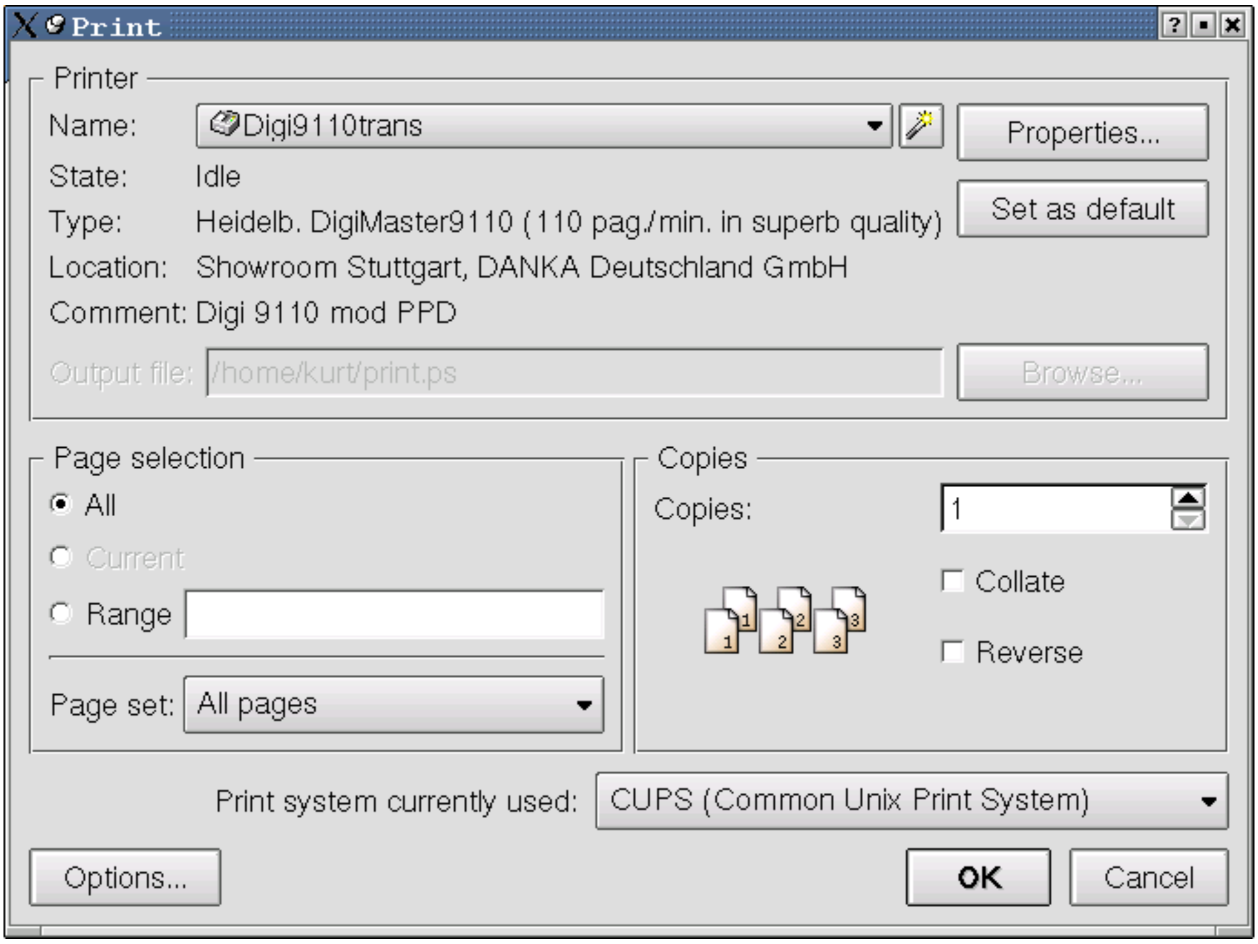

Run Command

Enter the name of the application you want to run or the URL you want to view.

Command: kprinter

Starting the **kprinter** dialogue from a Run Command... window.

Here is a Kivio drawing of the **kprinter** dialogue as it pops up after being started... You can always add a new printer by clicking on the small Wizard button (marked red/yellow in this drawing).


Print
Printer
Name: Digi9110trans
Properties...
Set as default
State: Idle
Type: Heidelb. DigiMaster9110 (110 pag./min. in superb quality)
Location: Showroom Stuttgart, DANKA Deutschland GmbH
Comment: Digi 9110 mod PPD
Output file: /home/kurt/print.ps
Browse...
Page selection
All
Current
Range
Page set: All pages
Copies
Copies: 1
Collate
Reverse
Print system currently used: CUPS (Common Unix Print System)
Options...
OK
Cancel

kprinter dialogue started (Kivio draft drawing)

Prev
Technical Overview

Home
Up

Next
KDEPrint's Highlights

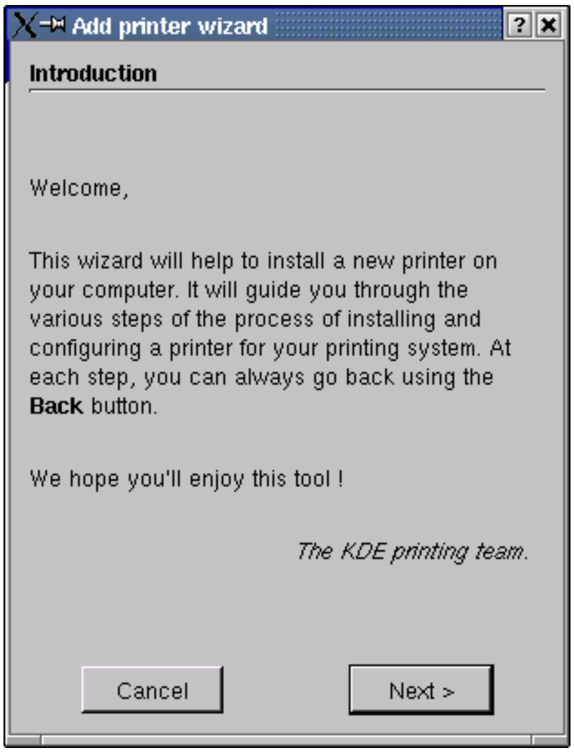
Chapter 3. KDEPrint's Highlights

The new KDEPrint system includes more than one highlight. Having worked in a not exactly sophisticated environment as far as printing is concerned, in previous years, have a look at some of the goodies that come with KDEPrint

The Add Printer Wizard

KDEPrint has an Add Printer Wizard . The Add Printer Wizard helps you adding and configuring a new printer. Of course, you may do this manually as well.

KDEPrint helps you discover printers. It is able to scan the environment for available devices and queues. This works for network connections against TCP (AppSocket, aka HP® JetDirect®, or IPP) or SMB/Samba (shared Windows®) printers and partially for directly attached printers over parallel, serial, or USB connections.


Add printer wizard

Introduction

Welcome,

This wizard will help to install a new printer on your computer. It will guide you through the various steps of the process of installing and configuring a printer for your printing system. At each step, you can always go back using the **Back** button.

We hope you'll enjoy this tool !

The KDE printing team.

The wizard makes the installation and handling of the drivers a snap . Selecting, configuring and testing should be easy as never before on any Linux®−like system.

Full Print Job Control

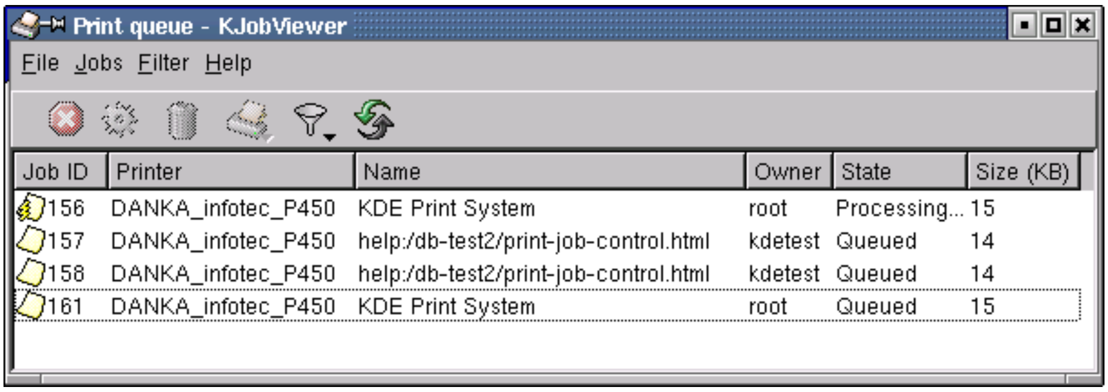
Full Print Job Control

The Print Job Viewer is automatically started by **kprinter**. It may be docked into the KDE panel (in the system tray). The Print Job Viewer allows full job management, if supported by the print subsystem.

You can:

- Hold and release jobs,
- Move pending jobs to another printer,
- Cancel pending or processing jobs.

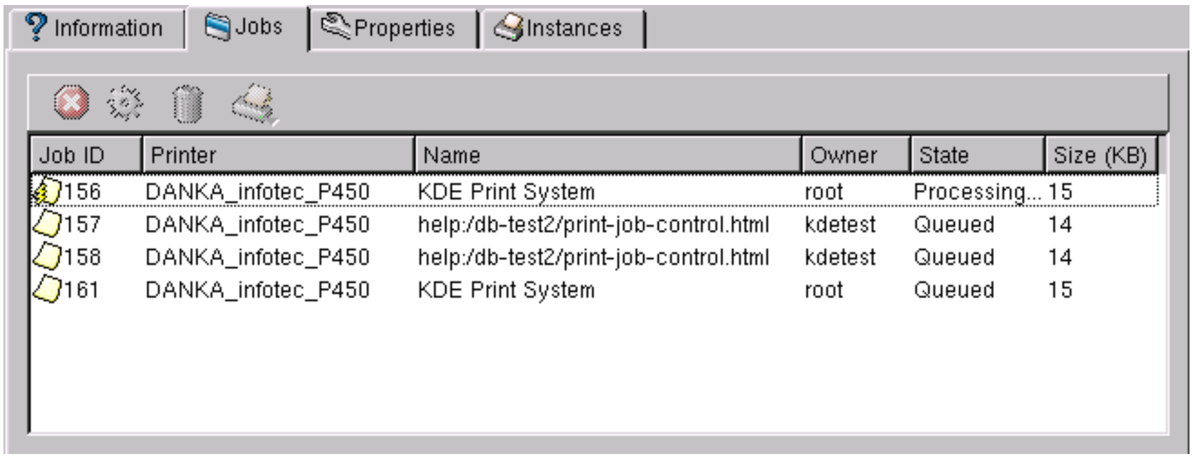
A screenshot of the KDEPrint PrintJob Viewer shows the info you get: Job−ID, target printer, job name, job owner, job status and job size. In the next KDEPrint release you will also see information about the number of pages (as CUPS calculates it; see chapter on page accounting for more info about its merits and limitations).



Job ID	Printer	Name	Owner	State	Size (KB)
156	DANKA_infotec_P450	KDE Print System	root	Processing...	15
157	DANKA_infotec_P450	help:/db-test2/print-job-control.html	kdetest	Queued	14
158	DANKA_infotec_P450	help:/db-test2/print-job-control.html	kdetest	Queued	14
161	DANKA_infotec_P450	KDE Print System	root	Queued	15

A screenshot of the KDEPrint PrintJob Viewer.

An alternative way to looking at the same info (and having the same amount of control is through the KDE Control Center going to System+Printing Manager. If you don't see the Printer Information, right click onto the window background and select View Printer Informations. Then go to the Jobs tab to see this:



Job ID	Printer	Name	Owner	State	Size (KB)
156	DANKA_infotec_P450	KDE Print System	root	Processing...	15
157	DANKA_infotec_P450	help:/db-test2/print-job-control.html	kdetest	Queued	14
158	DANKA_infotec_P450	help:/db-test2/print-job-control.html	kdetest	Queued	14
161	DANKA_infotec_P450	KDE Print System	root	Queued	15

Prev | Home | Next
KDEPrint's Highlights | Up | Modules for different print subsystems

Modules for different print subsystems

Modules for different print subsystems

KDEPrint uses different modules to realize the interface to the possible print subsystems. Not all the modules are yet developed fully, but you will have basic printing functionality with:

- LPD (BSD style)
- LPRng (Red Hat®, if you just use it's BSD style subset),
- RLPR (a command−line LPR utility, which doesn't need a `printcap` file.
- external print commands (Netscape® like).

Most importantly, full support for CUPS is already there. Modules for other print subsystems, such as PLP, PPR and PDQ may be available later.

KDEPrint makes KDE much more flexible. It gives freedom of choice to KDE 2.2 users. To use different available print subsystems, these must, of course, be installed independently from KDE. In former versions, users were stuck with the old LPD style print subsystems. Now they can even use CUPS. In the future, there will be easy integration of new subsystems, as they might appear on the scene.

Prev Home Next
Full Print Job Control Up More KDEPrint Goodies

More KDEPrint Goodies

Benefitting all Print SubSystems.

Some specific features of KDEPrint depend on the chosen print subsystem. This dependency might exist because those features are only implemented there; remember, KDEPrint is an intermediate layer between KDE applications, and the print subsystem, but it's no replacement for any print subsystem by itself. Such dependency may exist for another reason: that KDEPrint has not yet implemented an interface to all the features of all the subsystems.

Other features include goodies from KDEPrint that are independent of the chosen print subsystem, and are available with every one. At present there are special or virtual printers, and some generic pre−filters .

Print Preview
: From the Print Dialog, you can select to look at a preview. For this the print file is sent through filters which make it fit for display on screen by Kghostview.

Special Printers
: Amongst these additional KDEPrint features are a few special or virtual printers:

  These special printers may:

  Print to PDF
  : Convert our document into a PDF with the help of an external program.

  Print to email
  : Send your document as an email attached PDF file.

  Print to PS file
  : Save your document as a PostScript® file.

  Print to Fax
  : Send it through an available backend, such as Hylafax as a fax.

  These special printers appear in the user print dialog just like normal printers. They are entirely configurable on a per−user basis.

Generic Pre−Filtering
: KDEPrint provides you with a framework to define and configure your own pre−filters . These pre−filters may take effect *before* they are passed to your print subsystem for further processing, but *after* the (PostScript® or text or other) print files have been generated by your application.

  There are a few useful filters already predefined. These are:

  - The multiple pages per sheet filter,
  - the enscript text filter,
  - and three filters to help print pamphlets.

  You may create your own filters based on any third party program that is able to process PostScript®, text, or image files, and output any one of those formats.

  Those filters are configured through XML files. This makes an extension of the concept very easy for experienced geeks, but end−user configuration is also done through an intuitive graphical user interface. So, fear not, you don't need to learn XML because of KDEPrint!.

Multiple Pages Per Sheet Filter
: This is a predefined filter that installs with KDEPrint. It allows you to create a modified PostScript® output, from PostScript® input, that prints 1, 2, or 4 logical pages on a single sheet of physical paper.

Enscript Text Filter
: This is a predefined filter that installs with KDEPrint. It allows you to create PostScript® output from any text file input, that includes syntax highlighting for program listings, pretty−print, and nice configurable page frames and headers.

Pamphlet Printing Filters
: If your printer is able to produce duplex output, using either one−pass or two−pass technology, you may be able to use one or a combination of the pamphlet filters.

For duplexing printers, make sure to use the duplex option that turns the output along the short paper edge. Folding the output along the middle turns your document into a nice pamphlet.

If you are stuck to using a simplex−only device, you can do the same, using two different filters and a few additional steps.

Depending on your model, first use the filter for printing the odd pages, then insert the paper in the correct order back into the paper tray to get the even pages printed on the reverse side. Finish by folding.

Prev | Home | Next
Modules for different print subsystems | Up | CUPS Support: the Most Important Module in KDEPrint

CUPS Support: the Most Important Module in KDEPrint

CUPS Support: the Most Important Module in KDEPrint

KDEPrint contains a module for CUPS. CUPS, the Common UNIX Printing System (http://www.cups.org/), is the most advanced, powerful and flexible of all print subsystems on UNIX® and other Linux® like operating systems. It is still quite new on the horizon, but is based on IPP, the Internet Printing Protocol, the newly emerging standard for the future of network printing. CUPS is clearly the print system of choice for Michael Goffioul, the principal KDEPrint developer.

Experienced KDE users may already be familiar with Michael's utilities qtcups and kups (co−developed with Jean−Eric Cuendet). These were up until now the graphical GUI front ends for CUPS with a strong relation to KDE.

qtcups and kups The Predecessors

Both utilities are probably still in wide usage. For those not familiar with them, here are short explanations.

qtcups was a graphical front end for the **lp** or **lpr** print commands as installed by CUPS. Using qtcups opened a dialog. This dialog let you comfortably select your printer and the print job options. qtcups worked from the command line, or from within applications, when the application in question had a configurable print command.

kups was a graphical wrapper to do the administration tasks for your CUPS server, and the CUPS daemon at the heart of it. You could add, delete, modify, configure, start, and stop printers. You could cancel, delete, move, stop and restart print jobs, and you could change the settings of the daemon, start, stop, and restart it.

KDEPrint The Heir

The CUPS Module in KDEPrint now contains all (and more) functions that were provided by qtcups and kups in former KDE versions.

Instead of **qtcups** you now can use the **kprinter** command. And in place of **kups** you will in future probably use **kcmshell printmgr**.

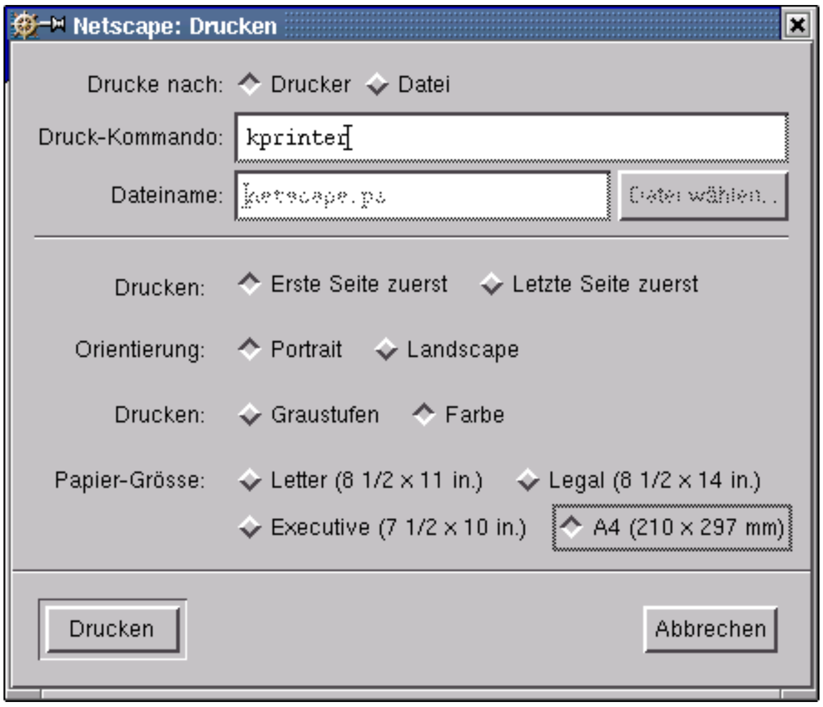
The KDEPrint module for CUPS also lets you fully administer the print subsystem, just like kups did before. It can start, stop and configure your CUPS daemon. It also can stop, start, add and delete printers (i.e. printer queues) and printer instances . Printer instances are printer queues that point to the same physical output device but with a different default setting of print options.

kprinter Graphical Print Command

KDEPrint's CUPS module gives you access to a graphical print command , like qtcups did before.

Use **kprinter** in any application, even non KDE application that lets you configure your print command. Examples of this are Netscape® and StarOffice, but *not* most pre−KDE 2.2 programs.

A screenshot how to use the new **kprinter** print command instead of the old−fashioned **lpr**... (Of course you need to have **kprinter** in your $PATH for this short version; otherwise give the full path in the dialog once, e.g. **`/opt/kde/bin/kprinter`**. Netscape® will remember this and from now on you always get the **kprinter** dialog to configure your printouts.



Netscape: Drucken

Drucke nach: Drucker Datei

Druck-Kommando: kprinter

Dateiname: netscape.ps Datei wählen...

Drucken: Erste Seite zuerst Letzte Seite zuerst

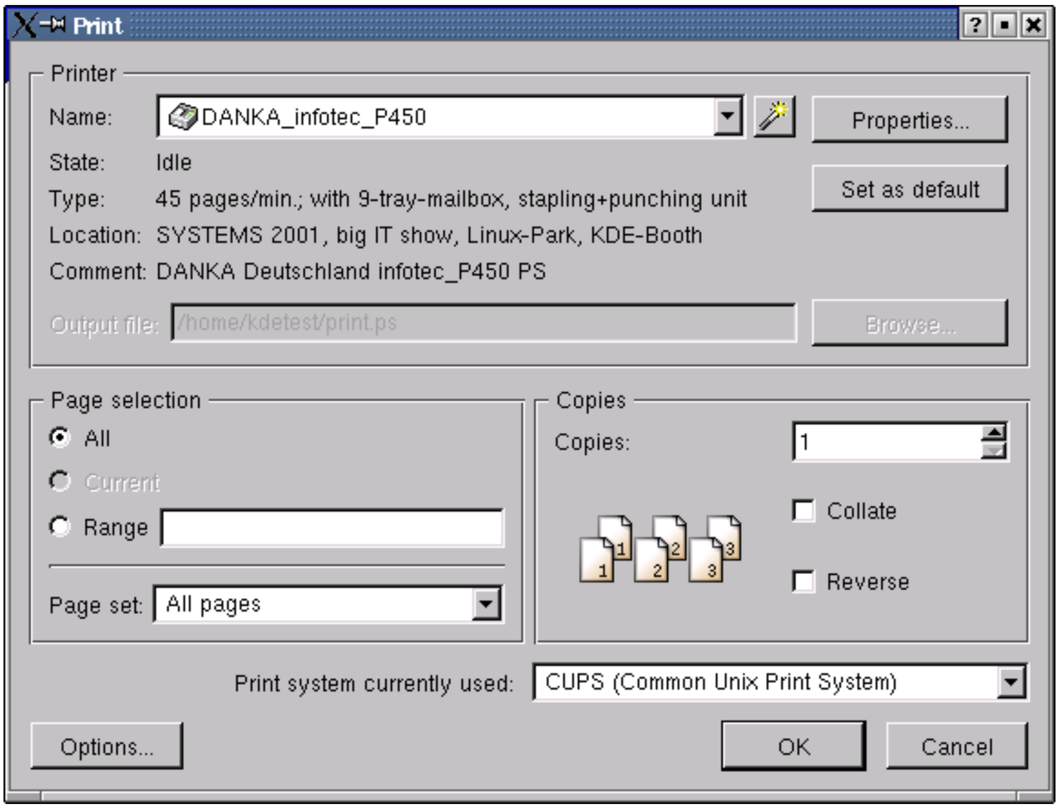
Orientierung: Portrait Landscape

Drucken: Graustufen Farbe

Papier-Grösse: Letter (8 1/2 x 11 in.) Legal (8 1/2 x 14 in.) Executive (7 1/2 x 10 in.) A4 (210 x 297 mm)

Drucken Abbrechen

You can also use **kprinter** from the commandline and see the resulting dialog box pop up:



Print

Printer

Name: DANKA_infotec_P450

State: Idle

Type: 45 pages/min.; with 9-tray-mailbox, stapling+punching unit

Location: SYSTEMS 2001, big IT show, Linux-Park, KDE-Booth

Comment: DANKA Deutschland infotec_P450 PS

Output file: /home/kdetest/print.ps

Properties... Set as default Browse...

Page selection

[✓ All] Current Range

Page set: All pages

Copies

Copies: 1

Collate

Reverse

Print system currently used: CUPS (Common Unix Print System)

Options... OK Cancel

Note

Just make sure you give at least the file to be printed from the commandline as well: **`kprinter /usr/share/doc/packages/cups/sam.pdf`**. This will hand over the CUPS Software Administrator Manual to the

kprinter dialogue, which then pops up with the default printer pre−selected.

To pre−select a specific printer from the commandline, use it like **`kprinter -d DANKAcolorC2000 /home/kurt/linuxtag2001-paper.ps`**. You can still de−select the printer `DANKAcolorC2000` and choose a different one.

You *cannot* however call **`kprinter`** without a print file and hope to open a file selection dialog box from the kprinter window. This is a feature that will be implemented only in the next version.

Through **kprinter** your are able to ring all the bells and blow all the whistles of your printer. You will need a device−specific so−called PPD (PostScript® Printer Description) to enable CUPS to make this nice tandem team do this for you. Read more about this in the section called Device Dependent Print Options .

Prev Home Next
More KDEPrint Goodies Up Plans for Future Development

Plans for Future Development

Plans for Future Development

What you have now is the first, already very feature−rich version of KDEPrint. This version is, of course, fully usable for printing. >You might even think, that it never was so easy (not even back in the dark days you had to use Microsoft® Windows®.

In the future, KDEPrint will become even better. It will do a better job of detecting your installed print subsystem itself. Already KDEPrint is doing quite well in automatically sensing if you have CUPS on your system. But in many cases you will have to tell KDEPrint what you are using, if you want to keep a legacy print system.

The most important improvement in the near future will be a completion of the LPRng plugin. This at present is still very basic. It is restricted to the pure classical LPD part of LPRng.

Also, you may be able to add printers directly from the print dialog to your system just in time , without going to KControl first.

Some smaller improvements already planned are:

- add a file selection dialog from the kprinter window to allow for combining additional files to the present printjob
- add a history button to the KJobViewer window and also a column to show the number of pages CUPS calculates for the job.

Finally, there will be an IO slave that will give you access to your print subsystem, via Konqueror for example. With this you will soon be able to browse your print subsystem from Konqueror through a URL like shortcut such as **`print://printers/printername`**. A KPart will add a virtual folder to the services section of the Konqueror sidebar, giving a nice integrated way to browse and manage your print system via the URL **`print:/manager`**.

Please contact Michael Goffioul at <`goffioul@imec.be`> with any further user or developer suggestions.

Prev Next

Chapter 4. Some Theoretical Background: CUPS, IPP, PostScript® and GhostScript

This chapter aims to give a bit of theoretical background to printing in general, and to CUPS especially. If you are not in need of this, you might like to skip ahead to the next chapter. I am betting you will come back to this chapter at some point anyway, because sometimes one needs extra theory to solve a practical problem.

Basics About Printing

Printing is one of the more complicated chapters in IT technology.

Earlier on in history, every developer of a program that was capable of spitting out printable output had to write his own printer drivers too. That was quite complicated, because different programs have different file formats. Even programs with the same usage, for example, word processors, often do not understand each others formats. So, there was no common interface to all printers, hence the programmers often supported only a few selected models.

A new device appearing on the market required the program authors to write a new driver if they wanted their program to support it. Also for manufacturers, it was impossible to make sure their device was supported by any program known to the world (although, there were far fewer than today.)

Having to support ten application programs and a dozen printers, meant a system administrator had to deal with 120 drivers. So the development of unified interfaces between programs and printers became an urgent need.

The appearance of Page Description Languages , describing the graphical representation of ink and toner on sheets of paper (or other output devices, like monitors, photo typesetters, etc.) in a common way was a move that found a big gap.

One such development was PostScript® by Adobe. It meant that an application programmer could concentrate on making his program give out a PostScript® language description of his printable page, while printing device developers could focus on making their devices PostScript® literate.

Of course, there came, over time, the development of other description methods. The most important competitors to PostScript® were PCL (Print Control Language , from Hewlett−Packard®), ESC/P (from Epson) and GDI (Graphical Device Interface from Microsoft®).

The appearance of these page description languages eased life, and facilitated further development for everybody. Yet the fact that there still remained different, incompatible, and competing page description languages keeps life for users, administrators, developers and manufacturers difficult enough.

PostScript® in memory − Bitmaps on Paper

PostScript® is most heavily used in professional printing environments such as PrePress and printing service industries. In the domains of UNIX® and Linux®, PostScript® is the pre−dominant standard as a PDL. Here, nearly every program generates a PostScript® representation of it's pages once you push the Print button. Let us look at a simple example of (hand−made) PostScript® code. The following listing describes two simple drawings:

Example 4.0. PostScript® Code

```
%!PS
100 100 moveto
0 50 rlineto
50 0 rlineto
0 −50 rlineto
closepath
.7 setgray fill
% first box over; next
160 100 moveto
```

```
0 60 rlineto
45 10 rlineto
0 −40 rlineto
closepath
.2 setgray fill
```

This tells the imaginary PostScript® pen to draw a path of a certain shape, and then fill it with different shades of gray. The first part translates into more comprehensive English as Go to coordinate (100,100), draw a line with length 50 upward; then one from there to the right, then down again, and finally close this part.Now take a paint of 70% gray, and use it to fill the drawn shape.

Example 4.1. Rendered PostScript®

Of course, PostScript® can be much more complicated than this simplistic example. It is a fully fledged programming language with many different operators and functions. You may even write PostScript® programs to compute the value of Pi, format a harddisk or write to a file. The main value and strength of PostScript® however lays in the field to describe the layout of graphical objects on a page: it also can scale, mirror, translate, transform, rotate and distort everything you can imagine on a piece of paper −− such as letters in different font representations, figures, shapes, shades, colors, lines, dots, raster...

A PostScript® file is a representation of one or more to−be−printed pages in a relatively abstract way. Ideally, it is meant to describe the pages in an device−independent way. PostScript® is not directly visible ; it only lives on the hard disks and in RAM memory as a coded representation of future printouts.

Raster Images on Paper Sheets

What you see on a piece of paper is nearly always a raster image . Even if your brain suggests to you that your eyes see a line: take a good magnifying glass and you will discover lots of small dots... (One example to the contrary are sheets that have been drawn by pen plotters). And that is the only thing what the marking engines of todays printers can put on paper: simple dots of different colors, size, resolution to make up a complete page image composed of different bitmap patterns.

Different printers need the raster image prepared in different ways. Thinking about an inkjet device: depending on its resolution, the number of used inks (the very good ones need different 7 inks, while a cheaper one might have use 3), the number of available jets (some print heads have more than 100!) spitting out ink simultaneously, the dithering algorithm used, and many other things, the final raster format and transfer order to the marking engine is heavily dependent on the exact model used.

Back in the early life of the Line Printer Daemon , printers were machines that hammered rows of ASCII text mechanically onto long media, folded as a zig−zag paper snake, drawn from cardboard boxes beneath the table... What a difference from today!

RIP: From PostScript® to Raster

Before the final raster images are put on paper cut−sheets, they have to be calculated somehow out of their abstract PostScript® representation. This is a very computing−intensive process. It is called the Raster Imaging Process , more commonly RIP).

With PostScript® printers the RIP−ping is taken care of by the device itself. You just send to it the PostScript® file. The Raster Imaging Processor (also called the RIP) inside the printer is responsible (and specialized) to fullfil quite well this task of interpreting the PostScript®−page descriptions and put the raster image on paper.

Smaller PostScript® devices have a hardware−RIP built in; it is cast in silicon, on a special chip. Big professional printers often have their RIP implemented as a software−RIP inside a dedicated fast UNIX® run computer, often a Sun SPARC Solaris or a SGI" IRIX® machine.

GhostScript as a Software RIP

But what happens, if you are not lucky enough to have a PostScript® printer available?

You need to do the RIP−ing before you send the print data the marking engine. You need to digest the PostScript® generated by your application on the host machine (the print client) itself. You need to know how the exact raster format of the target printers' marking engine must be composed.

In other words, as you can't rely on the printer to understand and interpret the PostScript® itself, the issue becomes quite a bit more complicated. You need software that tries to solve for you the issues involved.

This is exactly what the omnipresent ghostscript package is doing for many Linux®, *BSD and other UNIX® boxes that need to print to non−PostScript® printers: ghostscript is a PostScript® interpreter, a software RIP capable to run a lot of different devices.

Drivers and Filters in General

To produce rasterized bitmaps from PostScript® input, the concept of filters is used by ghostscript. There are many different filters in ghostscript, some of them specialized for a certain model of printer. ghostscript filterspecializedin devices have often been developed without the consent or support of the manufacturer concerned. Without access to the specifications and documentation, it was a very painstaking process to reverse engineer protocols and data formats.

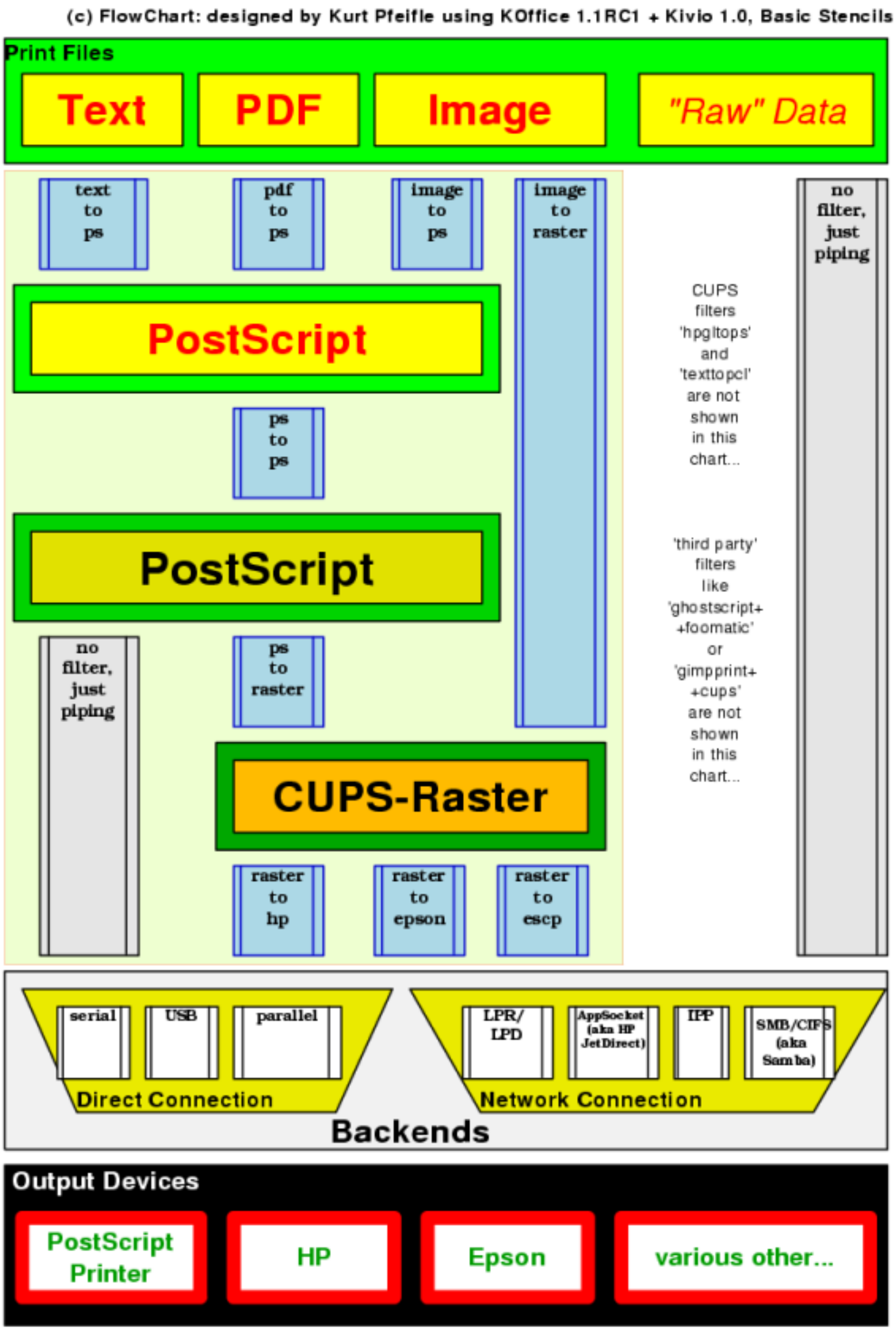
Not all ghostscript filters work evenly well for their printers. Yet, some of the newer ones, like the stp Filter of the Gimp Print project, produce excellent results leading to photographic quality on a par or even superior to their Microsoft® Windows® driver counterparts.

PostScript® is what most application programs produce for printing in UNIX® and Linux®. Filters are the true workhorses of any printing system there. Essentially they produce the right bitmaps from any PostScript® input for non−PostScript® target engines.

Drivers and Filters and Backends in CUPS

CUPS uses its own filters, though the filtering system is based on ghostscript. Namely the pstoraster and the imagetoraster filters are directly derived from ghostscript code. CUPS has re−organized and streamlined the whole mechanics of this legacy code and organized it into a few clear and distinct modules.

This next drawing (done with the help of Kivio) gives an overview of the filters and backends inside CUPS and how they fit to each other. The flow is from top to bottom. Backends are special filters: they don't convert date to a different format, but they send the ready files to the printer. There are different backends for different transfer protocols.



CUPS: "Core" Filter and Backend Architecture

Spoolers and Printing Daemons

Besides the heavy part of the filtering task to generate a print−ready bitmap, any printing software needs to use a SPOOLing mechanism: this is to line up different jobs from different users for different printers and different filters and send them accordingly to the destinations. The printing daemon takes care of all this.

This daemon is keeping the house in order: it is also responsible for the job control: users should be allowed to cancel, stop, restart etc. their jobs (but not other peoples's jobs) and so on.

Prev | Home | Next
Plans for Future Development | Up | Excursion: How CUPS uses the power of PPDs

Excursion: How CUPS uses the power of PPDs

Now that you know how a PostScript® language file (which describes the page layout in a largely device independent way) is traveling to become transformed into a Raster Image, you might ask: Well, there are different kinds of raster output devices: first they differ in their resolution; then there are the different paper sizes; it goes on with many finishing options (duplex prints, pamphlets, punched and stapled output with different sheets of colored paper being drawn from different trays, etc.). How does this fit into our model of device−independent PostScript®?

The answer comes with so called PostScript® Printer Description (PPD files. A PPD describes all the device dependent features which can be utilized by a certain printer model. It also contains the coded commands that must be used to call certain features of the device. But PPDs are no closed book, they are simple ASCII text files.

PPDs were invented by Adobe to make it easy for manufacturers to implement their own features into PostScript® printers, and at the same time retain a standard way of doing so. PPDs are well documented and described by Adobe. Their specification is a de−facto open standard.

Device Dependent Print Options

Remember, advanced PostScript® printing originally was developed for use on Microsoft® Windows® and Apple Mac® systems only. For a long time all the feature rich printing on modern devices was just unavailable for Linux® and UNIX®. CUPS changes this decisively. CUPS is very intimated with PPDs, and therefor existing PPDs can be utilized to the full by all systems powered by CUPS.

Via PPDs, printer manufacturers were able to insert device−specific hardware features into their products, for things such as duplexing, stapling, punching, finishing etc.. The printer drivers load this PPD just like an additional configuration file. Thus the printer driver learns about the available device options and how to call them; the driver also shows them in a GUI to the user. Through this mechanism you still are able to print device−independent PostScript® page description language files and specify device−dependent finishing options on top, which are added to the application−produced PostScript®.

Where to get the PPDs for PostScript® Printers

PPDs originally were not used routinely in UNIX® and Linux® systems. The vendors providing those PPDs never intended them for other than the originally supported operating systemes, Microsoft® Windows® and Mac® operating system. Through it's brilliant move to fully support and utilize the existing PPD specification, CUPS now gives the power to use all features of modern printers to users of Linux® and Linux®−like systems. KDEPrint makes it's usage even more comfortable than the CUPS developers ever dreamt of.

CUPS can use original Windows® PPDs, distributed by the vendors in the case of PostScript® printers. Those normally don't cost any money, and they can be grabbed from any Windows® computer with an installed PostScript® driver for the model concerned, or from the disks provided with the printer. There are also several places on the web to download them.

How Special PPDs are Now Useful Even For Non−PostScript® Printers.

Now you know how PostScript®−Printers can use PPDs. But what about non−PostScript® printers? CUPS has done a very good trick: by using the same format and data structure as the PostScript® Printer Descriptions (PPDs) in the PostScript® world, it can describe the available print job options for non−PostScript® printers just the same. For its own special purposes CUPS just added a few special options (namely the line which defines the filter to be used for further processing of the PostScript® file).

So, the developers could use the same software engine to parse the Printer Description Files for available options for all sorts of printers. Of course the CUPS developers could not rely on the non−PostScript® hardware manufacturers to suddenly develop PPDs. They had to do the difficult start themselves and write them from scratch. More than 1000 of these are available through the commercial version of CUPS, called ESP PrintPro.

Meanwhile there are a lot of CUPS−specific PPDs available. Even now those are in most cases not originating from the printer manufacturers, but from Free software developers. The CUPS folks proofed it, and others followed suit: where Linux® and UNIX® printing one or two years ago still was a kludge, it is now able to support a big range of printers, including 7−color inkjets capable of pushing them to Photo Quality output.

Different Ways to get PPDs for non−PostScript® Printers

You can get PPDs to be used with CUPS and non−PostScript® printers from different areas in the Web:

- first, there is the repository at www.linuxprinting.org, which lets you generate a CUPS−O−Matic −PPD online for any printer that had been supported by traditional ghostscript printing already. This helps you to switch over to CUPS with little effort, if you wish so. If your printer was doing well with the traditional way of ghostscript printing, take CUPS−O−Matic to plug your driver into th e CUPS system and you'll have the best of both worlds.
- second, there are CUPS−PPDs for the more than 120 printer models, which are driven by the new universal stp driver. stp (stood originally for Stylus Photo) is now developed by the gimp−print project; it was started by Mike Sweet, the leading CUPS developer and is now available through gimp−print.sourceforge.net. This driver prints real Photo quality on many modern inkjets and can be configured to make 120 CUPS−PPDs along its own compilation. HP® Laser− and DeskJet, Epson® Stylus and Photo Color models as well as some Canon® and Lexmark® are covered.
- third, there is the commercial extension to CUPS from the CUPS developers themselves: it is called ESP PrintPro and comes with more than 2.300 printer drivers. There are even improved imagetoraster and pstoraster filters included.

CUPS makes it really easy for manufacturers to start supporting Linux® and UNIX® printing for their models at reasonably low cost. The modular framework of CUPS facilitates to plug in any filter (=driver) with minimal effort and to access and utilize the whole printing framework that CUPS is creating.

Read more about the exciting CUPS features in the available CUPS documentation at http://www.cups.org/documentation.html and http://www.danka.de/printpro/faq.html. Also at http://www.linuxprinting.org/ is a universal repository for all issues related to Linux® and UNIX® printing.

Prev
Some Theoretical Background: CUPS, IPP, PostScript® and GhostScript

Home
Up

Next
How IPP Support Makes CUPS the Best Choice Around

How IPP Support Makes CUPS the Best Choice Around

How IPP Support Makes CUPS the Best Choice Around

LPD Must Die!

For a long time many developers were deeply dissatisfied with good old LPD. Quite a few new projects were started to improve printing: LPRng is the best known example. Others are PDQ, PPR, PLP, GNUlpr and RLPR. but none of the new programs were seen as a big shot ; most of them are just implementing the same old LPD specification with a few (or many) new extensions, which again make them incompatible to each other.

Having seen the development of not just one, but different viable alternatives to venerable BSD−style LPD, Grant Taylor, author of the *Linux Printing HOWTO,* finally rallied the call *LPD Must Die!* in his Campaign To Abolish The Line Printer Daemon .

How the IPP Came to Be

Along with the above, on the industry side of things, there were efforts to overcome the well−known weaknesses of LPD. It started with proprietary extensions to plain old LPD, and stretched as far as Hewlett−Packard®'s attempt to establish HP® JetDirect as a new standard for a network printing protocol. The result were even more incompatibilities.

In the end, an initiative to define a new common industry and IETF standard took shape. The Printer Working Group or PWG, a loose aggregation of vendors in hardware, software, and operating systems, drafted the new Internet Printing Protocol , IPP. IPP v1.1 has now been approved by the IETF (Internet Engineering Task Force) as a proposed standard, and now enjoys the unanimous support throughout the industry in Europe, USA and Japan. Most current network printer models have now built in IPP support on top of traditional LPR/LPD or JetDirect Printing.

Why IPP is Solving Many Problems

IPP promises to solve a lot of problems network administrators face. This trade normally deals with heterogeneous network environments and spends more than half of its working hours dealing with printing problems.

By creating a unified set of query functions for IPP enabled printers and servers, for transferring files and setting job−control attributes etc., IPP is destined to work across all operating system platforms. It's rollout however, will not happen overnight, as many legacy print devices will still be in use for many years to come. Therefore, in IPP there is a provision made for backwards compatibility of all IPP implementations. CUPS is proving the viability of IPP printing in all environments.

The most striking advantage will be it's integration into the existing set of other robust IP protocols. Being an extension of the proven and robust HTTP 1.1 protocol, for the special task of handling print file and related data, it is also very easy to plug in other standards as they are being developed and deployed:

- Basic, Digest, and Certificate Authentication for users seeking access to print services.
- SSL3 and TLS encryption for transferring data.
- Bi directional communication of clients with print devices, using the HTTP/IPP **GET** and **POST** mechanism.
- LDAP directory service integration to keep a consistent database of available printers, their capabilities and page−costs, etc., as well as user passwords, ACLs etc..
- Pull (as opposed to the usual Push model) printing, where a server or printer just needs to be told the URL of a document, whereupon it is retrieved from the resource on the internet and printed.

Printer Plug'n'Play for Clients

Have you ever seen a demonstration about CUPS capabilities in the network? You must have been quite impressed if you didn't know in advance what to expect.

Imagine you as the administrator of a LAN . For testing purposes you fully installed one KDE/CUPS box on your net, complete with a dozen printers configured and functional: PostScript®, LaserJets, InkJets and BubbleJets, and so on. Your KDE users on that box are very happy, they can print like never before, ringing all the bells and whistles of every printer. It took you 2 hours to make everything run perfectly... and now all the other 100 users on the network want the same. Two hours again for every box? No way you could do that before next year, you think?

Wrong. Just change one setting in the original CUPS box to make it a server . Install CUPS on five other boxes, as clients . By the time you turn back to your first client, you find the users happily playing with the settings for the dozen printers you had defined earlier on the server . Somehow magically the printers had appeared on all the Print dialogs of the five new CUPS client boxes.

Your users print, but nota single driver had been installed on the clients, nor a printer queue defined.

So, how does this magic work?

Seeing Printers Not Installed Locally?

The answer is not complicated at all.

If a CUPS server is on the LAN, it broadcasts the names of all available printers to the LAN, using the UDP protocol and port 631. Port 631 is reserved as a well−known port by IANA (the Internet Assigning Numbers Authority) for IPP purposes. All CUPS clients listen to CUPS server info sent to their port 631. That's how they know about available printers, and that's how they learn about the path to the printers as well.

Using IPP, which is really a clever extension to HTTP v1.1, CUPS is able to address all objects related to the printing system via Universal Resource Locators or URLs. Print jobs to be deleted or restarted, printers to be queried or modified, admin tasks to be performed on the server, with IPP and CUPS, everything is addressable by a certain URL. Many important things can be done through the Web interface to CUPS,accessible for example with Konqueror.

Printing Without Installing a Driver

And more, the clients basically can administer and use any printer they see, just as if it was a locally installed one. Of course, you can set restrictions on it with access control lists etc., so that not *any* clients may use *any* printer as it likes.

The clients even are able to print without the appropriate filter (or driver) installed locally.

So how does this work? If a client wants to know about and select printer−specific options, it sends a request (called **CUPS−get−ppd**) to the server. The server tells the client all about all printer−specific options, as read from the server side PPD. The user on the client side can see the options and select the required ones. He then sends the print file, usually unfiltered raw PostScript®, spiced up with the printer−options to the printer server, using IPP as the transport protocol. All further processing, especially the filtering to generate the final format for the target printer, is then done by the server. The server has the necessary programs (drivers or filters) to do this.

This way a client prints without needing to install a driver locally.

Any change on the server, such as adding or modifying a printer, is instantly known to the clients with no further configuration.

Zero Administration , Load Balancing, and Failover Switching

Some other advanced features built into CUPS are the capacity to do load balancing .

If you define the same printer queues on two or more different servers, the clients will send their jobs to the first responding or available server. This implies an automatic load balancing amongst servers. If you have to take off the net one server for admin maintainance the others will just take his tasks without the users even noticing the difference.

Prev
Excursion: How CUPS uses the power of PPDs

Home
Up

Next
Getting Started

Chapter 5. Getting Started

This chapter of the KDEPrint Handbook will walk you through most of the configuration or selection options of KDEPrint. It will mainly deal with CUPS in this version, as the author is most familiar with it, and also because KDEPrint started off with supporting CUPS best. Later versions of the KDEPrint software and editions of this handbook will support and explore more closely other printing systems.

Selecting Your Print Subsystem

You need to define your print subsystem, before you are able to install any printer with the KDEPrint framework. There are two areas where you can define this: either in KControl (The Printing Manager section), or directly and on the fly from the print dialog.

Navigate to K Menu−>Preferences−>System−>Printing Manager. At the bottom you can see a button that lets you select which printing subsystem you want to use. In KDE 2.2 you can choose from the following alternatives:

- CUPS (Common UNIX® Printing System)
- Print through an external program (generic)
- LPR (Standard BSD Print System)
- Generic UNIX® LPD print system (the default)
- RLPR environment (print to remote LPD servers from commandline.

Of course, the chosen system must be installed, up and running on your box prior to your selection, or before it takes effect.

On it's first startup, KDEPrint will try an autodetection. This only works for:

- CUPS, as it is checking first for a running CUPS daemon
- LPD, as it is checking for a running LPD daemon, plus a `printcap` file.

The system you choose must be installed on your system prior to your selection. The author's personal recommendation is CUPS.

Once autodetected, chosen, or changed, the active print subsystem will take effect for all KDE applications. Different users may have different print subsystems in use, if those do exist on the computer and are compliant to each other. Their settings are stored in the `kdeprintrc`. This file is unique to every user, and is normally installed in `$HOME/.kde/share/config/kdeprintrc`.

Warning

This file is not intended to be directly editable, and all available options can be set from the KDEPrint GUI.

You may even switch printer subsystem in use on the fly, from the **kprinter** dialog box.

Prev
How IPP Support Makes CUPS the Best Choice Around

Home
Up

Next
Working with the Printing Manager

Working with the Printing Manager

Working with the Printing Manager

Once you have chosen your preferred and installed print subsystem, you are ready to investigate, configure administer and work with this system through the KDEPrint framework.

Navigate to K Menu−>Preferences−>System−>Printing Manager. In the right part of the window you will see at least 4 printers predefined. These are the virtual or special purpose printers, explained in section . You will probably see a toolbar with 13 icons at the top of the window, and at least 4 tabs in the lower half of the window, labelled Information, Jobs, Properties and Instances.

Chapter 6. Print Server Configuration: CUPS

Start the print server configuration (now that you have chosen CUPS, this is equivalent to the configuration of the CUPS daemon) by clicking on the appropriate button. You can find it by moving the mouse slowly over the buttons and reading the tooltips. It should be the 11th from the left , or third from the right; its icon is a wrench.

The CUPS Server Configuration window pops up. It gives you a structured view of all the settings that apply to the CUPS daemon. The configuration file for that daemon is normally located in `/etc/cups/cupsd.conf`. This is a plain ASCII file with a syntax similar to the configuration file of the Apache web server. It is a good idea to create a backup copy, just in case something goes wrong with the configuration through KDEPrint/CUPS Server Configuration dialogs:

```
cp /etc/cups/cupsd.conf /etc/cups/cupsd.conf.bak
```

As this graphical user interface to edit the configuration file is such a new feature, you should have the second chance of resorting to the original file. So back it up, please.

Quick Help

One very nice feature is the Quick Help available. If you click on the little question mark (What's this?) on your window title bar, you'll see the cursor changing its form. Now click on a **cupsd** configuration setting field to find out what it means and what your options are. In most cases you should understand the meaning immediately, otherwise turn to the excellent CUPS documentation. (If your CUPS Daemon is running, you have it online on your own host at http://localhost:631/documentation.html.

If CUPS is not running, but installed on your system you could find it in your own host's file system. The exact location depends on your operating system, but on Linux® the default is `/usr/share/doc/cups/` or `/usr/share/doc/cups/documentation.html`.

Longer Help

Longer Help

For the best, most detailed and most recent information you should always refer to the original CUPS documentation. CUPS is, much like KDE in a rapid development process. There are constantly new features being added. New features might for times be only configurable by directly editing the configuration files. The KDEPrint GUI might not have caught up with CUPS development.

Just in case you want to look at the original configuration files of your CUPS system -- they are here:

Note

These paths are based on the default installation. Your operating system may have installed them to a different prefix, for example, `/usr/local/`, but the hierarchy should still match that shown below.

`/etc/cups/`
: The directory with the configuration files

`/etc/cups/cupsd.conf`
: The configuration file for the CUPS daemon

`/etc/cups/printers.conf`
: The configuration file that contains the information about your locally installed printers.

`/etc/cups/ppd/`
: The directory with PPD files of your installed printers.

The following links only work if your CUPS daemon is up and running. To access all the original CUPS documentation, go to:

http://localhost:631/documentation.html
: A page with all the links to the other documents.

http://localhost:631/sam.html
: Direct access to the CUPS Software Administrator Manual in HTML format.

http://localhost:631/sam.pdf
: Direct access to the CUPS Software Administrator Manual in PDF format.

http://www.cups.org/documentaiton.html
: The latest on line documentation from the CUPS web site.

The following links give you access to the same files (probably icons and grapphis will be missing) even if your CUPS daemon is not up and running. You need, however, CUPS installed on your system. (Some distributions might place the files somewhere else -- you're on your own then to find out where...) To access all the original CUPS documentation, go to:

This documentation is available even when the CUPS daemon is not installed, although you may find images and icons are missing when you view the HTML files.

As noted above, the hierarchy below should be intact, but your operating system may have installed CUPS to a different location.

`/usr/share/doc/cups/documentation.html`
: A page with all the links to the other documents.

`/usr/share/doc/cups/sam.html`
: Direct access to the CUPS Software Administrator Manual in HTML format.

`/usr/share/doc/cups/sam.pdf`
: Direct access to the CUPS Software Administrator Manual in PDF format.

There are a few WebSites and Newsgroups discussing CUPS (and Linux® Printing in General) and giving help to newbies at:

http://www.cups.org/newsgroups.php
    The CUPS website.

http://www.linuxprinting.org/newsportal/
    LinuxPrinting.org, the home of the Linuxprinting HOWTO and the Linux® Printer Database

And finally, there will be a WebSite for KDEPrint and related documentation, at http://kdeprint.sourceforge.net/

In the next section I will step you through most of the configuration options of KDEPrint with CUPS.

Prev
Print Server Configuration: CUPS
Home
Up
Next
Explaining different elements of the GUI

Explaining different elements of the GUI

Explaining different elements of the GUI

Upper Window: View on Printers, both Real and Virtual

This section is not yet complete

- Tree view, icon view and list view
- The icons of the task bar
- Different fonts for different printers
- Different printer icons mean different things

Lower Window: Tabbed View of Details

This section is not yet complete.

- The icons of the task bar
- The Tabs
- Changing printer settings

Prev Home Next
Longer Help Up Welcome to the CUPS Server Configuration

Welcome to the CUPS Server Configuration

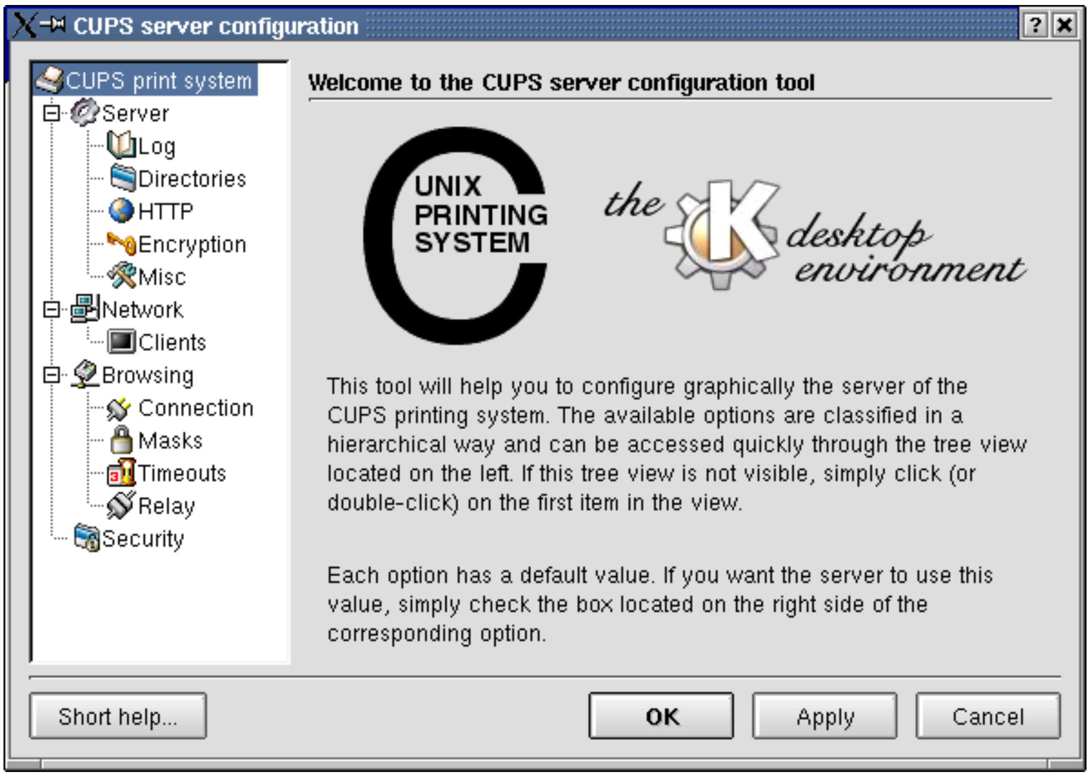
Welcome to the CUPS Server Configuration

This is the Welcome Screen for your server configuration dialogues. Clicking onto one of the items of the tree view on left side of the screen opens the appropriate part of the configuration settings.

Every setting has a default value. The defaults let CUPS normally work as a fully functional client. The clients listen on TCP/IP Port 631 for infos broadcast by CUPS servers on the LAN. This information let the clients print immediately after receiving them, without installing any driver or configuring any printer on the clients.

To configure a CUPS server (which is broadcasting its service to the LAN) you need to change settings from the defaults.

The dialog to configure the CUPS server: welcome screen.



Welcome to the CUPS server configuration tool

This tool will help you to configure graphically the server of the CUPS printing system. The available options are classified in a hierarchical way and can be accessed quickly through the tree view located on the left. If this tree view is not visible, simply click (or double-click) on the first item in the view.

Each option has a default value. If you want the server to use this value, simply check the box located on the right side of the corresponding option.

The dialog to configure the CUPS server: welcome screen

To select the default setting of any item just enable the checkbox on the right side of the screen. To set an item to a different value, disable the checkbox and then go on to do the setting you want on the left side of the screen.

The complete server configuration includes:

- Server General Configuration
- Server Logging Configuration
- Server Directories and Path Definitions
-

- Server HTTP Configuration
- Server Encryption and Certificate Support Configuration
- Server Miscellaneous Configuration
- Network General Configuration
- Network Clients Configuration
- Browsing General Configuration
- Browsing Connection Configuration
- Browsing Masks Configuration
- Browsing Timeouts Configuration
- Browsing Relay Configuration
- Security Configuration

Each of these configuration items will be described in the following sections of the manual.

Prev Home Next
Explaining different elements of the GUI Up Server General Configuration

Server General Configuration

Server General Configuration

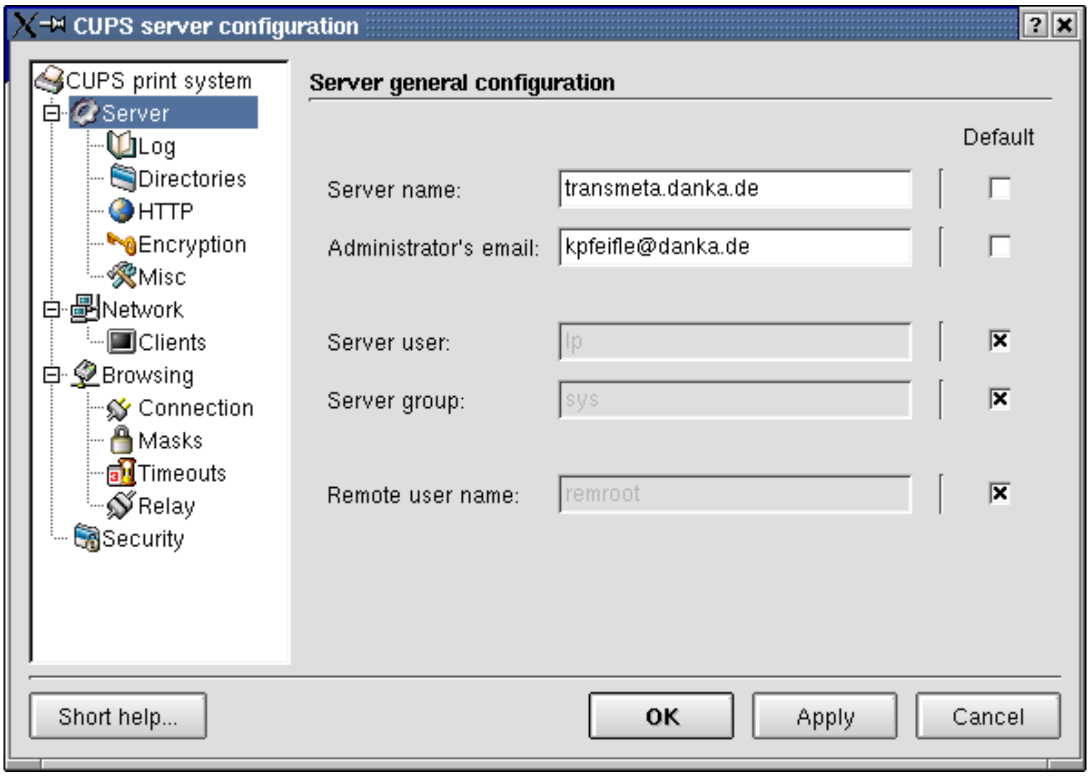
The server general configuration is done on this screen. It includes:

- Server name
- Administrators email
- Server user
- Server group
- Remote user name

The tab window to configure the CUPS server general settings lets you the change the default values. Click on the little question mark and then on one of the fields to get a Quick Help about the meaning of the setting.

If you are unsure, leave alone and turn to the original CUPS documentation first. If your CUPS daemon is already running, it is readable from the Konqueror by pointing it to URL http://localhost:631/documentation.html.

There, first make friends with the Software Adminstrator Manual. Otherwise, for example, if the CUPS daemon is not running, try looking in your local file system, by default at `/usr/share/doc/cups/` or `/usr/share/doc/cups/documentation.html`.



Server general configuration

		Default
Server name:	transmeta.danka.de	☐
Administrator's email:	kpfeifle@danka.de	☐
Server user:	lp	☒
Server group:	sys	☒
Remote user name:	remroot	☒

Server Name

The hostname of your server, as advertised to the world. By default, CUPS will use the hostname of the system. To set the default server usd by clients, see the `client.conf` file.

For example, enter **`myhost.domain.com`**

This is the hostname that is reported to clients. Should you ever encounter strange problems in accessing the server, put here its IP address for troubleshooting. This way you eliminate any potential name resolution problems; and you can more easily nail the real problem down.

Administrators email

This is the email address to send all complaints or problems to. By default CUPS will use root@hostname .

For example, enter **`root@myhost.com`**.

Note

Contrary to what the quickhelp suggests, it is also legal to send an email full of praise and enthusiasm about CUPS and KDEPrint to the server adminstrator.

Server User

The user the server runs under. Normally this must be `lp`, however you can configure things for another user if needed.

Note

The server must be initially run as root to support the default IPP port of 631. It changes users whenever an external program is run.

Enter for example **`lp`**.

This is the UNIX® user account for filters and CGI programs to run under. CGI programs are resposible for showing you the nice web administration interface accessible via http://localhost:631/).

Warning

There is no need to set the User directive to `root`, so never do this, as it only involves dangers. Should anyone discover security vulnerabilities in one of the used file filters, printer drivers or CGI programs, he could remotely execute arbitrary commands on your system with root user privileges. Always use an unprivileged account for the server directive User.

Server group

The group the server runs under. Normally this must be `sys`, however you can configure things for another group as needed.

Enter for example **`sys`**.

Remote user name

The name of the user assigned to unauthenticated accesses from remote systems. By default **`remroot`**.

This name will appear in log files and in queries about the job owner etc., for all resources and locations of the CUPS server that are configured to allow access *without* authentication. Authenticated entries will carry the authenticated names.

Prev
Welcome to the CUPS Server Configuration

Home
Up

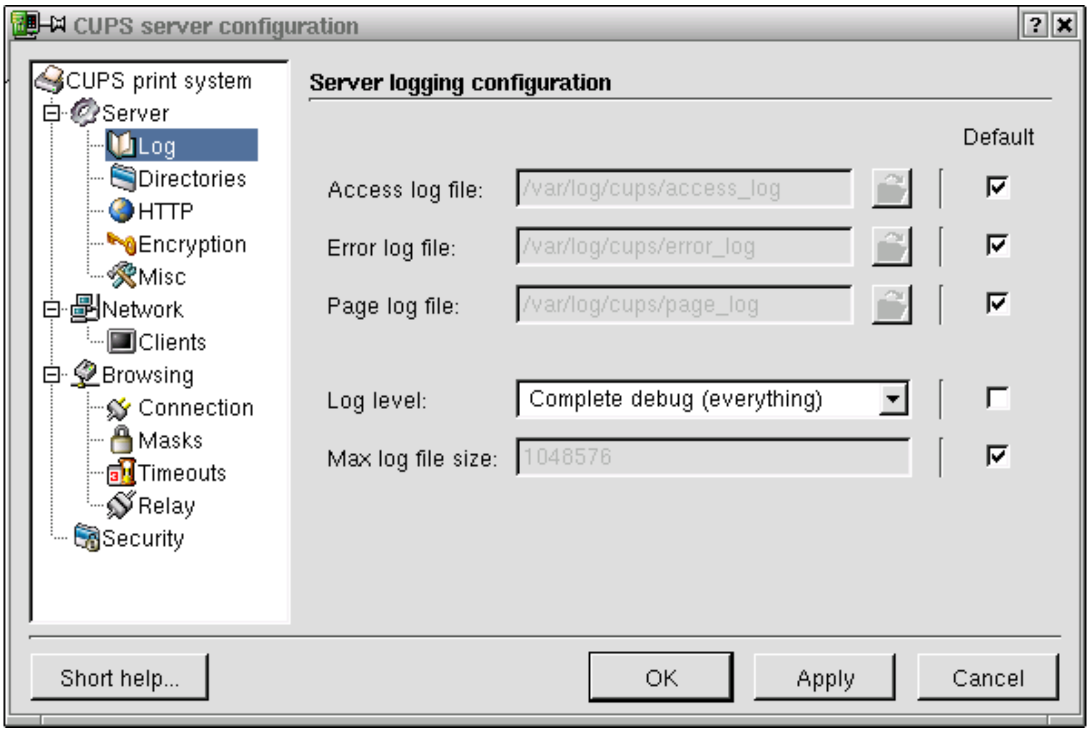
Next
Server Logging Configuration

Server Logging Configuration

The server logging configuration is done on this screen. It includes:

- Access log file setting
- Error log file setting
- Page log file setting
- Log level setting
- Max log file size setting

This is an important screen for you. Should you ever encounter problems: here is the place to set the Log level to debug , restart the CUPS daemon and then look at the Error log file defined here for entries that might give you an insight to the trouble.



Server logging configuration

		Default
Access log file:	/var/log/cups/access_log	✓
Error log file:	/var/log/cups/error_log	✓
Page log file:	/var/log/cups/page_log	✓
Log level:	Complete debug (everything)	
Max log file size:	1048576	✓

Access log file

This is where accesses to the server are logged. If this does not start with a leading `/`, then it is assumed to be relative to the server root.

You can also use the special name **`syslog`** to send the output to the syslog file or daemon.

Enter a path, for example **`/var/log/cups/acces_log`**.

The format of this file is stored in the so−called Common Log Format . This way you can use programs such as Webalyzer or any other Web access reporting tool to generate reports on the CUPS server activities.

To include the server name in the file name use a %s in the name. Example: **`/var/log/cups/access_log-%s`**.

```
kurt@transmeta:~ >tail /var/log/cups/access_log

127.0.0.1    - - [04/Aug/2001:20:11:39 +0100] "POST /printers/ HTTP/1.1" 200 109
127.0.0.1    - - [04/Aug/2001:20:11:39 +0100] "POST /admin/ HTTP/1.1" 401 0
127.0.0.1    - - [04/Aug/2001:20:11:39 +0100] "POST / HTTP/1.1" 200 210
127.0.0.1    - - [04/Aug/2001:20:11:39 +0100] "GET /ppd/DANKA_P450.ppd HTTP/1.1" 200 51021
127.0.0.1    - - [04/Aug/2001:20:11:39 +0100] "POST /jobs/ HTTP/1.1" 200 246
10.160.16.45 - - [04/Aug/2001:20:11:39 +0100] "GET /printers/DANKA_P450 HTTP/1.0" 200 0
127.0.0.1    - - [04/Aug/2001:20:11:39 +0100] "POST / HTTP/1.1" 200 80
127.0.0.1    - - [04/Aug/2001:20:11:39 +0100] "POST / HTTP/1.1" 200 139
10.160.16.45 - - [04/Aug/2001:20:11:40 +0100] "GET /cups.css HTTP/1.0" 200 198
127.0.0.1    - - [04/Aug/2001:20:11:40 +0100] "POST / HTTP/1.1" 200 139
10.160.16.45 - - [04/Aug/2001:20:11:39 +0100] "GET /printers/DANKA_P450 HTTP/1.0" 200 7319
10.160.16.45 - - [04/Aug/2001:20:11:40 +0100] "GET /images/title-logo.gif HTTP/1.0" 200 5729
```

You see a seperate line for each single access, showing the IP address of the accessing client, date and time of access, method of access (**POST** or **GET**), the requested ressource, the HTTP version used by the client, status code and the number of transferred bytes. Status code 200 means successful−OK the 401 in the above example was an unauthorized access which was denied. For a detailed explanation of the log format go to the CUPS Software Adminstrator Manual.

Error log file

If this does not start with a leading `/`, then it is assumed to be relative to the server root. The default setting is `/var/log/cups/error_log`.

You can also use the special name **`syslog`** to send the output to the syslog file or daemon.

Enter the path, for example **`/var/log/cups/error_log`**.

The error log excerpt below shows you the part logged for printing the test page with the default setting of Log level to info . For an explanation of the Log Level setting see further below.

```
kurt@transmeta:~ > tail  /var/log/cups/error_log

I [04/Aug/2001:23:15:10 +0100] Job 213 queued on 'DANKA_P450' by 'root'
I [04/Aug/2001:23:15:10 +0100] Started filter /usr/lib/cups/filter/pstops (PID 18891) for job 213.
I [04/Aug/2001:23:15:10 +0100] Started backend /usr/lib/cups/backend/lpd (PID 18892) for job 213.
```

Page log file

If this does not start with a leading `/` then it is assumed to be relative to the server root. The default is `/var/log/cups/page_log`

You can also use the special name **`syslog`** to send the output to the syslog file or daemon.

Enter the path, for example **`/var/log/cups/page_log`**.

The page log file has a line for every single page of every job printed.

Here is what some entries look like:

```
kurt@transmeta:~ > tail  /var/log/cups/page_log

GIMP_print_stp_HP kdetest 201 [03/Aug/2001:03:18:03 +0100] 4 1
GIMP_print_stp_HP kdetest 201 [03/Aug/2001:03:18:03 +0100] 5 1
GIMP_print_stp_HP kdetest 202 [03/Aug/2001:11:46:49 +0100] 1 1
GIMP_print_stp_HP kdetest 203 [03/Aug/2001:11:46:54 +0100] 1 1
DANKA_infotec_P450 kurt 204 [04/Aug/2001:03:29:00 +0100] 1 33
DANKA_infotec_P450 kurt 204 [04/Aug/2001:03:29:00 +0100] 2 33
DANKA_infotec_P450 kurt 204 [04/Aug/2001:03:29:00 +0100] 3 33
DANKA_infotec_P450 kurt 204 [04/Aug/2001:03:29:00 +0100] 4 33
DANKA_infotec_P450 root 205 [04/Aug/2001:19:12:34 +0100] 1 14
DANKA_infotec_P450 root 206 [04/Aug/2001:19:15:20 +0100] 1 1
```

In this excerpt of the file you find information on the name of the printers (`GIMP_print_stp_HP` and `DANKA_infotec_P450`) used through this server, the user names (`kdetest`, `kurt` and `root`), the job−IDs (Â01 to Â05), time of printing, page number inside the job and the number of copies for the pages. For example, job−ID 204 had 4 pages and 33 copies printed, job−ID 205 had 14 copies of just 1 page) .

Note

CUPS is dependent (for its calculation of the number of pages in a job) on passing the PostScript® through the pstops filter. See the Kivio Flowchart on the CUPS filter architecture for an idea about were this filter fits into the whole printing process). More, **pstops** depends for the counting on a DSC conforming (DSC is Document Structuring Conventions, a standard defined by Adobe) to be sent by the client. In most cases this is working.

However, this page accounting does not work for any raw printer queues (as those, by definition, don't use any filtering on the CUPS host and are by−passing **pstops**.) Every job going through a raw queue is counted as a 1−page−job (with possibly multipe copies). This is especially true for all Jobs send from Microsoft® Windows® clients via Samba to the CUPS server, as those jobs are already arriving in the correct format for the printer, because the clients use the original printer driver.

Note

I am still looking for someone who will write a nice CUPS page log analysing tool. It should generate a report with a graphical output similar to the Webalizer's access log reports. This way you could have nice statistics to be used for accounting about usage of printers, load dependent on daytime or weekday, users etc. Anyone?

Log level
This setting controls the number of messages logged to the error log file. It can be one of the following:

debug2
Log everything.

debug
Log almost everything.

info
Log all requests and state changes.

warn
Log errors and warnings.

error
Log only errors.

none
Log nothing.

If you need to troubleshoot (or if you want to study the inner workings of CUPS), set the log level to debug or debug2. Then the error_log will have a lot more entries (not just errors, but also informational entries).

You can use this to watch live what CUPS is doing when you send a print job. In a Konsole type:

```
kurt@transmeta:~ >tail -f -n100 /var/log/cups/error_log
```

This will give you the last 100 lines (`-n 100`) of the file onto the screen and a realtime update (`-f`)of what is happening. The following listing shows the printing of a test page (some pieces have been cut off for space reasons... Try it yourself if you need more info):

```
I [04/Aug/2001:23:15:12 +0100] Job 214 queued on 'DANKA_P450' by 'root'
D [04/Aug/2001:23:15:12 +0100] StartJob(214, 08426fe0)
D [04/Aug/2001:23:15:12 +0100] StartJob() id = 214, file = 0/1
D [04/Aug/2001:23:15:12 +0100] job-sheets=none,none
D [04/Aug/2001:23:15:12 +0100] banner_page = 0
D [04/Aug/2001:23:15:12 +0100] StartJob: argv = "DANKA_P450","214","root","KDE Print Test",
[....]
D [04/Aug/2001:23:15:12 +0100] StartJob: envp = "PATH=/usr/lib/cups/filter:/bin:/usr/bin", [....]
D [04/Aug/2001:23:15:12 +0100] StartJob: statusfds = 5, 6
D [04/Aug/2001:23:15:12 +0100] StartJob: filterfds[1] = 7, -1
```

```
D [04/Aug/2001:23:15:12 +0100] StartJob: filter = "/usr/lib/cups/filter/pstops"
D [04/Aug/2001:23:15:12 +0100] StartJob: filterfds[0] = 8, 9
D [04/Aug/2001:23:15:12 +0100] start_process("/usr/lib/cups/filter/pstops", [....]
I [04/Aug/2001:23:15:12 +0100] Started filter /usr/lib/cups/filter/pstops (PID 18991) for job 214.
D [04/Aug/2001:23:15:12 +0100] StartJob: backend = "/usr/lib/cups/backend/lpd"
D [04/Aug/2001:23:15:12 +0100] StartJob: filterfds[1] = -1, 7
D [04/Aug/2001:23:15:12 +0100] start_process("/usr/lib/cups/backend/lpd", [....]
I [04/Aug/2001:23:15:12 +0100] Started backend /usr/lib/cups/backend/lpd (PID 18992) for job 214.
D [04/Aug/2001:23:15:12 +0100] Page = 595x842; 15,16 to 580,833 [....]
```

The lines tagged D at the beginning are debug level entries, the ones tagged I are there in info level.

Max log file size
Controls the maximum size of each log file before they are rotated. Defaults to 1048576 (1 Mb). Set this to 0 to disable log rotation.

Enter an size in bytes, for example **`1048576`**

Prev
Server General Configuration

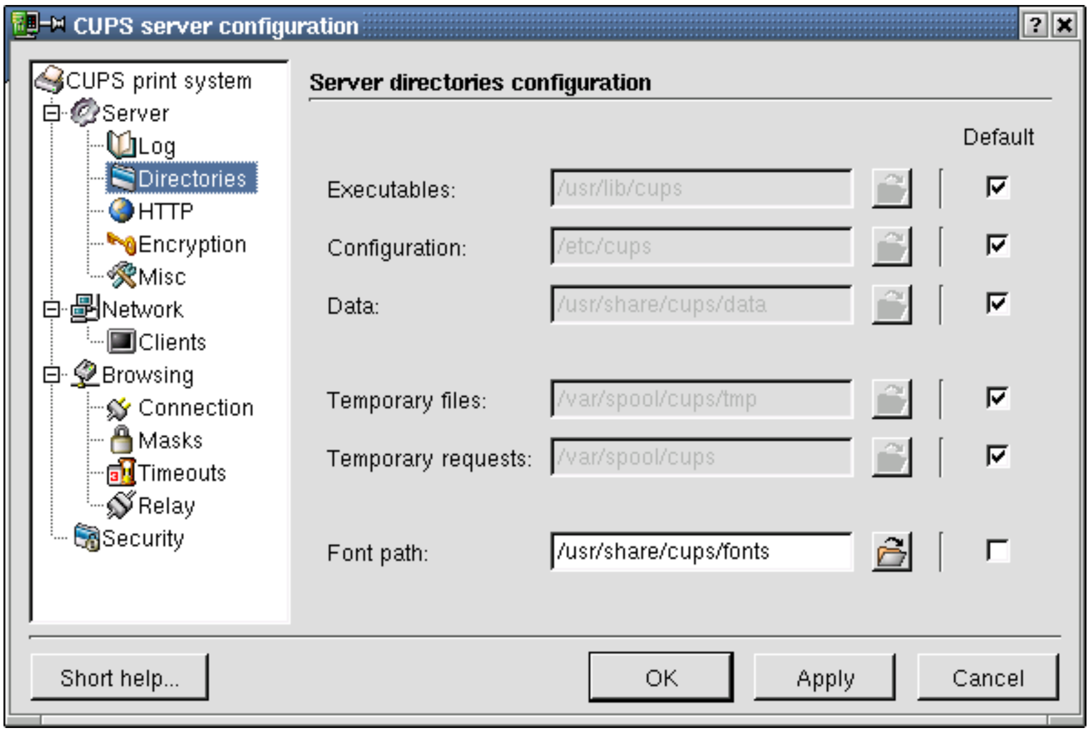
Home
Up

Next
Server Directories Configuration

Server Directories Configuration

The dialog to configure the CUPS server. Different directories are to be set here. Normally you don't need to change anything in this section. In case you play around with fancy (TrueType, PostScript® or other) fonts on your system, this qis the place to do the settings for using those fonts when printing. Server directory settings include:

- Executables: where to find the server executables
- Configuration: where to find the server configuration files
- Data: where to find the server data files
- Temporary files: where to put the server temporary print files
- Temporary Requests: where to find the server
- Font Path: where to find the server fonts



		Default
Executables:	/usr/lib/cups	✓
Configuration:	/etc/cups	✓
Data:	/usr/share/cups/data	✓
Temporary files:	/var/spool/cups/tmp	✓
Temporary requests:	/var/spool/cups	✓
Font path:	/usr/share/cups/fonts	☐

Executables
: The root directory for the scheduler executables. By default this is `/usr/lib/cups` (or `/usr/lib32/cups` on IRIX 6.5)

Configuration
: The root directory for the scheduler. By default, `/etc/cups`.

On the authors SuSE system, this is `/usr/share/doc/cups`. It contains all the HTML or PDF documentation for CUPS which is available through the Web interface at http://localhost:631/documentation.html

Data

The root directory for the CUPS data files. By default this is `/usr/share/cups`

It contains such things as banners, charsets, data, drivers, fonts, and **pstoraster** templates.

Temporary files

The directory to put temporary files in. This directory must be writable by the user defined on the previous screen. This defaults to either `/var/spool/cups/tmp` or the value of the `TMPDIR` environment variable.

Temporary Requests

The directory where request files are stored. By default this is `/var/spool/cups`

Font path

The place to configure the CUPS server for handling your fancy fonts (TrueType or PostScript®). CUPS will look here for fonts to embed in printfiles. This currently only affects the **pstoraster** filter, and the default is `/usr/share/cups/fonts`.

To specifiy more than one directory, list them with double colons as separator. Do it like this:

```
/path/to/first/fontdir/:/path/to/second/fontdir/:/path/to/last/fontdir/
```

For the Font path directive to work as intended, the application that wants to print needs to:

- Either correctly reference its desired fonts in the header of the generated PostScript®
- Or embed the font into the PostScript® file.

Referencing the font by name leaves it up to the RIP and print device to respect and actually use it. RIP or printer *can* only use the desired font, if it is available on the system.

In the case of a PostScript® printer, this needs to be a printer−resident font. If the printers doesn't have this font, it will try and replace it by an adequately similar font.

In the case of a non PostScript® printer, this is done by CUPS and its RIP−ing filtering system. CUPS will use the font path directive to grab the correct font when RIP−ing the PostScript® in the **pstoraster** filter.

In the case of a PostScript® output device, CUPS is just spooling the file (actually, it is passing it through the **pstops** filter for accounting or n−up purposes), not working on it. Therefore, if you print to a PostScript® printer it is solely the printer's responsibility to use the font asked for. It can't, if the font is neither loaded into the printer nor embedded in the PostScript®.

Prev Home Next
Server Logging Configuration Up Server HTTP Configuration

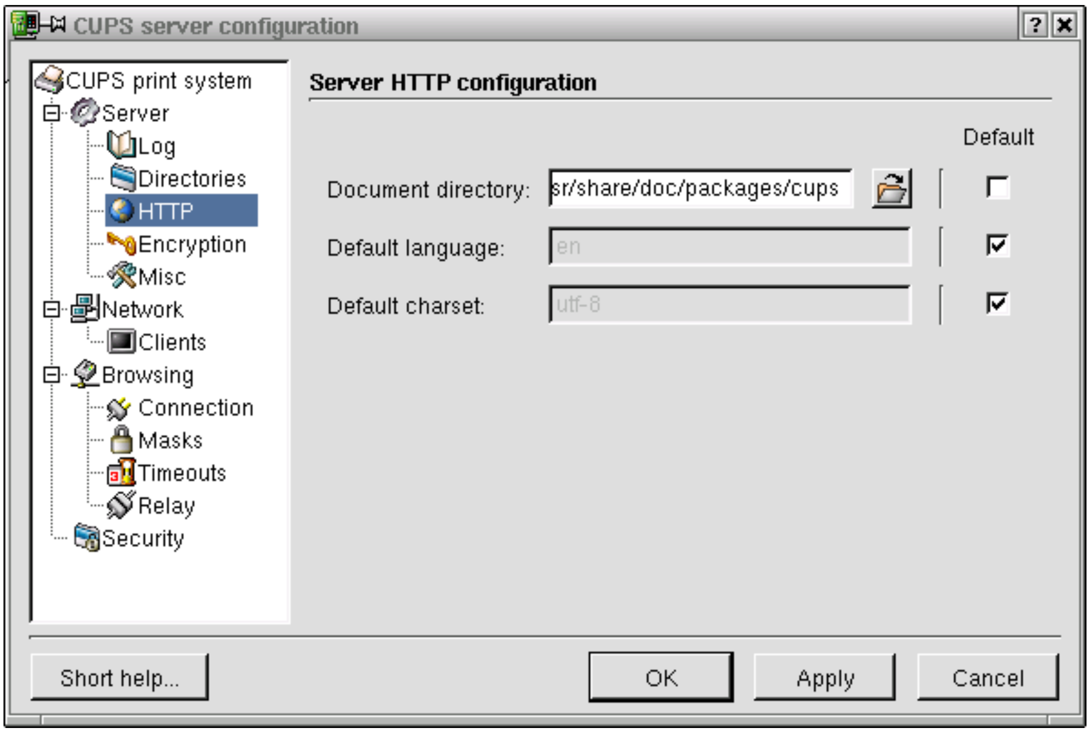
Server HTTP Configuration

Server HTTP Configuration

The dialog to configure the CUPS server HTTP settings is shown here.

CUPS server HTTP settings are the following ones:

- the Document directory
- the Default Language
- the Default Charset



Server HTTP configuration

		Default
Document directory:	sr/share/doc/packages/cups	☐
Default language:	en	☑
Default charset:	utf-8	☑

Document directory
: The root directory for HTTP documents that are served. By default the compiled in directory, `/usr/share/cups/doc`

Default Language
: The default language, if not specified by the browser. If not specified, the current locale is used.

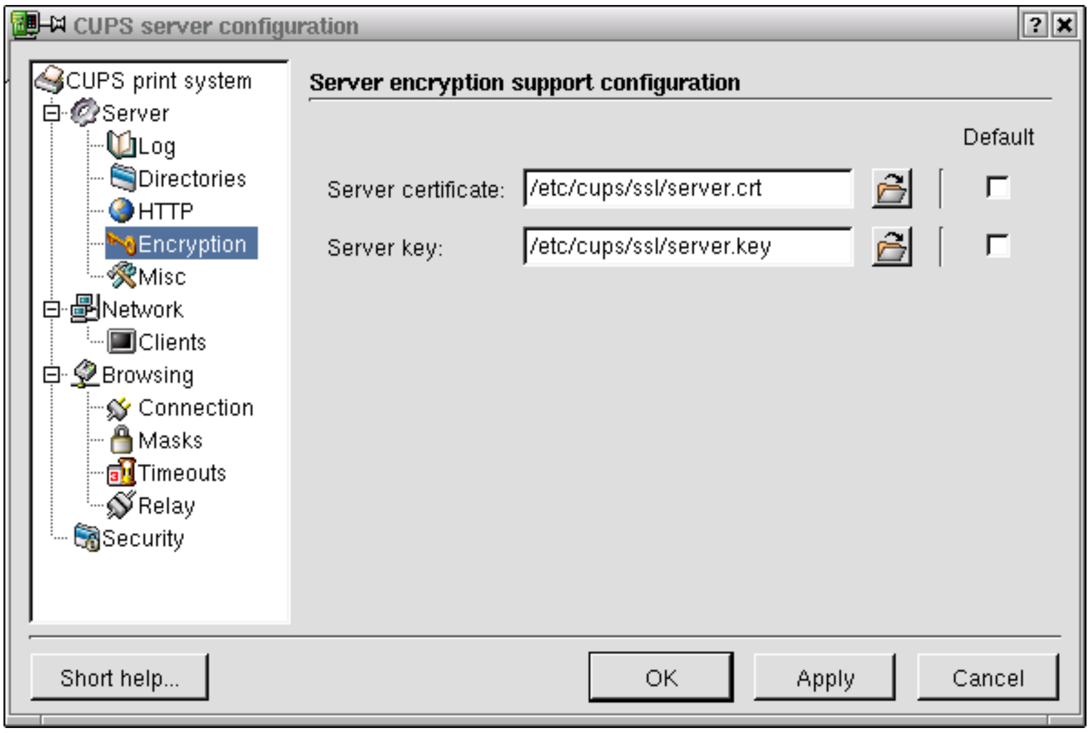
  Use the two letter locale codes, for example **en** or **de**.

Default charset
: The default character set to use. If not specified, this defaults to UTF−8. This can also be overridden directly in the HTML documents.

Server encryption support configuration

This is the dialog to configure the CUPS server security settings. The server encyption support settings are these:

- Server certificate: the file to read containing the server's sertificate
- Server key: the file to read containing the server's key



Server encryption support configuration

Default

Server certificate: /etc/cups/ssl/server.crt

Server key: /etc/cups/ssl/server.key

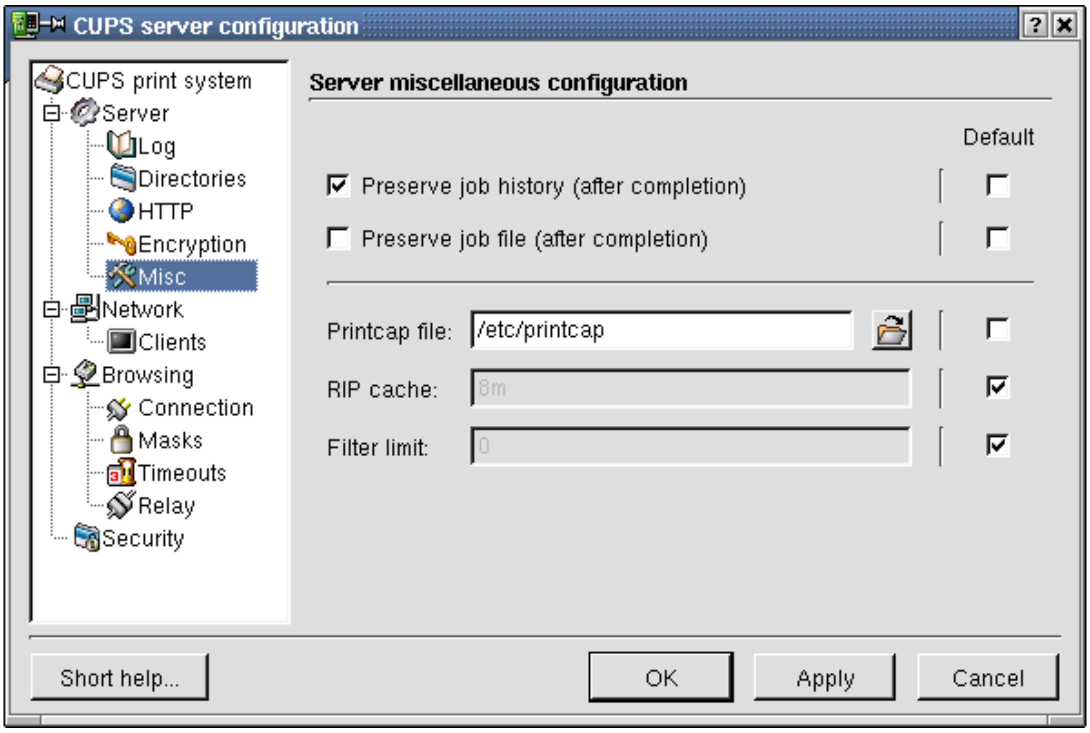
Server certificate
: The file to read containing the server's certificate. Defaults to `/etc/cups/ssl/server.crt`.

Server key
: The file to read containing the server's key. Defaults to `/etc/cups/ssl/server.key`

Server Miscellaneous Configuration

The dialog to configure the CUPS server miscellaneous settings is shown here. The following server settings are done through this screen:

- Preserve job history: whether to preserve a job history for later re−view
- Preserve job files: whether to preserve fully RIP−ed job files for later re−print
- Printcap file: setting the name of and the path to a printcap file
- RIP Cache: setting the size of the RIP cache in memory
- Filter Limit: defining a filter limit



Server miscellaneous configuration

		Default
[✓] Preserve job history (after completion)		[ ]
[ ] Preserve job file (after completion)		[ ]
Printcap file:	/etc/printcap	[ ]
RIP cache:	8m	[✓]
Filter limit:	0	[✓]

Preserve job history (after completion)
Whether or not to preserve the job history after a job is completed, cancelled, or stopped. The default is yes

Preserve job file (after completion)
Whether or not to preserve the job files after a job is completed, cancelled, or stopped. The default is no.

Printcap file
The name of the printcap file. The default is no filename. Leave this blank, to disable printcap file generation.

The printcap setting is only needed to satisfy older applications in need of such a file.

RIP cache

The amount of memory that each RIP should use to cache bitmaps. The value can be any real number, followed by k for kilobytes, m for megabytes, g for gigabytes, or t for tiles, where one tile is 256 x 256 pixels. The default value is 8m.

Filter limit

Sets the maximum cost of all job filters that can be run at the same time. A limit of 0 means no limit. A typical job may need a filter limit of at least 200. Limits less than the minimum required by a job force a single job to be printed at any time. The default limit is 0 (unlimited).

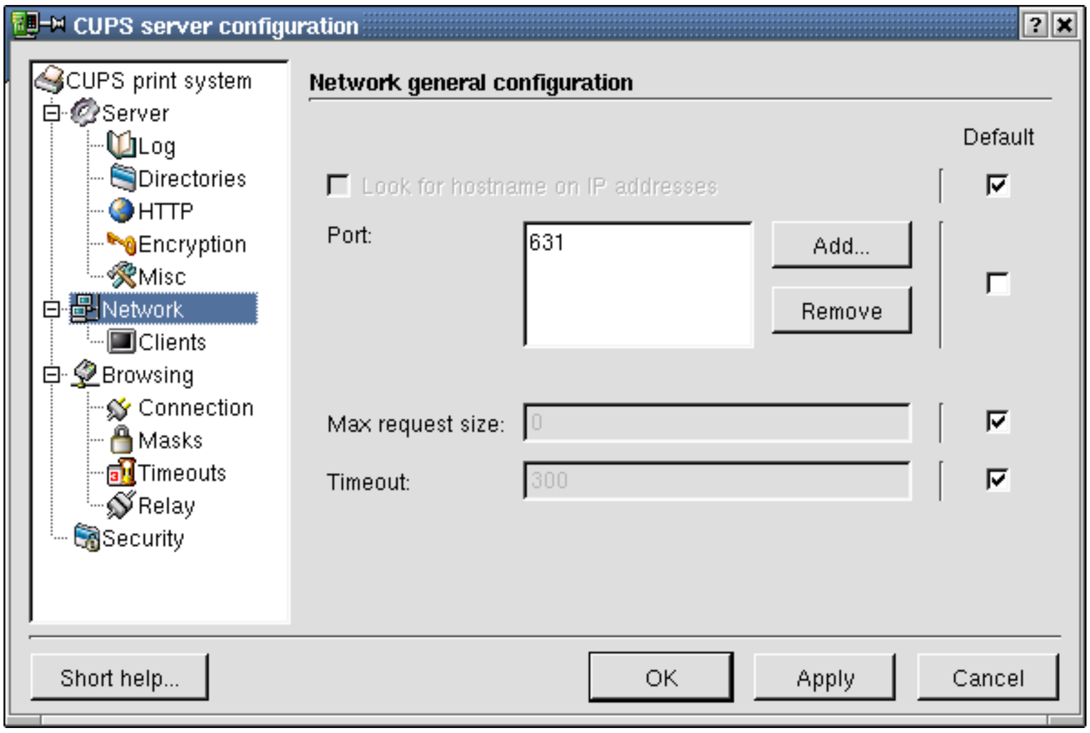
Prev Home Next
Server encryption support configuration Up Network General Configuration

Network General Configuration

Network General Configuration

The dialog to configure the CUPS server network settings is shown here. It includes:

- Look for hostname on IP addresses
- Port
- Max request size
- Timeout



Look for hostname on IP addresses: off; Default ✓
Port: 631; Default unchecked
Max request size: 0; Default ✓
Timeout: 300; Default ✓

Look for hostname on IP addresses
: Whether or not to do lookups on IP addresses to get a fully−qualified hostname. This defaults to off, for performance reasons.

Port
: Enter here Ports and addresses that the server will listen to. The default port 631 is reserved for the Internet Printing Protocol, and is what we use here.

  You can have multiple entries, to listen to more than one port or address, or to restrict access.

Note

Unfortunately, most web browsers don't support TLS or HTTP upgrades for encryption. If you want to support web−based encryption, you'll probably need to listen on port 443, the HTTPS port.

Use the Add and Remove buttons to add and remove entries from the list.

You can enter ports on their own, e.g. **`631`**, or hostnames with ports, e.g. **`myhost:80`** or **`1.2.3.4:631`**.

Max request size
: Controls the maximum size of HTTP requests and print files. The default setting is 0, which disables this feature.

Timeout
: The timeout (in seconds) before requests time out. The default is 300 seconds.

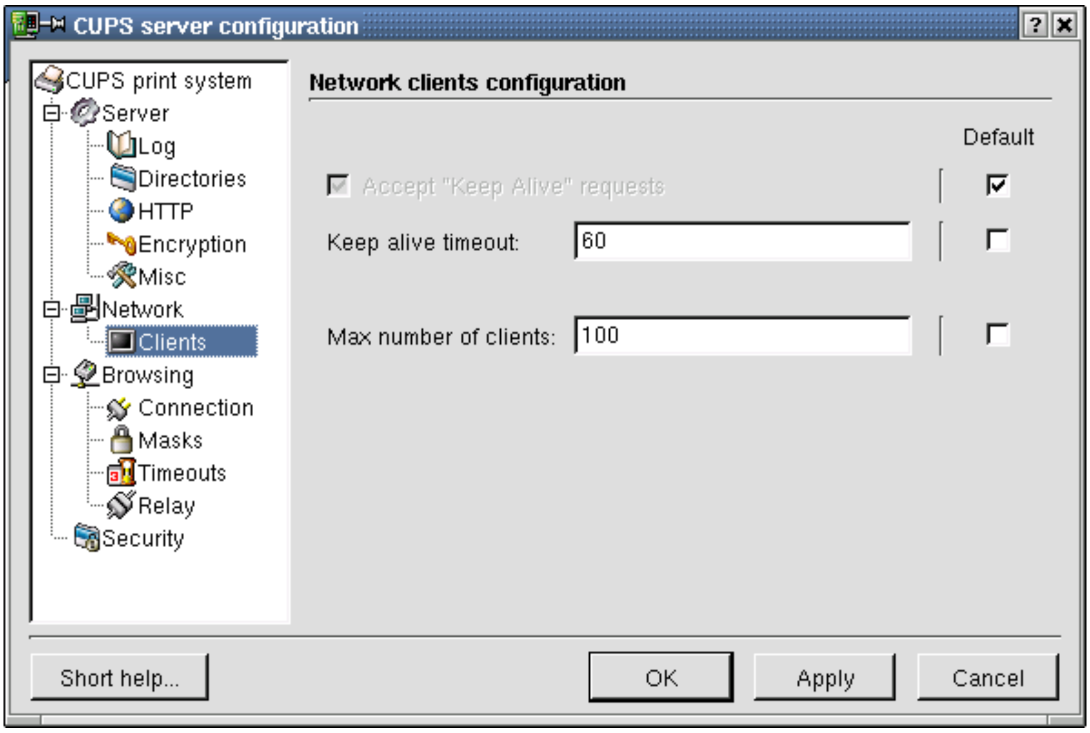
Prev | Home | Next
Server Miscellaneous Configuration | Up | Network Clients Configuration

Network Clients Configuration

Network Clients Configuration

The dialog to configure the CUPS network client settings is shown here. It includes:

- Accept "Keep Alive" requests
- KeepAliveTimeout:
- MaxClients:



Network clients configuration

Default

[✓] Accept "Keep Alive" requests — Default [✓]

Keep alive timeout: 60 — Default [ ]

Max number of clients: 100 — Default [ ]

Accept "Keep Alive" requests
: Whether or not to support the Keep−Alive connection option. The default is on.

Keep alive timeout
: The timeout (in seconds) before Keep−Alive connections are automatically closed. The default is 60 seconds.

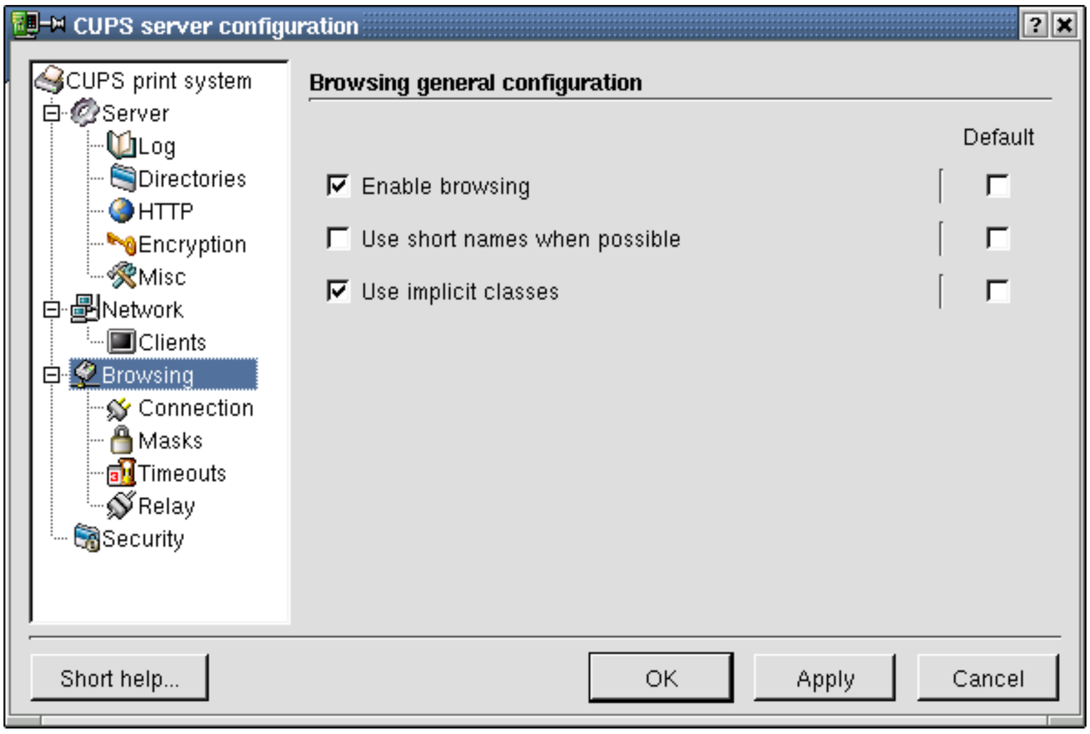
Max number of clients
: Controls the maximum number of simultaneous clients that will be handled. Defaults to 100.

Browsing General Configuration

Browsing General Configuration

The dialog to configure the CUPS browsing general settings is shown here. It includes:

- Enable browsing
- Use short names when possible
- Use implicit classes



Browsing general configuration

Default

[✓] Enable browsing [ ]
[ ] Use short names when possible [ ]
[✓] Use implicit classes [ ]

Enable browsing
: Whether or not to broadcast printer information to other CUPS servers. Enabled by default.

Use short names when possible
: Whether or not to use short names for remote printers when possible (e.g. `printer` instead of `printer@host`). Enabled by default.

Use implicit classes
: Whether or not to use implicit classes.

  Printer classes can be specified explicitly, in the `classes.conf` file, implicitly based upon the printers available on the LAN, or both.

  When Implicit classes are enabled, printers on the LAN with the same name (e.g. `Acme-LaserPrint-1000`) will be put into a class with the same name. This allows you to setup multiple redundant queues on a LAN without a lot of administrative difficulties. If a user sends a job to `Acme-LaserPrint-1000`, the job will go to the first available queue.

This option is enabled by default.

Prev
Network Clients Configuration

Home
Up

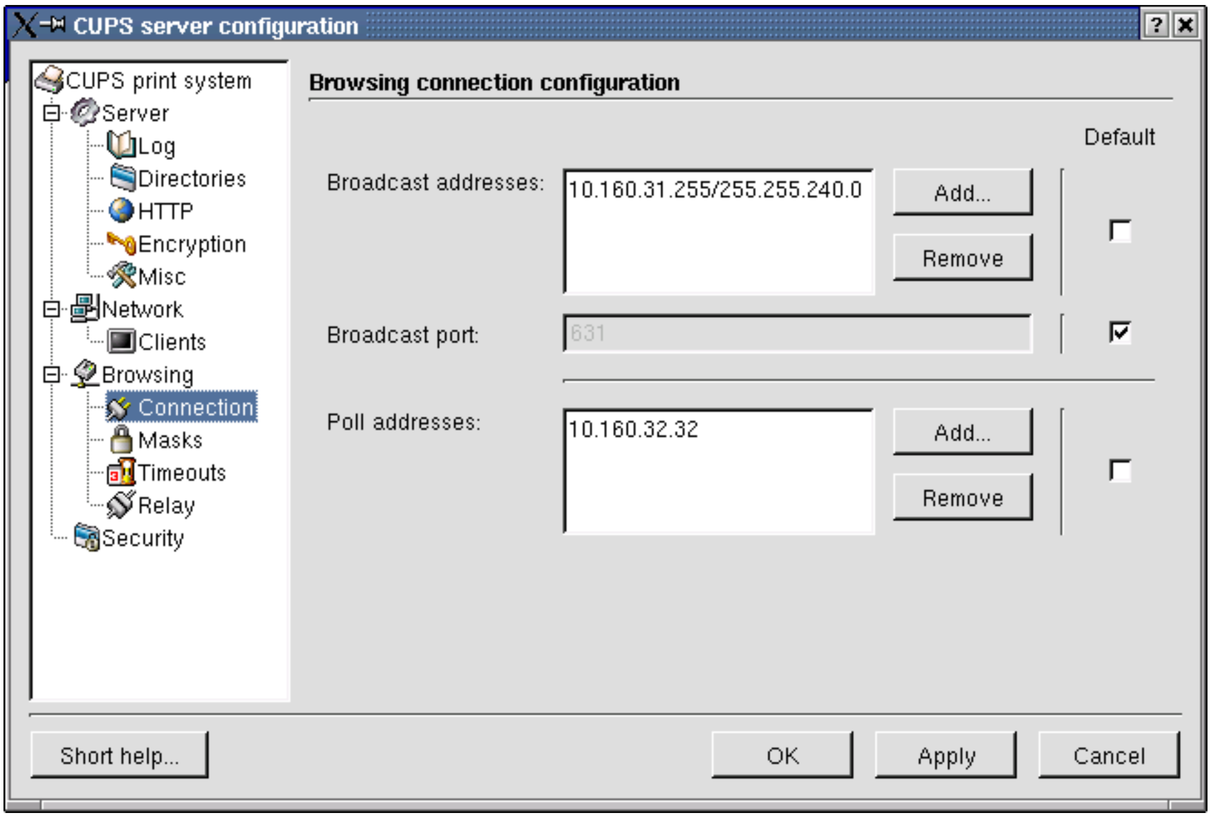
Next
Browsing Connection Configuration

Browsing Connection Configuration

Browsing Connection Configuration

The dialog to configure the CUPS server browsing connection is shown here. Browsing connection settings include:

- Broadcast addresses: The (UDP) broadcast address to transmit printer information to
- Broadcast Port: The port number to use for broadcasting
- Poll addresses: The address(es) to poll for information about printers on servers that might not broadcast (or whose broadcasts might not reach your LAN due to routers in between).



Browsing connection configuration

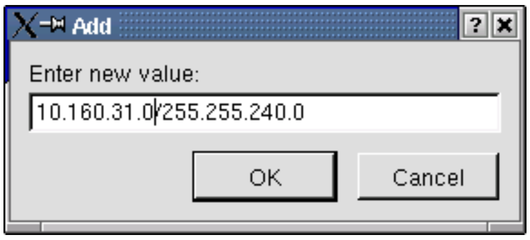
Default

Broadcast addresses: 10.160.31.255/255.255.240.0 ☐

Broadcast port: 631 ☑

Poll addresses: 10.160.32.32 ☐

Broadcast addresses
: After pressing the Add button, you will see the following dialog to enter a new value for outgoing broadcasting browse packets. It is the same kind of dialog as for adding other CUPS server addresses to be polled for printer information.



Enter new value: 10.160.31.0/255.255.240.0

This option specifies a broadcast address to be used. By default, browsing information is broadcast to all active interfaces.

Note

HP−UX® 10.20 and earlier do not properly handle broadcast unless you have a Class A, B, C or D netmask (i.e., there is no CIDR support).

Broadcast port
The port used for UDP broadcasts. By default this is the IPP port; if you change this, you need to do it on all servers. Only one BrowsePort is recognized.

Poll addresses
Poll the named server(s) for printers.

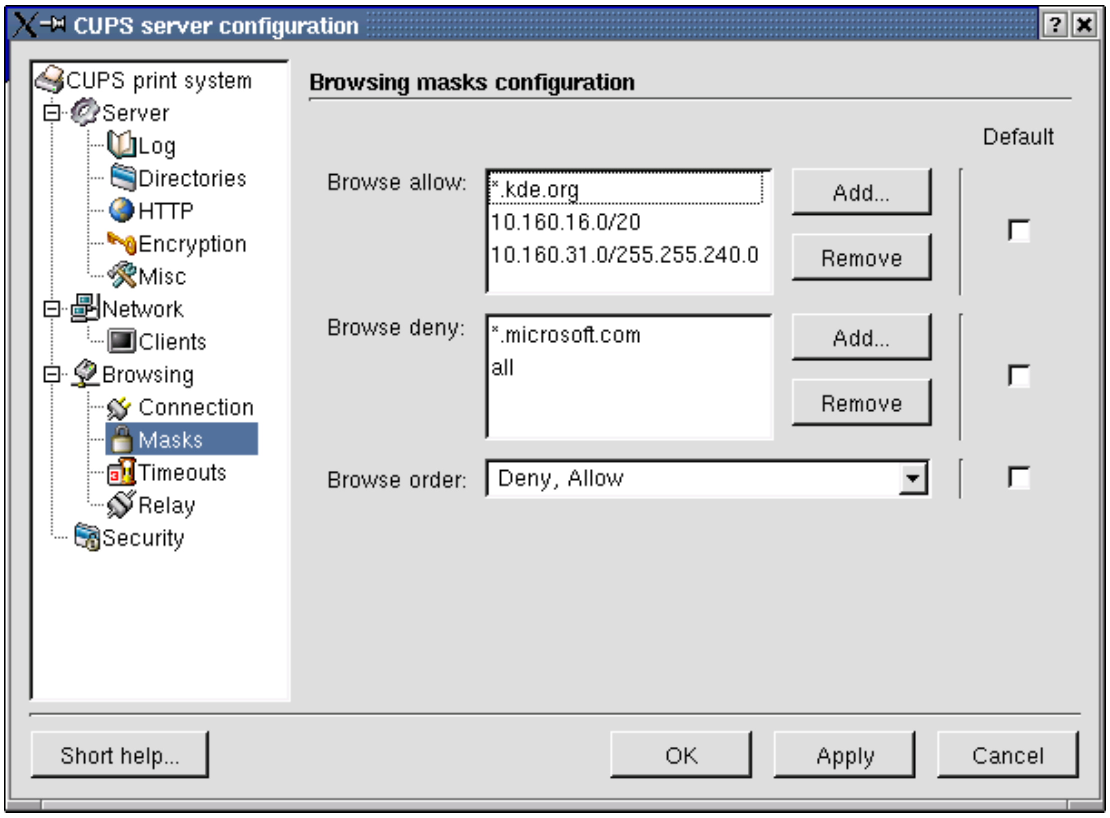
Prev | Home | Next
Browsing General Configuration | Up | Browsing Masks Configuration

Browsing Masks Configuration

Browsing Masks Configuration

The dialog to configure the CUPS server allowed and/or denied browse packets from other servers is shown here.

- Browse allow:
- Browse deny:
- Browse order:



Browsing masks configuration

Browse allow: *.kde.org, 10.160.16.0/20, 10.160.31.0/255.255.240.0

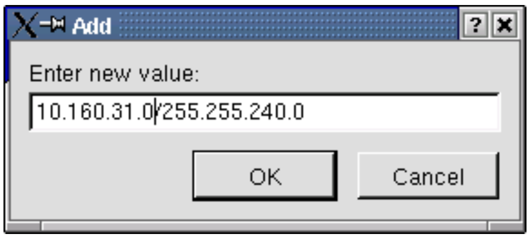
Browse deny: *.microsoft.com, all

Browse order: Deny, Allow

Add Browse Address dialog

The dialog to enter a new value for the address of another CUPS server to accept browse packets from is shown here. It is opend by clicking on the Add... button beside the field named Browse Allow:. It is the same dialog as for adding denied broadcast sending addresses.

The dialog to enter a new value for the address of another CUPS server to accept browse packets from is shown here.



Enter new value: 10.160.31.0/255.255.240.0

Browse allow and Browse deny

Browse allow specifies an address mask to allow for incoming browser packets. The default is to allow packets from all addresses.

Browse deny specifies an address mask to deny for incoming browser packets. The default is to deny packets from no addresses.

Both Browse allow and Browse deny accept the following notations for addresses:

- **`All`**
- **`None`**
- **`*.domain.com`**
- **`.domain.com`**
- **`host.domain.com`**
- **`nnn.*`**
- **`nnn.nnn.*`**
- **`nnn.nnn.nnn.*`**
- **`nnn.nnn.nnn.nnn`**
- **`nnn.nnn.nnn.nnn/mmm`**
- **`nnn.nnn.nnn.nnn/mmm.mmm.mmm.mmm`**

The hostname/domain name restrictions will only work if you have turned hostname lookups on!

Browse order

Specifies the order of the allow/deny comparisons.

Prev
Browsing Connection Configuration

Home
Up

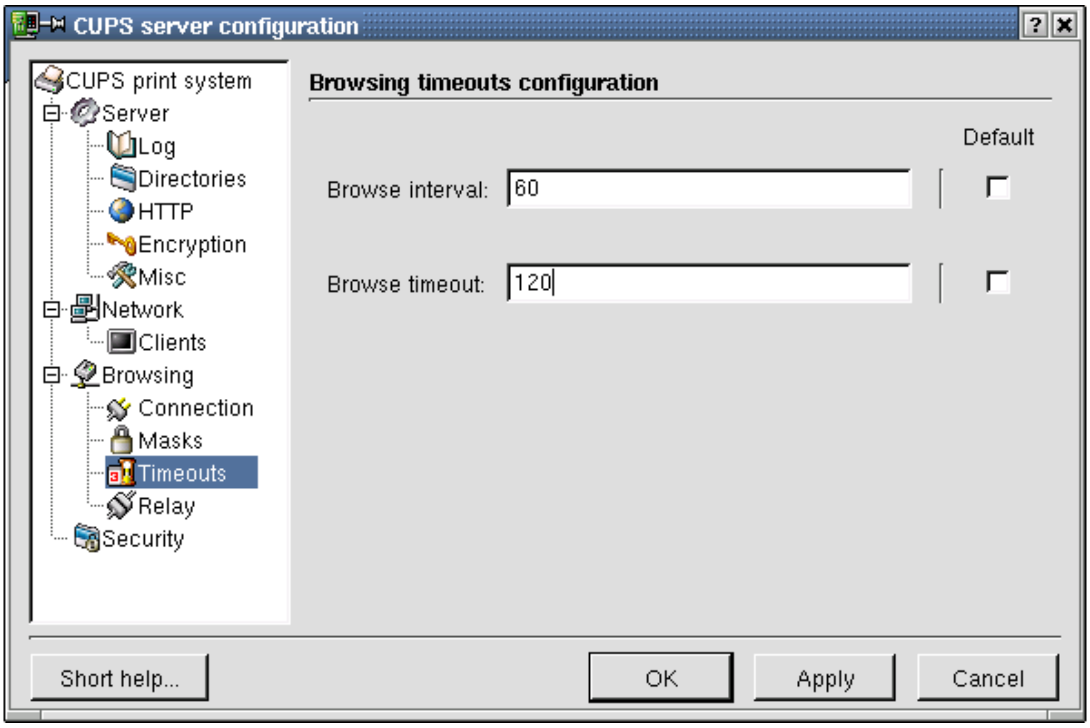
Next
Browsing Timeouts Configuration

Browsing Timeouts Configuration

Browsing Timeouts Configuration

The dialog to configure the CUPS server browse timeout settings is shown here. Browse timeout settings include:

- Browse Interval
- Browse Timeout



Browsing timeouts configuration

Default

Browse interval: 60

Browse timeout: 120

Browse interval
: The time between browsing updates in seconds. The default is 30 seconds.

  Note that browsing information is sent whenever a printer's state changes as well, so this represents the maximum time between updates.

  Set this to 0 to disable outgoing broadcasts so your local printers are not advertised, but you can still see printers on other hosts.

Browse timeouts
: The timeout (in seconds) for network printers − if we don't get an update within this time, the printer will be removed from the printer list.

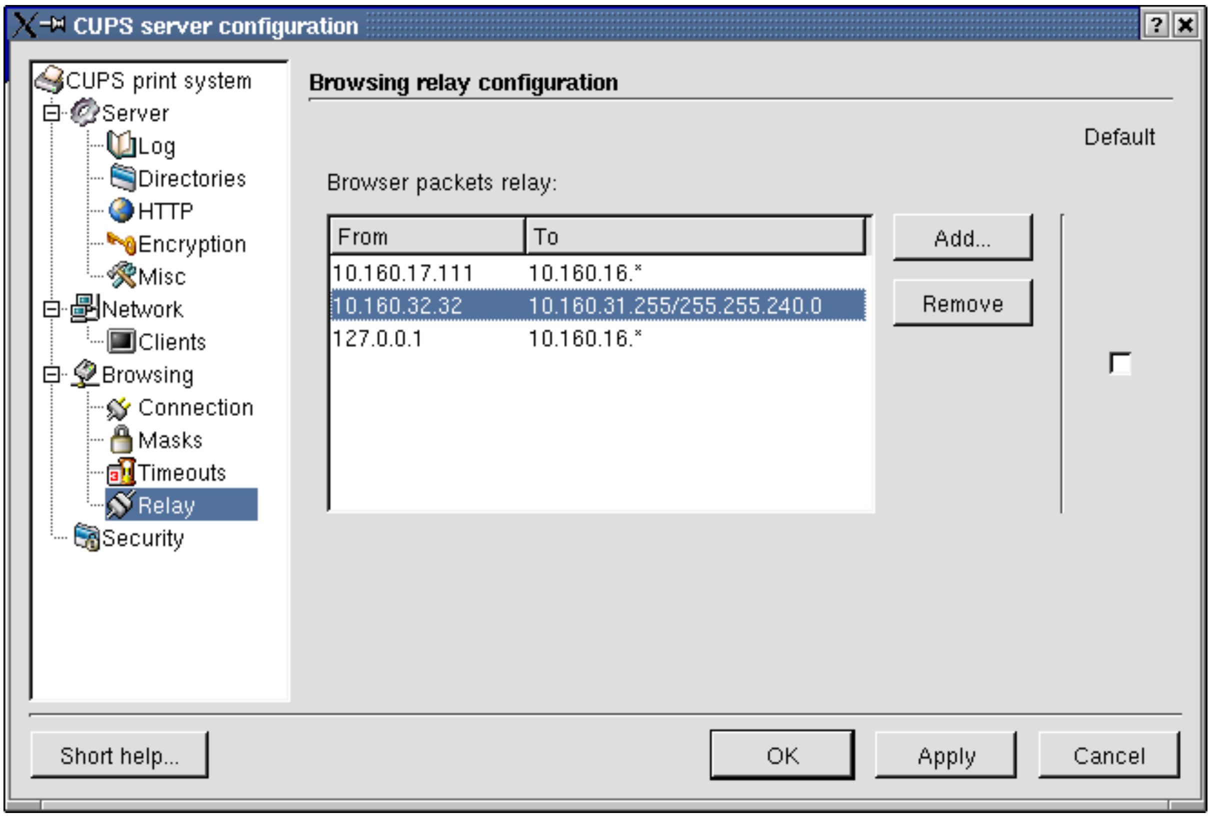
  This number definitely should not be less than the browse interval period, for obvious reasons. Defaults to 300 seconds.

Browsing Relay Configuration

Browsing Relay Configuration

The dialog to configure the CUPS server as a browsing relay is shown here. Browsing relay settings include:

- Browser packets relay



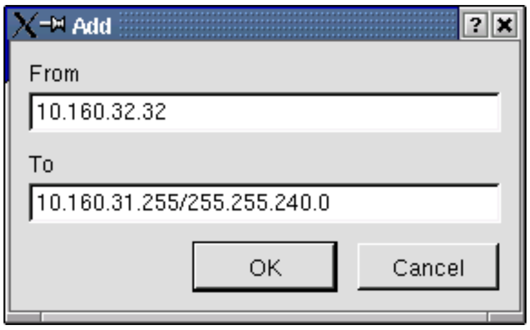
Browsing relay configuration

Default

Browser packets relay:

From	To
10.160.17.111	10.160.16.*
10.160.32.32	10.160.31.255/255.255.240.0
127.0.0.1	10.160.16.*

Add Browse Relay dialog
The dialog to enter a new value for an address pair to define browsing relaying between a CUPS server and a network is shown here.



From: 10.160.32.32

To: 10.160.31.255/255.255.240.0

Browser packets relay
Relay browser packets from one address or network to another.

Prev | Home | Next
Browsing Timeouts Configuration | Up | Security Configuration

Security Configuration

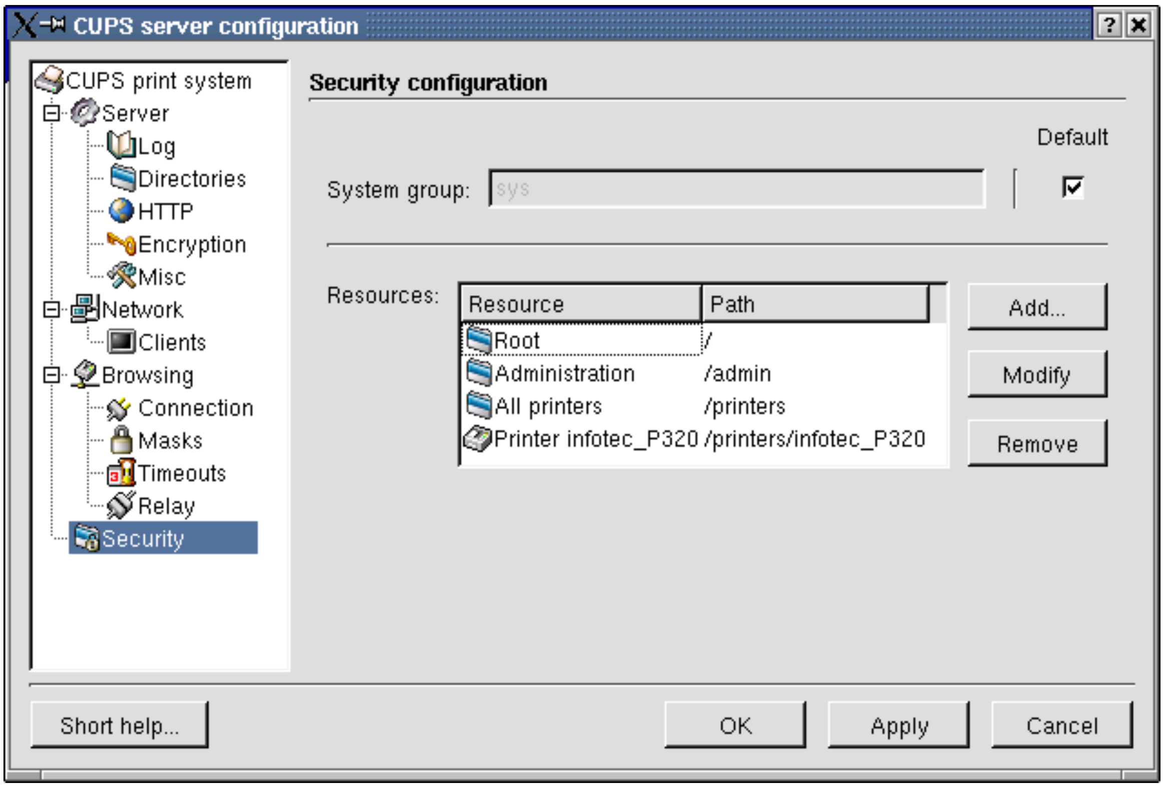
Security Configuration

The dialog to configure the CUPS server security settings for any of the defined server locations is shown here. It contains the following settings, which may be defined seperately for any valid resource (or location) of the CUPS server:

- System Group:
- Access Permissions:
- Auth Type:
- Auth Class:
- Auth Group Name:
- Encryption:
- Allow:
- Deny:
- Order:

Valid resources (or locations) of the CUPS server are:

- Server Root Location: `/`
- Server Administration Location: `/admin`
- All printers on the server: `/printers`
- Any individual printer on the server: e.g. `/printers/infotec_P320`
- All printer classess on the server: `/classes`:
- Any individual printer class on the server: e.g. `/classes/all_infotecs_P320_or_P450`



Security configuration

Default

System group: sys [✓]

Resources:

Resource	Path
Root	/
Administration	/admin
All printers	/printers
Printer infotec_P320	/printers/infotec_P320

Note

For all locations that are not defined seperately the setting of the location   above   it is valid.

For example, you have a printer named `infotec_P450` with no set security options. Then the security of the location `/printers` will take the reponisibility for this printer as it is a sub−location of`/printers`. If, in turn there is no security set for `/printers`, then the security for `/` (the general security) of the server takes responsibility. Either you have set this for your purpose or the compiiled−in default value takes over.

SystemGroup
: The group name for `System` or printer administration access. The default varies depending on the operating system, but will be `sys`, `system` or `root` (checked for in that order).

Access Permissions
: Access permissions for each directory served by the scheduler. Locations are relative to the document root.

Authorization Type
: The authorization to use:

  None
  : Perform no authentication.

  Basic
  : Perform authentication using the HTTP Basic method.

  Digest
  : Perform authentication using the HTTP Digest method.

  ### Note

  Local certificate authentication can be substituted by the client for Basic or Digest, when connecting to the localhost interface.

Authorization Class

The authorization class. Currently only Anonymous , User , System (valid user belonging to the group set as system group), and group (valid user belonging to the specified group) are supported.

Authorization Group Name
: The group name for Group authorization

Encryption
: Whether or not to use encryption. This depends on having the OpenSSL linked into the CUPS library and scheduler.

  Possible values are:

  Always
  : Always use encryption (SSL)

  Never
  : Never use encryption.

  Required
  : Use TLS encryption upgrade.

  IfRequested
  : Use encryption if the server requests it.

Allow
: Allows access from the specified hostname, domain, IP address or network. Possible values are:

  - **`All`**
  - **`None`**
  - **`*.domain.com`**
  - **`.domain.com`**
  - **`host.domain.com`**
  - **`nnn.*`**
  - **`nnn.nnn.*`**
  - **`nnn.nnn.nnn.*`**
  - **`nnn.nnn.nnn.nnn`**
  - **`nnn.nnn.nnn.nnn/mmm`**
  - **`nnn.nnn.nnn.nnn/mmm.mmm.mmm.mmm`**

  The host and domain address require that you enable hostname lookups, as described earlier.

Deny
: Denies access from the specified hostname, domain, IP address or network. Possible values are:

- `All`
- `None`
- `*.domain.com`
- `.domain.com`
- `host.domain.com`
- `nnn.*`
- `nnn.nnn.*`
- `nnn.nnn.nnn.*`
- `nnn.nnn.nnn.nnn`
- `nnn.nnn.nnn.nnn/mmm`
- `nnn.nnn.nnn.nnn/mmm.mmm.mmm.mmm`

The host and domain address require that you enable hostname lookups, as described earlier.

Order

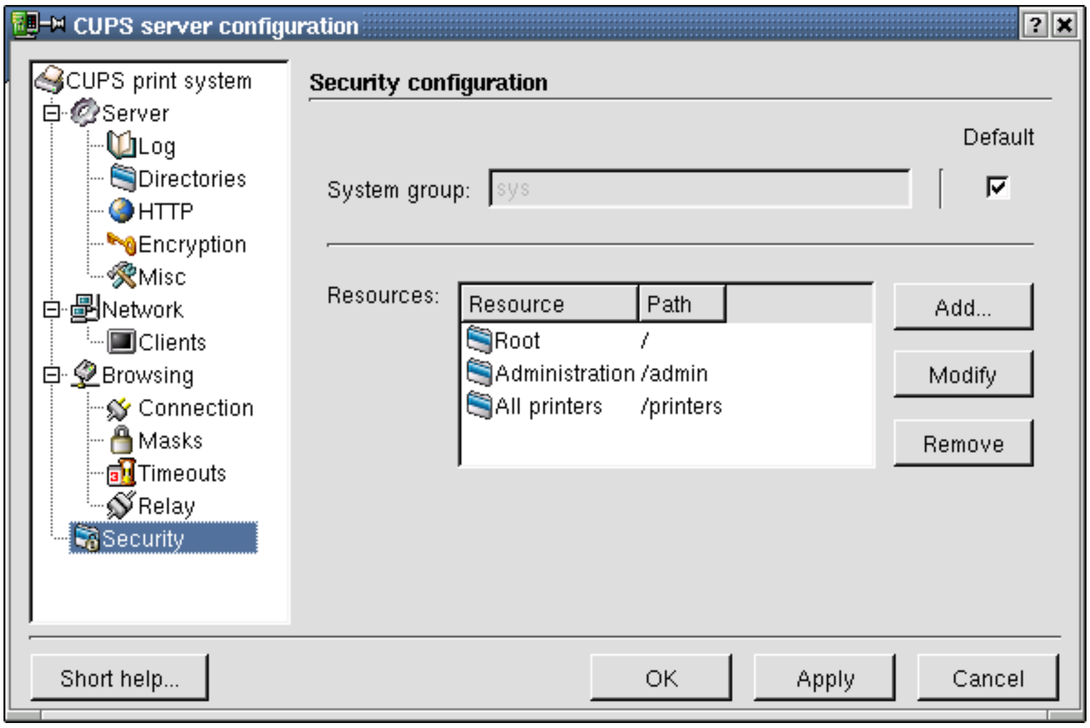
The order of the allow and deny processing.

Prev | Home | Next
Browsing Relay Configuration | Up | Example: How To Define The Security For All Printers

Example: How To Define The Security For All Printers

The dialog to configure the CUPS server security settings is discussed here. We use the example to add security definitions other than the default ones for the resource named `all printers`. For the CUPS web server, this is the location you access through http://localhost:631/printers/ or (remotely) through http://cups.server.name:631/printers/

The first screenshot shows the general location for this setting. Select Add or Modify a resource for which you want to decide about its security settings.



Security configuration

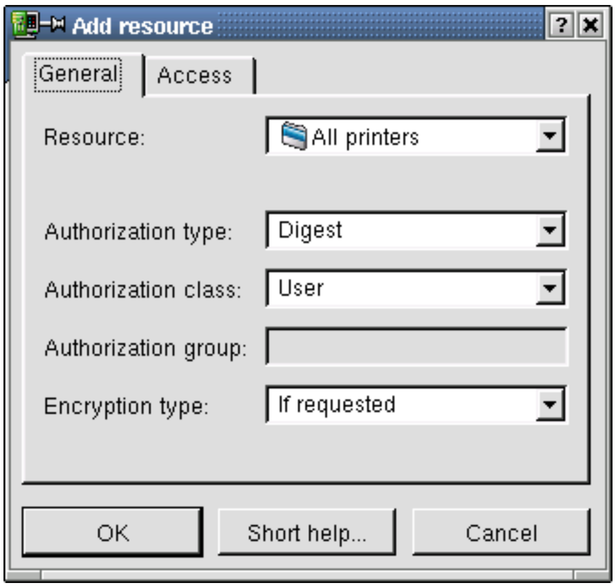
Default

System group: sys [✓]

Resources:

Resource	Path
Root	/
Administration	/admin
All printers	/printers

This dialog is to add a new resource. It looks similar if you want to modify an already existing resource. Here are the general options:


Resource: All printers

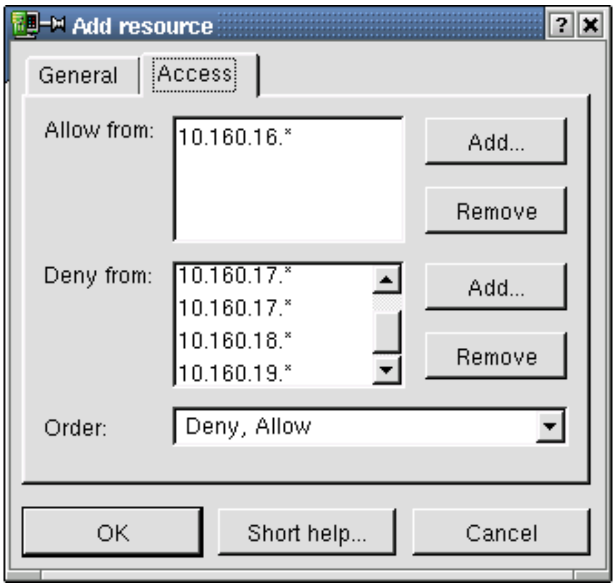
Authorization type: Digest

Authorization class: User

Authorization group:

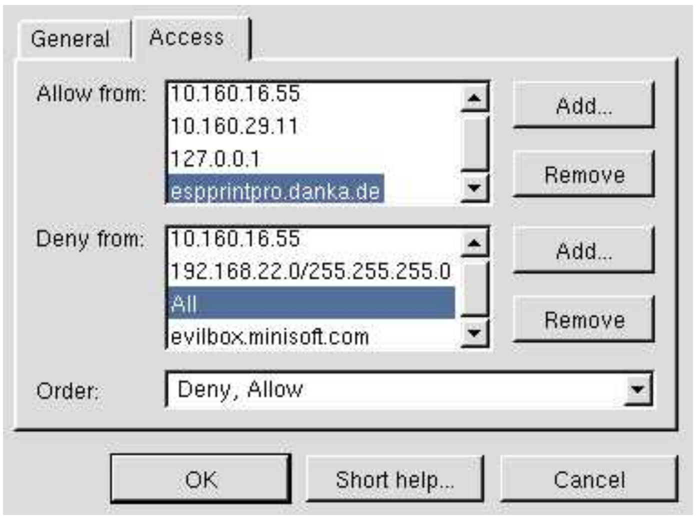
Encryption type: If requested

.This is the second part or the dialog is to add a new ressource. It looks similar if you want to modify an already existing resource. Here you define the actual access masks for the resource in question.


Allow from: 10.160.16.*

Deny from: 10.160.17.* 10.160.17.* 10.160.18.* 10.160.19.*

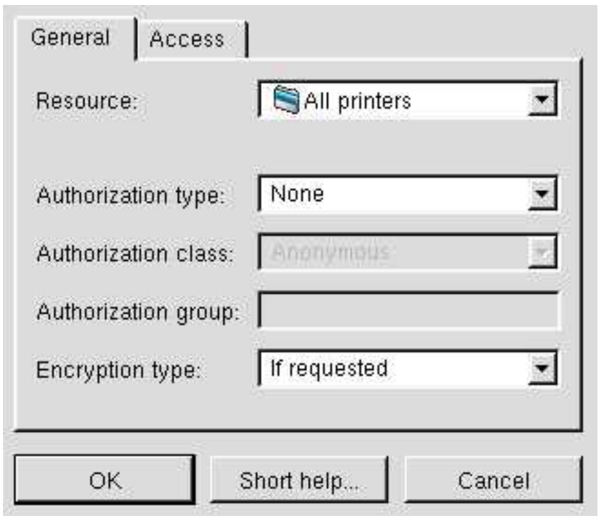
Order: Deny, Allow



Allow from: 10.160.16.55, 10.160.29.11, 127.0.0.1, espprintpro.danka.de

Deny from: 10.160.16.55, 192.168.22.0/255.255.255.0, All, evilbox.minisoft.com

Order: Deny, Allow



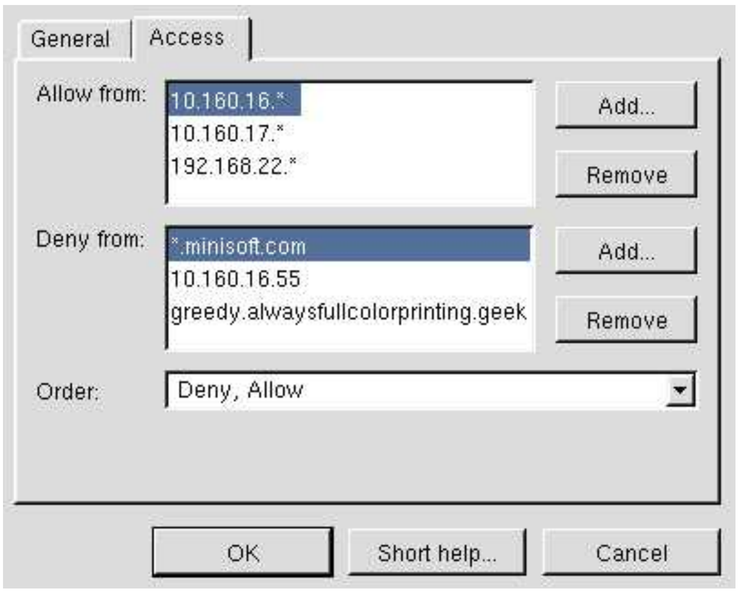
Resource: All printers

Authorization type: None

Authorization class: Anonymous

Authorization group:

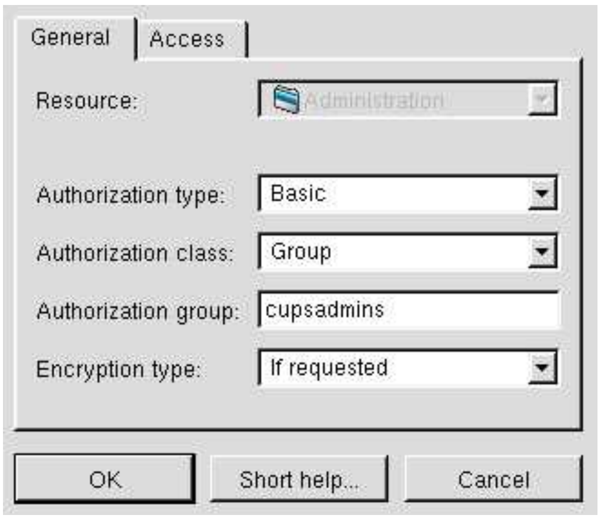
Encryption type: If requested



Allow from: 10.160.16.*, 10.160.17.*, 192.168.22.*

Deny from: *.minisoft.com, 10.160.16.55, greedy.alwaysfullcolorprinting.geek

Order: Deny, Allow



Resource: Administration

Authorization type: Basic

Authorization class: Group

Authorization group: cupsadmins

Encryption type: If requested

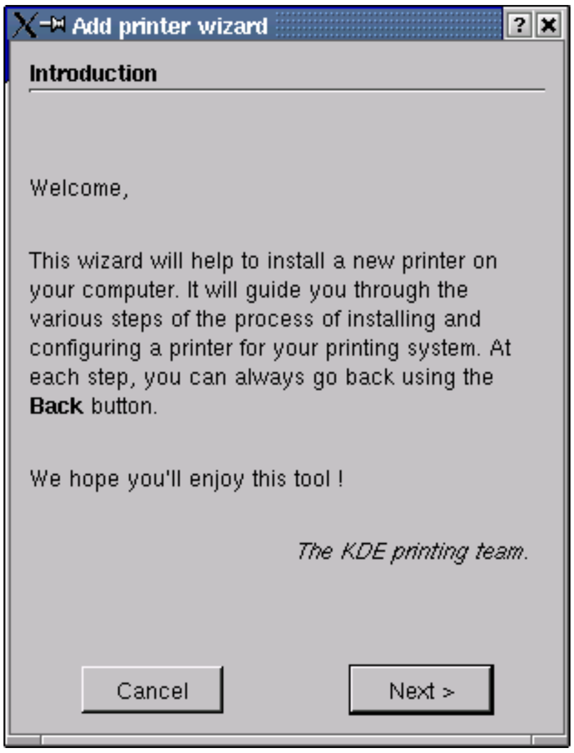
Chapter 7. The Add Printer Wizard for CUPS

Clicking on the most leftward icon of the toolbar in the upper part of the window starts the Add Printer Wizard).

This wizard steps you through various screens to install a new printer. At present this Wizard works for CUPS and the RLPR environment module. The number of steps depend on the actual print−subsystem which is active for you on your box.

Starting

The welcome screen tells you that you can go back any time to change a setting.



Welcome,

This wizard will help to install a new printer on your computer. It will guide you through the various steps of the process of installing and configuring a printer for your printing system. At each step, you can always go back using the **Back** button.

We hope you'll enjoy this tool !

The KDE printing team.

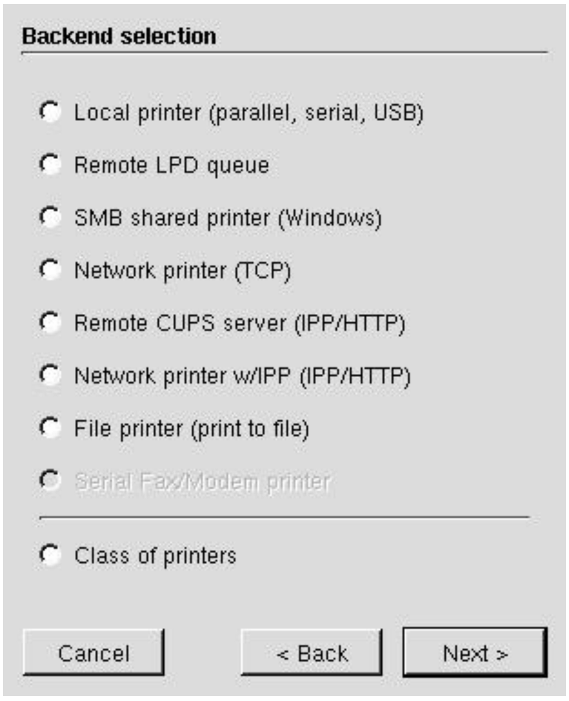
Backend Selection

Backend Selection

Choose the backend protocol CUPS is supposed to use with your new printer. There are:

- local printer (serial, parallel, USB)
- remote LPD queue
- SMB shared printer (Windows®)
- Network Printer (TCP, HP® JetDirect, AppSocket)
- Network printer with IPP (IPP/HTTP)
- File printer
- serial fax /modem printer
- Class of Printers

If some choices are grayed out, they are not available. For example, maybe you have no fax backend software or no modem installed to use this.

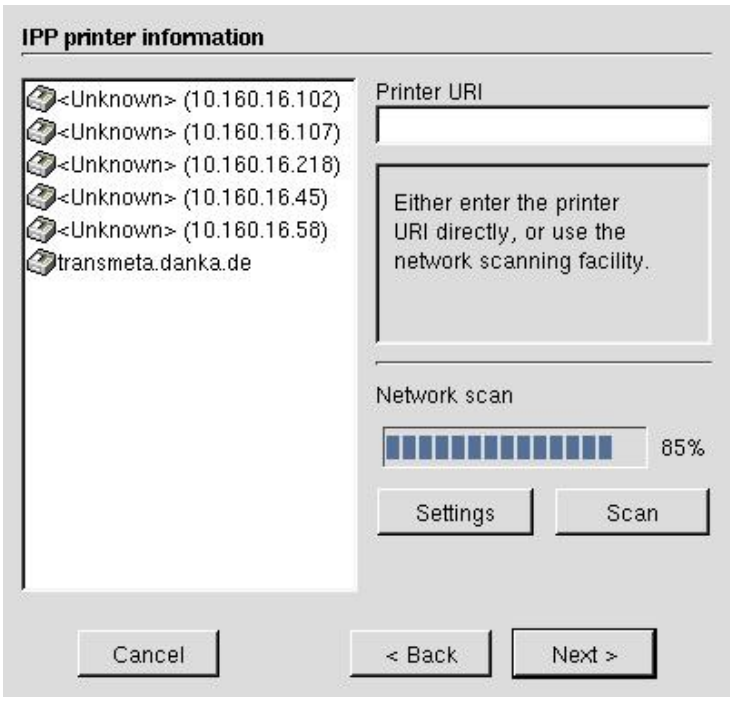

Backend selection
Local printer (parallel, serial, USB)
Remote LPD queue
SMB shared printer (Windows)
Network printer (TCP)
Remote CUPS server (IPP/HTTP)
Network printer w/IPP (IPP/HTTP)
File printer (print to file)
Serial Fax/Modem printer
Class of printers
Cancel
< Back
Next >

Prev
The Add Printer Wizard for CUPS
Home
Up
Next
Direct Network Setting

Direct Network Setting

The contents of your next screen is dependent on your choice in the previous screen. If you know the details, just type them in to configure your network settings directly.

In other cases the wizard can scan the network for you to help you decide which setting could be useful.


IPP printer information

<Unknown> (10.160.16.102)
<Unknown> (10.160.16.107)
<Unknown> (10.160.16.218)
<Unknown> (10.160.16.45)
<Unknown> (10.160.16.58)
transmeta.danka.de

Printer URI

Either enter the printer URI directly, or use the network scanning facility.

Network scan

85%

Information Retrieval by Scanning the Network

Information Retrieval by Scanning the Network

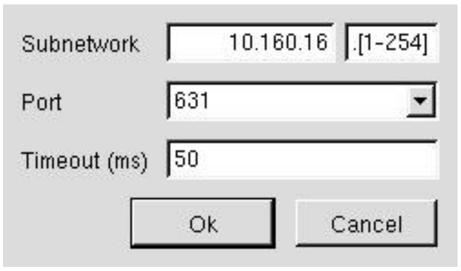
If you use one of the network connections (remote LPD, SMB, remote CUPS, network printer with IPP), you have an option for scanning the net. Be careful when applying this; in some environments network scanning is considered a hostile and harmful act!

In the case of SMB, KDEPrint will use the Samba utilities **nmblookup** and **smbclient** (which need to be installed for this to work) to retrieve the information it presents in a tree structure.

In the case of IPP (Port 631) and TCP Network/AppSocket (Port 9100) KDEPrint will try to open the port and in case of success send an **ipp−get−printer−attribute** request to the printer. For newer HP® printers the latter usually works, because they support both AppSocket and IPP.

Some printers or manufacturers use other port numbers for direct TCP/IP printing. You may need to look up which one to use. The Settings button in the dialog lets you configure your scan, including IP addresses, ports and timeout to use.

Once again: be careful not to be misunderstood as an intruder to your network, if you use the scanning technique.


Subnetwork 10.160.16 .[1-254]
Port 631
Timeout (ms) 50

Printer Model Selection

Printer Model Selection

The hardest part probably is the Printer Model Selection . In former years the situation was difficult, because there were hardly any drivers to find. Now it is, because there are too many, with some of them very good, but many quite broken.

If you have a current database of available drivers on your system, select the manufacturer in the left part of the window first, then the device model in the right part. This split window shows all PPDs found by CUPS in its standard repository of installable PPDs. This repository normally is `/usr/share/cups/model/`. If you want your driver found automatically by CUPS and KDEPrint, throw it in there.

Prev Home Next
Information Retrieval by Scanning the Network Up Driver Selection

Driver Selection

Driver Selection

On the next screen you will see a description of the driver selected previously. This description is extracted from the actual PPD used.

Warning

For a real PostScript® printer *never* try to install a Foomatic or Gimp−Print PPD, even if it is offered. You won't be happy with it. Instead find the original PPD from the manufacturer, preferably the one written for Windows® NT and use it.

Some Linux® distributions have supplied for CUPS every possible combination of ghostscript filters and foomatic PPD files they could find on the net, Many of those are quite useless; they were generated a year ago, when the folks at www.linuxprinting.org made their first experiments with suppying third party PPDs for CUPS. Although dubbed Alpha at the time, these started to take a life of their own and can now be found at various places on the net, doing CUPS no favours.

If you are not sure which ones to use go to:

- http://www.linuxprinting.org
- http://www.cups.org

And ask for help. At a later stage a document detailing the differences of the different driver and PPD models will appear at http://kdeprint.sourceforge.net/ Watch out for this!

Via the Other... button you are able to retrieve any PPD located somewhere on your available file system.

Printer Test and Finding the Right Settings

Printer Test and Finding the Right Settings

Do your first driver settings now. The most important one is the default paper size. This is in many cases set ot Letter . If you are at home in an A4 country and don't want yout first test page to jam: now is the time to prevent this.

You are ready to start a test print. Hit the Test button.

Banner Selection

Banner Selection

The last but one screen lets you select, if and which banners you want to use to mark the beginning and/or end of every printjob on that printer. You can also select and deselect banners before printing in the job options dialogs.

If you need to use custom banners, copy them into `/usr/share/cups/banners/` to make them available for selection. They must be PostScript® files however.

Finally: Baptizing Your New Printer

Finally: Baptizing Your New Printer

The last screen lets you insert a name for your new printer.

The name must start with a letter and may contain numbers and underscores with a maximum size of 128 characters. Stick to it if you want to avoid erratic behaviour of your CUPS daemon. The printer names in CUPS are *not* case sensitive! This is a requirement of IPP. So the names `DANKA_infotec`, `Danka_Infotec` and `danka_infotec` all represent the same printer.

The Final Confirmation Screen

The Final Confirmation Screen

Prev	Home	Next
Finally: Baptizing Your New Printer	Up	CUPS options presently not available through KControl

CUPS options presently not available through KControl

Prev Next

Chapter 8. CUPS options presently not available through KControl

This chapter gives you some hints about further configuration posibilities which may not be available through the KDEPrint GUI interface to CUPS.

Overview of provided features

All of the most often used features and functions CUPS provides are supported in KDEPrint.

- Printer management is supported: add, remove, modify, configure, test, disable, enable ...
- Job management is supported: cancel, hold, release, move to different printer
- Print options: for full control as provided by CUPS.

Prev Home Next
The Final Confirmation Screen Up Where to find help when using CUPS

Where to find help when using CUPS

A lot of information about the inner workings of CUPS is available through the web interface, which CUPS will always support. It works with any browser (yes, even text−based ones). Just go to http://localhost:631/ for a start. There you find a link to locally available CUPS documentation in HTML and PDF if you are new to CUPS.

CUPS is accessible through other means than KDEPrint: commandline and browser are two native CUPS interfaces. The many commandline utilities add up to the most complete control you have on CUPS. The web interface is only a subset of all available configuration or control options.

This is also true for KDEPrint. Generally, as CUPS develops, most new features will first be implemented through the commandline. Be sure to check the latest versions of the man pages for CUPS to stay up−to−date with new features after you install a new version.

Tip

Depending on your update method for CUPS, your active configuration file might not have been re−placed by a new one; thus your new, more capable CUPS−daemon might not have been told by the old configuration file about the new features to use.

A complete list of available files and man pages should always be in the CUPS Software Administrator Manual (http://localhost:631/sam.html#FILES. In the Konqueror URL/location field, type **`man:/lpadmin`** and **`man:/cupsd.conf`** to find out about the most important command and configuration file. You knew already about Konqueror's nice abilties to show you the traditional UNIX® man pages, didn't you? Read this. From there you find more interesting hints and links to other man pages and documentation.

How to find CUPS related man pages

Here is a way to find out which CUPS related man pages there are on your system:

```
kurt@transmeta:~ > apropos cups

cups-calibrate (8)- ESP Printer Calibration Tool
lpstat (1)         - print cups status information
cups-lpd (8)       - receive print jobs + report printer status to lpd clients
classes.conf (5)   - class configuration file for cups
backend (1)        - cups backend transmission interfaces
filter (1)         - cups file conversion filter interfaces
cups-polld (8)     - cups printer polling daemon
mime.types (5)     - mime type description file for cups
cupsd (8)          - common unix printing system daemon
lpadmin (8)        - configure cups printers and classes
cupsd.conf (5)     - server configuration file for cups
mime.convs (5)     - mime type conversion file for cups
printers.conf (5) - printer configuration file for cups
mime.convs (5)     - mime type conversion file for cups
cups-polld (8)     - cups printer polling daemon
lpstat (1)         - print cups status information
backend (1)        - cups backend transmission interfaces
mime.types (5)     - mime type description file for cups
cupsd (8)          - common unix printing system daemon
lpadmin (8)        - configure cups printers and classes
printers.conf (5) - printer configuration file for cups
cupsd.conf (5)     - server configuration file for cups
filter (1)         - cups file conversion filter interfaces
```

CUPS options presently not available through KControl

Outside KDEPrint: Hints & Tips Tricks with CUPS on the Commandline

Outside KDEPrint: Hints & Tips & Tricks with CUPS on the Commandline

Here are a few examples of options that are presently only available if you use the commandline.

Allowing or denying printer access for certain users

When installing (or modifying) a printer through the command line, you can either deny or allow the usage of that printer to certain users. Take the example of a Heidelberg DigiMaster 9110:

```
lpadmin -p Digi9110 -v lpd:/10.160.16.99/mqueue -u allow:kurt,sylvi,hansjoerg -E -P /home/kurt/PPDs/DVHV.ppd
```

will allow the usage of this (believe me: very nice and also very professional) printer to only the three mentioned users and at the same time deny it to all others. If another user wants to print on the DigiMaster via this CUPS server, he will receive an error message along the lines client−error−not−possible.

```
lpadmin -p Digi9110 -v lpd:/10.160.16.99/mqueue -u deny:tackat,boss,waba -E -P /home/kurt/PPDs/DVHV.ppd
```

will deny the usage of this same printer to the three mentioned users and at the same time allow it to all others. If denied user wants to print on the DigiMaster via this CUPS server, he will receive an error message along the lines client−error−not−possible.

Note

Only one of the two options may be used at one time; at present there is no support to have a similar option in a per−group based way. This will be implemented in the future.

Imposing Quotas for certain printers

Sometimes you want to impose quotas for certain printers. With quotas you can set upper limits for the number of pages or the amount of data to be printed over a certain period to a certain printer.

Quotas can be set with the `−o` option when installing a printer with the **lpadmin** command, or afterwards for an already existing printer. Following are some guidelines (which are missing at the time of writing in the, official CUPS documentation):

- With CUPS you may have pagecount− and filesize−based quotas for individual printers.
- Quotas are calculated for each user individually (so a single set of limits applies to all users for the printer concerned).
- Quotas include banner pages (if those are used).
- This means: you can limit every user to 20 pages per day on an expensive printer, but you cannot limit every user except `Kurt` or `root`.
- There are `job-k-limit`, `job-page-limit`, and `job-quota-period` options to give when setting up a printer.
- `job-quota-period` sets a time interval for quota computing (intervals are determined in seconds; so a day is 60x60x24=86.400, a week is 60x60x24x7=604,800, and a month is 60x60x24x30=2.592.000 seconds.)
- For quotas to be enforced, the time−period *plus* at least one job−limit must be set to non−zero.
-

The default value of 0 for `job-k-limit` specifies that there is no limit.

- The default value of 0 for `job-page-limit` specifies that there is no limit.
- The default value of 0 for `job-quota-period` specifies that the limits apply to all jobs that have been printed by a user that are still known to the system.

Working Examples:

Working, as both, time−period *plus* one or both job−limits are defined

```
lpadmin -p danka_infotec_4850 -o job-quota-period=604800 -o job-k-limit=1024
```

This sets a limit of a file size of 1 MB (in total) for each user of existing printer `danka_infotec_4850` during one week.

```
lpadmin p danka_infotec_4105 -o job-quota-period=604800 -o job-page-limit=100
```

This sets a limit of 100 pages (in total) for each user of existing printer `danka_infotec_4105` during one week.

```
lpadmin -p danka_infotec_P450 -o job-quota-period=604800 -o job-k-limit=1024 -o job-page-limit=100
```

This sets a combined limit of 1 MB (in total) and 100 pages (in total) for each user of existing printer `danka_infotec_P450` during one week. Whichever limit is reached first will take effect.

Not working examples

NOT working, as only *one*, time−period *or* job−limit is defined)

```
lpadmin -p danka_infotec_P320 -o job-quota-period=604800
lpadmin -p danka_infotec_FullColor -o job-page-limit=100
lpadmin -p danka_infotec_HiSpeed -o job-k-limit=1024
```

Related Error Messages

Once a user reaches his quota limit, he'll get a client−error−not−possible message, if he wants to print.

Installing a raw printer

There are different ways to define a raw printer. One comfortable one is to use the **lpadmin** command. Just don't define a PPD file to be used for that printer and it will be a raw one:

```
lpadmin -p Raw_Danka_infotec -E -v lpd://10.160.16.137/PORT1
```

Raw printer queues are those which don't touch the print file to transform it to a different file format. You need this for example when printing from Windows® clients via Samba through a CUPS server to a PCL printer: in this case the Windows® side printer driver would generate the finished print file format for the target printer and filtering it through CUPS filters would only harm the purpose. Under certain circumstances (if you want to make sure that the file goes to the printer unfiltered by CUPS) the **lpadmin** without a PPD comes in handy.

Troubleshooting CUPS in KDEPrint

This section of the KDEPrint Handbook will live from the readers' feedback. Here is just a small beginning.

Error Messages

1 What does the error client−error−bad−request mean?

The user sent a file to the CUPS which the server could not process. You get this also upon sending an empty file.

2 And client−error−not−possible?

User is either not allowed to print to a certain printer or has achieved his quota (based on file size and/or page number)

3 How about client−error−not−found?

The user tried to access a nonexistant resource on the CUPS server, such as trying to print a nonexistent file, or one that you are denied permission to read.

Questions and Answers

1 Why can't I re−start my jobs?

To be able to re−start your completed jobs from the web interface, you need a setting in the `/etc/cups/cupsd.conf` file: set **`PreserveJobFiles True`**.

2 How do I get rid of the long list of completed jobs in the web interface?

TODO

3 How does page accounting work?

CUPS does the print accounting by passing nearly every job through the pstops filter. This one does, amongst other things, the page counting. Output of this filter there may be piped into other filters (like pstoraster −−> rastertopcl) or sent to the printer directly (if it is a PostScript® printer).

In any case, this works for network, parallel, serial or USB printers the same. For pstops to work, it needs DSC, Document Structuring Convention compliant PostScript® (or near−equivalent) as input. So it calculates the pages during filtering on the print server and writes info about every single page (what time, which user, which job−ID and −name, which printer, how many copies of which pages of the document, how many kilo−bytes?) into `/var/log/cups/page_log`.

By the way: on my personal wishlist is a hack of webalizer to read and analyse the page_log and give a similar output. Anyone?

However, it is *not* giving correct results in the following cases:

*

- The printer jams and maybe therefor throw away the job (real live experience; or maybe throwing away the job because of problems with the data format)
- Jobs printed as raw are always counted as size of 1 page (and maybe multiple copies).

Therefore the page accounting of CUPS is only an approximation (in many cases an excellent or at least good one, in others a quite poor one). The only reliable print count is the one done by the internal printer counter. (Because this is the one you pay for, if you are on a click price or similar.) Some, by far not most, printers can be queried remotely for that information via SNMP (Simple Network Management Protocol). That means, in a bigger network with many different printers there *is* just no completely reliable and accurate page accounting tool!

4 Why doesn't page−accounting work with Windows® clients?

From Windows® clients jobs nearly always need to be sent as raw . Why? If CUPS works as a print server for Windows® clients using the original native Windows® driver for the target print device, this guarantees the correct formatting of the job on the clients already; therefor the server should not touch it and print raw ; therefor no filtering is started (and this is not even possible as the input from the clients is not PostScript® as pstops expects; hence no page−count other than the default á .

5 How do I get a list of available options for a given printer or a PPD file?

See the man page for the **lpoptions** command. You may investigate a CUPS−enabled box about any option of its available printers. There is no need to have the printer installed locally. As long as the printer is available locally (through the CUPS printer browsing feature), it will also work remote.

To query for a printers' option typing **`lpoptions −p`** ***`HitachiDDP70MicroPress`*** **`−l`** will give a long listing of all available options as read from the PPD file for the given Hitachi−Printer (in my case installed on remote server transmeta). Remote server `Transmeta` and its CUPS daemon as well as the localhost's CUPS daemon need to be up and running for this to succeed.

6 How do I read the listing retrieved by the **lpoptions** command?

You know that for PostScript® printer manufacturers it is legal to define their own internal names and procedures even for standard PostScript® options. As long as the driver is able to retrieve the option from the PPD and show it to the user in a way that he understands it everything is OK. But what do *you* do, if you want to use some obscure printer options on the command line? How do you find out its exact syntax?

Let's take an example. Looking at Hitachi's DDP70 printer and how it implements duplex printing is revealing somehow. How do you tell how to print double sided? duplex or Duplex? Or another name altogether?.

```
lpoptions −h transmeta −p Hitachi_DDP70_ClusterPrintingSystem −l | grep uplex
```

This leads to the output

```
TR−Duplex/Duplex: False *True
```

This is to be interpreted like follows:

- The name of the investigated option is `TR−Duplex`;
- Behind the slash you see the translation of the option, as it should be shown in a GUI or Web interface (Duplex);
- The option may take one of the two values *`False`* or *`True`*;
- The present setting is *`True`* to be recognized by the marking with a star `*`.

To override the present default setting (duplex) and print a job in simplex, you need to use the following command:

```
lpr −P Hitachi_DDP70_ClusterPrintingSystem −o TR−Duplex=False /path/to/your/printjob
```

7 How do I get a nicely formatted listing of available options for a given printer or PPD?

Use the **lphelp** command which may be installed on your system locally. There is not yet a man page for **lphelp**.

```
lphelp infotecP450
```

This lists the available options for the named printer. It is nicely formatted and does explain every available option and how to use it. You can query different printers' options at once:

```
lphelp infotec7410color DANKA_fullcolor_D2000 HP_ColorLaserJet8550
```

It also works for PPD files. Just specify the path to the PPD:

```
lphelp /home/kurt/PPDs/HP-ColorLaserJet8550.ppd
```

Solving Problems

No system is perfect. Here are some commonly seen traps people have fallen into.

1 My printer named ã−lp−duplex shows erratic behavior. What's wrong?

The printer names used in CUPS shall start with a letter and may contain up to 128 letters, numbers or underscores. Using dashes may lead to problems. Speaking about naming: printer names in CUPS are not case sensitive. So a printer named `Best_of_Danka` will be the same as `best_of_danka` or `BEST_OF_DANKA`. (This is a requirement of IPP, which CUPS is fully compliant with).

2 Why do I get Unable to connect to SAMBA host: Success with my printer shares from Windows® accessed via Samba?

Are the rights on the remote Windows® box set correctly for you? Are you actually allowed to print on the Windows® shared printer?

3 My files for printer `lp` sometimes mysteriously disappear and two days later I am told they got printed on a printer 3 storeys below my office. What's on?

Believe me, it is very unlikely that your printer is the only one with the name `lp`. Maybe CUPS is playing a trick on you. As you might have the setting ImplicitClasses On activated, CUPS tries to stuff all printers it sees on the network into a Class name lp. All jobs destined to lp are sent to this class and the first available member prints it. So if you had this nice fellow (who listened closely when you raved about CUPS and KDEPrint) install CUPS and poke around the system...get the idea?

Take my advice: choose a unique name for any network printer! (Mind you, the one on your parallel port also turns out to be a network printer for the rest of the world if you don't take care of your settings).

Prev
Outside KDEPrint: Hints & Tips Tricks with CUPS on the Commandline

Home
Up

Next
Module Built Around rlpr Utility

Module Built Around rlpr Utility

Chapter 9. Module Built Around rlpr Utility

Overview of provided features

Printer management: basic operations are supported (add/remove/modify).

Each user can predefine the printers he wants to use by specifying the host and related printer queues. Printers are stored on a per user basis . This module is built around the rlpr utility rlpr

Chapter 10. Generic LPD Module (UNIX®)

Overview of Provided Features

Module used by default (on first start for example).

Generic modules that only allows to send print jobs. No printer or job management supported. It is made to work on a wide variety of UNIX® flavor: Linux®/LPR, HP−UX®, Solaris, IRIX®. It also supports some LPRng extensions (like the absence of continuation character \ in `printcap` files).

Chapter 11. LPR (BSD)

Plain (old?) LPR support. An LPRng module is in development, and hopefully available for 2.3 release.

Overview of Provided Features

- Printer management: basic support to add/remove/configure a printer, compatible with Red Hat®−6.x systems (**printtool** + rhs−printfilers packages).
- Job management: not supported
- Print options: basic control

Prev Next

Chapter 12. LPRng

An LPRng module for KDEPrint is in development, and hopefully available for the KDE 2.3 release.

Prev Home Next
LPR (BSD) Up Module For External Print Command (Netscape®−like)

Chapter 13. Module For External Print Command (Netscape®−like)

This module allows to completely specify the print command (Netscape®−like). An edit line is added in the print dialog for that purpose. Can be used in many cases, for example with a self−made print program.

Overview of provided features

- Printer management: not supported
- Job management: not supported.
- Print options: basic control, depending on your knowledge of the print command

Chapter 14. KDEPrint Extensions To All Print Subsystems

Prev Home Next
Module For External Print Command (Netscape®−like) Up External Filters

External Filters

External Filters

The enscript Filter for Text Files

The n−up Filter for Any File

Three different Make Pamphlet Filters for PostScript® Files

Prev Home Next
KDEPrint Extensions To All Print Subsystems Up Final word from the Author

Chapter 15. Final word from the Author

Who am I, what is my business?

My employer is Danka Deutschland GmbH, a leading and manufacturer−independent provider of professional and hi−speed digital printing systems, black−and−white as well as color. Danka provides for Hardware, Software, Service, Maintainance. Consumables and customized solutions around the products in its portfolio. There I work as a System Engineer. Amongst the brands Danka offers are Heidelberg (formerly Kodak), Canon, Hewlett−Packard®, Hitachi, infotec and EfI.

My acquaintance with Linux® and the Free Software community is not too old. When I started to play around with Linux® at the beginning of 1999, my deepest disappointment was the poor support for printing. True, I made all our machines spit out simplex prints −− but what about duplex? What about punching the output? How to make sorting work? Or stapling, cover sheets and all the other beautiful finishing options our engines offer to customers? No way −− at least for me as a non−geek!

I went on a search in the internet for a solution. Fortunately not much later, in May 1999, Mike Sweet, principal developer of CUPS, announced the first Beta release of this superb piece of printing software. After trying it shortly, I knew this was just it!

Next thing I attempted: to make Linux® distributions interested in this new stuff. Believe me −− it was more than tenacious! They seemed to think they had already the best thing they could get in printing. One reason probably was that they (and many Linux® developers) never had to think about how to best support a printer duplexer −− because there had never come one near their own desks...

Finally, my attempts to make some Linux® print publications interested in CUPS backfired on me − one editor squeezed me into writing a series on the subject myself. And this is how some people started to give me the nickname CUPS Evangelist . I will not get rid of this nick anytime soon, now, that even the KDE people wedged me into their timeframe of releases. Oh, boy...

Anyway, CUPS is now making its way around the world and it might well become a triumphal one: I am a little bit proud to have supported and contributed to this from the near beginning.

It should encourage you: Even if some more experienced Linux® users than you are sceptical about it and even if your programming skills are next to zero (like mine) − there are a lot of tasks and jobs and ideas and talent that you can contribute to the Free Software community. Not least inside the KDE project... ;−)

Credits

Credits

I'd like to thank...

- Mike Sweet for developing CUPS in the first place
- Jean−Eric Cuendet for starting kups and qtcups, the predecessors of KDEPrint
- Michael Goffioul for doing all the hard work recently
- Martin Konold for thinking twice
- Sven Guckes for teaching me a few things about the art of survival on the terminal (just in case KDE is not there ;−)
- ...too numerous others to mention who also let me snatch bits and bytes of knowledge off them
- and last, but not least: Tom Schwaller for encouraging me to get into documentation writing

Prev Home Next
Final word from the Author Up Caveats

Caveats

Caveats

KDEPrint has been developed on a system using CUPS 1.1.6. KDEPrint has been tested on other versions of CUPS and so far no incompatibilities are known. By the time of writing this Handbook, CUPS 1.1.9 is out with a few new features not yet supported by KDEPrint. Of course you are able to access these features, but you will need to bypass KDEPrint and use the CUPS command−line tools or edit configuration files manually. KDEPrint's development will go on and this Handbook strives to always be the best available user documentation resource for it.

Credits And Licenses

Prev

Chapter 16. Credits And Licenses

Prev Home
Caveats Up