SAMBA Project Documentation

Printing with Samba 3.0
(Traditional Printing Systems & CUPS)

Kurt Pfeifle, Danka Deutschland Holding GmbH

I ask you to provide feedback on this documentation. It is intended for inclusion into the Samba 3.0 HOWTO Collection, due to be released around May/June this year. I also want to kindly ask you for a donation into the Samba Team Fund if you appreciate the effort. −− Samba Team Fund Donations are used to cover travel expenses of Samba Team members visiting CIFS conferences etc. You can make donations by check. Thanks in advance! Please make checks payable to the 'Samba Team' and send them to:

```
Samba Team
26 Carstensz st
Griffith, ACT
2603 Australia
```

Version 0.97beta11 (*Last Update: Sun Jun 01st 23:44:15 UTC 2003*)
originally prepared for SambaXP in Goettingen (07th−08th of April, 2003)

supplemented afterwards

About the author:

Kurt is working as a System Specialist for Danka Deutschland Holding GmbH. His employer is part of the world wide Danka organisation, one of the biggest manufacturer−independent vendors for software and hardware solutions in the digital and office printing business. Back in winter 1998/99, when he started to take an interest in Linux, one of the things he was appalled was the poor support for professional printing requirements, compared to what he was used to from his Windows and MacOS work experiences. Even if he used the most modern printing machines at his company (which do a lot of inline finishing automatically like booklet−making, stapling, edge−trimming, folding and saddle−stitching), he didn't succeed to produce other things than what he called "a collection of loose paper sheets, but no real printout". Soon he discovered that the home Linux user had no more comfort in configuring his (smaller) printer.

Fascinated by Linux' other features nevertheless, he started to search the Internet for some alternative printing solutions working on Linux. As this was the time when CUPS saw its first announcement to the world (still in Alpha or Beta stage, and hardly anyone took notice then), he gave it a spin and became addicted very quickly.

With CUPS it worked. His claim to fame now is to have been the first person on the planet **;−)** to print a professional A4 booklet, 88 pages on 22 A3−sheets, finished with folding, edge−trimming and saddle−stitched stapling on a 250.000 EURO Heidelberg−Digimaster 110ppm−printer −− all under the complete control of a native Linux client printing system!

Slightly dissappointed at the lame reception CUPS had harvested initially, he started to peddle at different Linux distributions to try make them include it in their bundles. As he had no quick success he soon extended his activities to different Unix and Linux print publications to get them writing a feature on this new promising printing thingy. However, most journalists he talked to on the phone seemed to be too busy with other occupations than to seriously lend an ear to this "Linux rookie" who seemed to talk them into some new software they had not heard of... "You can not really print with Linux? Gosh, what a nonsense this bloke is talking..."

Only one of the editors was receptive for the request and promised to look for an author... But Kurt's initiative heavily backfired on him soon after: he got the editor's phone−call who told him he couldn't find a competent author and: "Write it yourself or you won't see anything like it!"

That's how the story started of a Linux newbie (with some general knowledge about network printing) to become known as the "CUPS evangelist" who quickly found himself writing a series of articles for different print publications (Linux−Magazin, Linux−User, LinuxEnterprise, iX) as well as the web; later he authored the "KDE Print Handbook" and the CUPS−FAQ, gave talks and highly successfull tutorials and workshops on "CUPS Printing" at different big and small Linux events.

His latest project now includes organising and leading professional seminars and trainings on behalf of his employer on "IPP, CUPS and Printing in hetergeneous environments".

(taken from the "UKUUG Linux Developers Conference" website)

Current work.

At the present time, Kurt is working on a book about "CUPS and Network Printing". It will be published in spring 2004, in English as well as in German. A very small part of its content is now condensed into this first draft for the CUPS Printing chapter of the "Samba HOWTO Collection, Version 3.0".

Printing Support in SAMBA 3.0

0.97beta11, 01−06−2003. post−SambaXP
Kurt Pfeifle, Danka Deutschland GmbH

Table of Contents

```
Changelog

01-05-2003: corrected typos
            add a very lean smb.conf example
            work on other smb.conf examples
            how to use "testparm" to discover "hidden" settings
            how to set default driver values for all clients
            sketch out a logon script to install printers without user interaction

02-05-2003: add paragraph about "tdbbackup"

03-05-2003: extended explanation for adding drivers remotely (APW + rpcclient)
            added info about how to find out which driver files are required
            added info about how to get hold of the driver files
            added info about running "rpcclient adddriver|setdriver" manually

04-05-2003: made a new structure of content, reorganized headings
            heavily extended manual installation of driver with the help of rpcclient,
                                                (this is now a step-by-step tutorial)

06-05-2003: a lot of typos corrected -- most is spellchecked now
            checked some of my recipes (they still worked ;-)
            wrote a few sentences about printing *from* Samba *to* CUPS
            include more rundll32 examples (one using also "runas")

17-05-2003: integrated most of the feedback comments from various people
            A lot of feedback from Ken Sarkies -- thanks a lot!
            Ciprian Vizitiu crafted the PNG flowcharts from my original ASCII art. Wow!
                                       (a few bugs and typos in the PNGs need to be corrected.)

25-05-2003: don't confuse files with identical names from "version 2" and "version 3" drivers!
            (included an example)
            Many suggestions by Ken Sarkies made it into this draft

01-06-2003: *complete* overhaul for the foomatic-rip/Linuxprinting.org drivers' section; needs
            to be further streamlined
            inclusion of a more complex example smb.conf for CUPS
            re-create paragraphs about CUPS backends (had mysteriously disappeared sometime back)
            elaborated the example about "installing large numbers of printers"


================
  CONTRIBUTORS
================
Special thanks go to Dragan Krnic, Ken Sarkies and Ciprian Vizitiu. Dragan provided some
very valuable feedback about the manual installation of drivers using rpcclient; Ken did
a meticulous review of the text with many suggestions for improvements; Ciprian converted
my initial ASCII-art flowcharts into the nice PNGs. 2 dozen more people contributed spelling
and translation fixes and other improvements. Thanks!
```

PART IV. PRINTING

Chapter 6. "Classical" Printing Support in Samba 3.0

Last Update : Sun Jun 01st 23:44:15 UTC 2003

Abstract

This is a part of the upcoming collection of HOWTOs added to Samba documentation in preparation of the 3.0 version release. I try to ensure that "everything printing" is current, but it seems a larger job than one person can maintain. Please send updates and corrections to kpfeifle@danka.de.

Cheers, Kurt

6.1. Features and Benefits

Printing is often a mission−critical service for the users. Samba can provide this service reliably and seamlessly for a client network consisting of Windows workstations.

A Samba−3.0 print service may be run on a Standalone or a Domain member server, side by side with file serving functions, or on a dedicated print server. It can be made as tight or as loosely secured as needs dictate. Configurations may be simple or complex. Available authentication schemes are essentially the same as described for file services in previous chapters. Overall, Samba's printing support is now able to replace an NT or Windows 2000 print server full−square, with additional benefits in many cases. Clients may download and install drivers and printers through their familiar "Point'n'Print" mechanism. Printer installations executed by "Logon Scripts" are no problem. Administrators can upload and manage drivers to be used by clients through the familiar "Add Printer Wizard". As an additional benefit, driver and printer management may be run from the commandline or through scripts, making it more efficient in case of large numbers of printers. If a central accounting of print jobs (tracking every single page and supplying the raw data for all sorts of statistical reports) is required, this is best supported by CUPS as the print subsystem underneath the Samba hood.

This chapter deals with the foundations of Samba printing, as they implemented by the more traditional UNIX (BSD− and System V−style) printing systems. Many things apply to CUPS, the newer Common UNIX Printing System, too; so if you use CUPS, you might be tempted to jump to the next chapter −− but you will certainly miss a few things if you do so. Better read this chapter too.

Note, that the author verified most of the given examples on Windows XP Professional clients. If you encounter quotes describing the responses to commands given, bear in mind that Windows 2000 clients are very similar, but may differ in details. Windows NT is somewhat more different. Also, this was done on a German Win XP editon. Translating back to English may not be as accurate as desired. (If you want to help with future editions of this document, please send in correct transcripts of what you see on your screen for the English versions of the various clients).

6.2. Technical Introduction

Samba's printing support always relies on the installed print subsystem of the Unix OS it runs on. Samba is a "middleman". It takes printfiles from Windows (or other SMB) clients and passes them to the real printing system for further processing. Therefore it needs to "talk" to 2 sides: to the Windows print clients and to the Unix printing system. Hence we must differentiate between the various client OS types which behave differently as well as the various UNIX print subsystems, which have different features and are accessed differently. This part of the Samba HOWTO Collection deals with "the traditional" way of Unix printing first; the next chapter covers in great detail the more modern *Common UNIX Printing System* (CUPS). *CUPS users, be warned: don't just jump on to the next chapter*. You might miss important info contained only here!

We assume here that you have your printer(s) already configured to print from UNIX: At the very least they must be set up to print when you send pre−formatted printjobs to them. It is then Samba's job to "share" this service to Windows clients.

6.2.1. What happens if you send a Job from a Client

To successfully print a job from a Windows client via a Samba print server to a UNIX printer, there are 6 (potentially 7)

stages:

- 1. Windows opens a connection to the printer share
- 2. Samba must authenticate the user
- 3. Windows sends a copy of the printfile over the network into Samba's spooling area
- 4. Windows closes the connection again
- 5. Samba invokes the print command to hand the file over to the UNIX print subsystem's spooling area
- 6. The Unix print subsystem processes the print job
- (7.) The printfile may need to be explicitely deleted from the Samba spooling area.

6.2.2. Printing related Configuration Parameters

There are a number of configuration parameters in *smb.conf* controlling Samba's printing behaviour. Please do also turn to the man page for smb.conf to acquire an overview of these parameters. As with other parameters, there are Global (tagged with a "*G*" in the listings) and Service Level ("*S*") parameters.

- Service level parameters ***may*** go into the [Global] section of smb.conf. In this case they define the default behaviour of all individual or service level shares (provided those don't have a different setting defined for the same parameter, thus overriding the global default).
- Global parameters ***may not*** go into individual shares. If they go in by error, the "testparm" utility can discover this (if you run it) and tell you so.

6.2.3. Parameters recommended for Use

These configuration parameters are directly related to Samba−3.0 printing. Their meaning is explained further below.

LIST OF PRINTING RELATED PARMETERS IN SAMBA−3.0

The following `smb.conf` parameters directly related to printing are used in Samba 3.0. See also the smb.conf man page for detailed explanations:

Global level parameters

- *addprinter command (G)*
- *deleteprinter command (G)*
- *disable spoolss (G)*
- *enumports command (G)*
- *load printers (G)*
- *lpq cache time (G)*
- *os2 driver map (G)*
- *printcap name (G), printcap (G)*
- *show add printer wizard (G)*
- *total print jobs (G)*
- *use client driver (G)*

Service level parameters

- *hosts allow (S)*
- *hosts deny (S)*
- *lppause command (S)*
- *lpq command (S)*
- *lpresume command (S*
- *lprm command (S)*
- *max print jobs (S)*
- *min print space (S)*
- *print command (S)*
- *printable (S), print ok (S)*
- *printer name (S), printer (S)*
- *printer admin (S)*
- *printing = [cups|bsd|lprng...] (S)*
- *queuepause command (S)*
- *queueresume command (S)*
- *total print jobs (S)*

Samba's printing support implements the Microsoft Remote Procedure Calls (MS−RPC) methods for printing. These are used by Windows NT (and later) print servers. The old "LanMan" protocol is still supported as a fallback resort, and for

older clients to use. More details will follow further beneath.

6.2.4. Parameters for backwards Compatibility

There have been two new parameters added in Samba 2.2.2, still present in Samba−3.0:

- One is for better support of Samba 2.0.x backwards capability (***disable spoolss***). It will disable Samba's support for MS−RPC printing and yield identical printing behaviour to Samba 2.0.x.
- The other one for using local printer drivers on Windows NT/2000 clients (***use client driver)***. It does not apply to Windows 95/98/ME clients.

Both of these options are described in the smb.conf(5) man page and are disabled by default. Use them with caution!

PARAMETERS "FOR BACKWARD COMPATIBILITY ONLY", USE WITH CAUTION

The following *smb.conf* parameters were already in Samba 2.2; they are only used for backward compatibility:

- *disable spoolss (G)*
- *use client driver (S)*

6.2.5. Parameters no longer in Use

Samba users upgrading from 2.2.x to 3.0 need to be aware, that some previously available settings are no longer supported (as was announced in time). Here is a list of them:

"OLD" PARAMETERS, REMOVED IN SAMBA−3.0

The following `smb.conf` parameters have been deprecated already in Samba 2.2 and are now completely removed from Samba 3.0. You can not use them in new 3.0 installations:

- *printer driver file (G)*
- *total print jobs (G)*
- *postscript (S)*
- *printer driver (S)*
- *printer driver location (S)*

6.3. A simple Configuration to print with Samba 3.0

Here is a very simple example configuration for print related settings in the *smb.conf* file. If you compare it with your own system's smb.conf, you probably find some additional parameters included there (as pre−configured by your OS vendor). Further below is a discussion and explanation of the parameters. Note, that this example doesn't use many parameters. However, in many environments these are enough to provide a valid smb.conf which enables all clients to print.

```
[global]
        printing = bsd
        load printers = yes

[printers]
        path = /var/spool/samba
        printable = yes
        public = yes
        writable = no
```

This is only an example configuration. Many settings, if not explicitly set to a specific value, are used and set by Samba implicitly to its own default, because these have been compiled in. To see all settings, let root use the "*testparm*" utility. "testparm" also gives warnings if you have mis−configured certain things. Its complete output is easily 340 lines and more. You may want to pipe it through a pager program.

The syntax for the configuration file is easy to grasp. You should know that smb.conf is not very picky about its syntax. It has been explained elsewhere in this document. A short reminder: It even tolerates some spelling errors (like "browsable" instead of "browseable"). Most spelling is case−insensitive. Also, you can use "Yes|No" or "True|False" for boolean settings. Lists of names may be separated by commas, spaces or tabs.

6.3.1. Verification of "Settings in Use" with *testparm*

To see all (or at least most) printing related settings in Samba, including the implicitly used ones, try the command outlined below (hit "ENTER" twice!). It greps for all occurrences of "lp", "print", "spool", "driver", "ports" and "[" in testparm's output and gives you a nice overview about the running smbd's print configuration. (Note that this command does not show individually created printer shares, or the spooling paths in each case). Here is the output of my Samba, with exactly the same scarce settings in smb.conf as shown above:

```
transmeta: # testparm -v | egrep "(lp|print|spool|driver|ports|\[)"

Load smb config files from /etc/samba/smb.conf.simpleprinting
Processing section "[homes]"
Processing section "[printers]"

[global]
       smb ports = 445 139
       lpq cache time = 10
       total print jobs = 0
       load printers = Yes
       printcap name = /etc/printcap
       disable spoolss = No
       enumports command =
       addprinter command =
       deleteprinter command =
       show add printer wizard = Yes
       os2 driver map =
       printer admin =
       min print space = 0
       max print jobs = 1000
       printable = No
       printing = bsd
       print command = lpr -r -P'%p' %s
       lpq command = lpq -P'%p'
       lprm command = lprm -P'%p' %j
       lppause command =
       lpresume command =
       printer name =
       use client driver = No

[homes]

[printers]
       path = /var/spool/samba
       printable = Yes
```

You can easily verify which settings were implicitly added by Samba's default behaviour. Don't forget about this point -- it may be important in you future dealings with Samba.

NOTE: testparm in Samba-3.0 behaves differently from 2.2.x: used without the "-v" switch it only shows you the settings actually written into smb.conf! To see the complete configuration used, add the "-v" parameter to testparm.

6.3.2. A little Experiment to warn you

Should you need to troubleshoot at one stage, please always go back to this point first and verify if "testparm" shows the parameters you expect! To give you an example from personal experience as a warning, try to just "comment out" the "load printers" parameter. If your 2.2.x system behaves like mine, you'll see this:

```
kde-bitshop:/etc/samba # grep "load printers" smb.conf
#      load printers = Yes    # This setting is commented oooouuuuut!!

kde-bitshop:/etc/samba # testparm -v smb.conf | egrep "(load printers)"
       load printers = Yes
```

Despite of my imagination that the commenting out of this setting should prevent Samba from publishing my printers, it still did! Oh Boy -- it cost me quite some time to find out the reason. But I am not fooled any more... at least not by this ;-)

```
kde-bitshop:/etc/samba # grep -A1 "load printers" smb.conf
       load printers = No
       # This setting is what I mean!!
#      load printers = Yes
```

```
       # This setting is commented oooouuuuut!!

kde-bitshop:/etc/samba # testparm -v smb.conf.simpleprinting | egrep "(load printers)"
       load printers = No
```

Only if setting the parameter explicitly to "***load printers = No***" would Samba recognize my intentions. So my strong advice is:

- Never rely on "commented out" parameters!
- Always set it up explicitly as you intend it to behave.
- Use "testparm" to uncover hidden settings which might disturb your intentions.

You can have a working Samba print configuration with this minimal smb.conf:

```
kde-bitshop:/etc/samba # cat /etc/samba/smb.conf-minimal
       [printers]
```

This example should show you, that you can use testparm to test any filename for fitness as a Samba configuration. Actually, we want to encourage you to *not* change your smb.conf on a working system (unless you know exactly what you do)! Don't rely on an assumption that changes will only take effect after you re-start smbd! This is not the case. Samba re-reads its smb.conf every 60 seconds and on each new client connection. You might have to face changes for your productive clients you didn't mean to apply at this time! You will now note a few more interesting things. Just ask *testparm* what the Samba print configuration would be, if you used this minimalistic file as your real smb.conf:

```
kde-bitshop:~ # testparm -v /etc/samba/smb.conf-minimal | egrep "(print|lpq|spool|driver|ports|[)"
Processing section "[printers]"
WARNING: [printers] service MUST be printable!
No path in service printers - using /tmp

       lpq cache time = 10
       total print jobs = 0
       load printers = Yes
       printcap name = /etc/printcap
       disable spoolss = No
       enumports command =
       addprinter command =
       deleteprinter command =
       show add printer wizard = Yes
       os2 driver map =
       printer admin =
       min print space = 0
       max print jobs = 1000
       printable = No
       printing = bsd
       print command = lpr -r -P%p %s
       lpq command = lpq -P%p
       printer name =
       use client driver = No
[printers]
       printable = Yes
```

testparm issued 2 warnings,

- because we didn't specify the [printers] section as printable, and
- because we didn't tell it which spool directory to use.

However, this was not fatal, and Samba-3.0 will default to values that will work here. But, please!, don't rely on this and don't use this! This example was only meant to make you carefully design and spell out your setup what you really want it to be. The outcome on your system may vary on some parameters, since you may have a Samba built with a different compile time configuration. ***[WARNING: Don't put a comment sign *at the end* of a valid smb.conf line. It will make it to be ignored (just as if you had put it on the front). This I at first regarded as a bug in my Samba version(s). But the man page says: "Internal whitespace in a parameter value is retained verbatim." - This means a line consisting of "printing =lpring #This defines LPRng as the printing system" will regard the whole of the string after the "=" sign as the value you want to define. And this is an invalid value that will be ignored, and a default value used instead.]***

6.4. Extended sample Configuration to print with Samba 3.0

Here we show a more verbose example configuration for print related settings in an smb.conf. Below is a discussion and explanation of the various parameters. We chose to use BSD−style printing here, because we guess it is still the most commonly used system on legacy Linux installations. (New installs now predominantly have CUPS, which is discussed entirely in the next section of this document). Note, that this example specifically names many parameters which don't need to be −− because they would be used anyway "by default". You might be able to do with a leaner smb.conf. (Tip: if you read access it with "SWAT", and then write it to disk again, it will be optimized in a way such that it doesn't contain any superfluous parameters and comments. SWAT organizes the file for best performance. Remember, that each smbd re−reads the Samba configuration once a minute, and that each connection spawns an smbd process of its own −− so it is not a bad idea to optimize the smb.conf in environments with hundreds or thousands of clients...).

```
[global]
        printing = bsd
        load printers = yes
        show add printer wizard = yes
        printcap name = /etc/printcap
        printer admin = @ntadmin, root
        total print jobs = 100
        lpq cache time = 20
        use client driver = no

[printers]
        comment = All Printers
        printable = yes
        path = /var/spool/samba
        browseable = no
        guest ok = yes
        public = yes
        read only = yes
        writable = no

[my_printer_name]
        comment = Printer with Restricted Access
        path = /var/spool/samba_my_printer
        printer admin = kurt
        browseable = yes
        printable = yes
        writeable = no
        hosts allow = 0.0.0.0
        hosts deny = turbo_xp, 10.160.50.23, 10.160.51.60
        guest ok = no
```

This *also* is only an example configuration. For the [my_printer_name] service it defines a different "printer admin" than is defined in the [global] section, and a differfent spool path. Additionally, it includes a parameter which denies access to 3 hosts. You may not find all the settings in your own smb.conf (as pre−configured by your OS vendor). Many configuration parameters, if not explicitly set to a specific value, are used and set by Samba implicitly to its own default, because these have been compiled in. To see all settings, let root use the "testparm" utility. "testparm" also gives warnings if you have mis−configured certain things.

6.5. Detailed Explanation of the Example's Settings

Following is a discussion of the settings from above shown example.

6.5.1. The [global] Section

The [global] section is one of 4 special sections (along with [homes], [printers] and [print$]...) It contains all parameters which apply to the server as a whole. It is the place for parameters which have only a "global" meaning (G). It may also contain service level parameters (S) which then define settings for all other sections and shares. This way you can simplify the configuration and forego to set the same value repeatedly. (Within each individual section or share you may however override these globally set "share level" settings and specify other values).

printing = bsd

this sets Samba to use default print commands applicable for the BSD (a.k.a. RFC 1179 style or LPR/LPD) printing system. In general, the "printing" parameter informs Samba about the print subsystem it should expect. Samba supports CUPS, LPD, LPRNG, SYSV, HPUX, AIX, QNX and PLP. Each of these systems defaults to a different "print command" (and other queue control commands). (Note, that the "printing" parameter is normally a service level parameter. Since it is included here in the [global] section, it will take effect for all printer shares that are not

defined differently). Samba−3.0 does no longer support the SOFTQ printing system.

load printers = yes

this tells Samba to create automatically all available printer shares. "Available" printer shares are discovered by scanning the printcap file. All created printer shares are also loaded for browsing. If you use this parameter, it is not required to specify separate shares for each printer. Each automatically created printer share will clone the configuration options found in the [printers] section. (A "load printers = no" setting will allow you to specify each UNIX printer you want to share separately, leaving out some you don't want to be publicly visible and available).

show add printer wizard = yes

this setting is normally enabled by default (even if the parameter is not written into the smb.conf). It makes the "Add Printer Wizard" icon show up in the "Printers" folder of the Samba host's share listing (as shown in "Network Neighbourhood" or by the "net view" command). To disable it, you need to explicitly set it to "no" (commenting it out will not suffice!). The Add Printer Wizard lets you upload printer drivers to the [print$] share and associate it with a printer (if the respective queue exists there before the action), or exchange a printer's driver against any other previously uploaded driver.

total print jobs = 100

this setting sets the upper limit to 100 print jobs being active on the Samba server at any one time. Should a client submit a job which exceeds this number, a "no more space available on server" type of error message will be returned by Samba to the client. A setting of "0" (the default) means there is *no* limit at all!

printcap name = /etc/printcap

this tells Samba where to look for a list of available printer names. (If you use CUPS, make sure that a printcap file is written: this is controled by the "Printcap" directive of cupsd.conf).

printer admin = @ntadmin

members of the ntadmin group should be able to add drivers and set printer properties ("ntadmin" is only an example name −− it needs to be a valid UNIX group name); root is implicitly always a 'printer admin'. The "@" sign preceeds group names in smb.conf. "printer admins" can do anything to printers via the remote administration interfaces offered by MS−RPC (see below). Note that the "printer admin" parameter is normally a share level parameter, so you may associate different groups to different printer shares in larger installations, if you use the "printer admin" parameter on the share levels).

lpq cache time = 20

this controls the cache time for the results of the lpq command. It prevents the lpq command to be called too often and takes load from a heavily frequented print server.

use client driver = no

if set to "yes", this setting only takes effect for Win NT/2k/XP clients (and not for Win 95/98/ME). Its default value is "No" (or "False"). It must NOT be enabled on print shares (with a "yes" or "true" setting) which have valid drivers installed on the Samba server! For more detailed explanations see the man page of smb.conf.

6.5.2. The [printers] Section

This is the second special section. If a section with this name appears it the smb.conf, users are able to connect to any printer specified in the Samba host's printcap file, because Samba on startup then creates a printer share for every printername it finds in the printcap file. You could regard this section as a general convenience shortcut to share all printers with minimal configuration. It is also a container for settings which should apply as default to all printers. (For more details see the smb.conf man page.) Settings inside this container must be share level parameters (S).

comment = All printers

the "comment" is shown next to the share if a client queries the server, either via "Network Neighbourhood" or with the "net view" command to list available shares.

printable = yes

please note well, that the [printers] service ***must*** be declared as printable. If you specify otherwise, smbd will refuse to load smb.conf at startup. This parameter allows connected clients to open, write to and submit spool files into the directory specified with the "path" parameter for this service. It is used by Samba to differentiate printer shares from file shares.

path = /var/spool/samba

this must point to a directory used by Samba to spool incoming print files. *It must **not** be the same as the spool directory specified in the configuration of your UNIX print subsystem!* The path would typically point to a directory which is world writeable, with the "sticky" bit set to it.

browseable = no

this is always set to "no" if "printable = yes". It makes the [printer] share itself invisible in the list of available shares in a "net view" command or in the Explorer browse list. (Note that you will of course see the individual printers).

guest ok = yes

if set to "yes", then no password is required to connect to the printers service. Access will be granted with the privileges of the "guest account". On many systems the guest account will map to a user named "nobody". This user is in the UNIX passwd file with an empty password, but with no valid UNIX login. (Note: on some systems the guest account might not have the privilege to be able to print. Test this by logging in as your guest user using "su −

guest" and run a system print command like "*lpr −P printername /etc/motd*"....)

public = yes
: this is a synonym for "guest ok = yes". Since we have "guest ok = yes", it really doesn't need to be here! (This leads to the interesting question: "What, if I by accident have to contradictory settings for the same share?" −− The answer is: "The last one encountered by Sambe wins:" The "winner" is shown by testparm. Testparm doesn't complain about different settings of the same parameter for the same share! You can test this by setting up mulitple lines for the "guest account" parameter with different usernames, and then run testparm to see which one is actually used by Samba.)

read only = yes
: this normally (for other types of shares) prevents users to create or modify files in the service's directory. However, in a "printable" service, it is ALWAYS allowed to write to the directory (if user privileges allow connection), but only via print spooling operations. "Normal" write operations are not allowed.

writeable = no
: synonym for "read only = yes"

6.5.3. Any [my_printer_name] Section

If a section appears in the smb.conf, which is tagged as "***printable = yes***", Samba presents it as a printer share to its clients. Note, that Win95/98/ME clients may have problems with connecting or loading printer drivers if the share name has more than 8 characters! Also be very carefull if you name a printer the same as an existing user or file share name: Upon a client's connection request to a certain sharename, Samba always tries to find file shares with that name first; if it finds one, it will connect to this and will never come round and attempt to connect to a printer with the same name!

comment = Printer with Restricted Access
: the comment says it all.

path = /var/spool/samba_my_printer
: here we did set the spooling area for this printer to another directory than the standard. It is not a requirement to set it differently, but an option.

printer admin = kurt
: the printer admin definition is different for this explicitly defined printer share from the general convencience [printers] share. It is not a requirement; we did it to show that it is possible if you want it.

browseable = yes
: we also made this printer browseable (for the clients' convenience to find it when browsing the Network Neighbourhood).

printable = yes
: see explanation in last subsection.

writeable = no
: see explanation in last subsection.

hosts allow = 10.160.50., 10.160.51.
: here we exercise a certain degree of access control by using the "hosts allow" and "hosts deny" parameters. Note, that this is not by any means a safe bet. It is not a way to secure your printers. This line accepts all clients from a certain subnet in a first evaluation of access control

hosts deny = turbo_xp, 10.160.50.23, 10.160.51.60
: all listed hosts are not allowed here (even if they belong to the "allowed subnets"). As you can see, you could name IP addresses as well as NetBIOS hostnames here.

guest ok = no
: this printer is not open for the guest account!

6.5.4. Print Commands

In each section defining a printer (or in the [printers] section), a "***print command***" parameter may be defined. It sets a command to process the files which have been placed into the Samba print spool directory for that printer. (That spool directory was, remember?, set up with the "path" parameter). Typically, this command will submit the spool file to the Samba host's print subsystem, using the suitable system print command. But there is no requirement that this needs to be the case. For debugging purposes or some other reason you may want to do something completely different than "print" the file. An example is that your command just copies it to a save place for further investigation when you need to debug printing. If you craft your own print commands (or even print command shell scripts), make sure you pay attention to also remove the files from the Samba spool directory. Otherwise your hard disk may soon suffer from shortage of free space.

6.5.5. Default Print Commands for various Unix Print Subsystems

You learned earlier on, that Samba in most cases uses its built−in settings for many parameters if it can not find an explicitly stated one in its configuration file. The same is true for the "*print command* ". The default print command varies depending on the "*printing =..."* parameter setting. You will notice some funny "*%<LETTER>*" notions in all the commands beneath. These stand for "printername", "spoolfile" and "job ID". They are explained in more detail further below. Here is an

overview (excluding the special case CUPS, which is discussed in the next chapter):

```
If this setting is active...        ...this is used in lieu of an explicitly set command:
----------------------------        -----------------------------------------------------

printing = bsd|aix|lprng|plp   -->  print command is "lpr -r -P%p %s"
printing = sysv|hpux           -->  print command is "lp -c -P%p %s; rm %s"
printing = qnx                 -->  print command is "lp -r -P%p -s %s"


printing = bsd|aix|lprng|plp   -->  lpq command is "lpq -P%p"
printing = sysv|hpux           -->  lpq command is "lpstat -o%p"
printing = qnx                 -->  lpq command is "lpq -P%p"


printing = bsd|aix|lprng|plp   -->  lprm command is "lprm -P%p %j"
printing = sysv|hpux           -->  lprm command is "cancel %p-%j"
printing = qnx                 -->  lprm command is "cancel %p-%j"


printing = bsd|aix|lprng|plp   -->  lppause command is "lp -i %p-%j -H hold"
printing = sysv|hpux           -->  lppause command    (...is empty)
printing = qnx                 -->  lppause command    (...is empty)


printing = bsd|aix|lprng|plp   -->  lpresume command is "lp -i %p-%j -H resume"
printing = sysv|hpux           -->  lpresume command    (...is empty)
printing = qnx                 -->  lpresume command    (...is empty)
```

We excluded the special CUPS case here, because it is discussed in the next chapter. Just a short summary. For *printing = CUPS*: If SAMBA is compiled against libcups, it uses the CUPS API to submit jobs, etc. (Please set also "*printcap = cups*" in case your *cupsd.conf* is set to write its autogenerated printcap file to an unusual place). Otherwise it maps to the System V commands with the -oraw option for printing, i.e. it uses `lp -c -d%p -oraw; rm %s` . With *printing = cups* , and if SAMBA is compiled against libcups, any manually set print command will be ignored!

Having listed the above mappings here, you should note that there used to be a ***bug*** in recent 2.2.x versions which prevented the mapping from taking effect. It lead to the "bsd|aix|lprng|plp" settings to take effect for all other systems, for the most important commands (the print command, the lpq command and the lprm command). The lppause command and the lpresume command remained empty. (Of course, these commands worked on bsd|aix|lprng|plp -- but they didn't work on sysv|hpux|qnx systems.) To work around this bug, you need to explicitly set the commands. Use "testparm -v" to check which command takes effect. Then check, if this command is adaequate and actually works for your installed print subsystem. It is always a good idea to explicitly set configuration files up the way you want them to work and not rely on any built-in defaults.

6.5.6. Setting up your own Print Commands

After a print job has finished spooling to a service, the *print command* will be used by Samba via a *system()* call to process the spool file. Usually the command specified will submit the spool file to the host's printing subsystem. But there is no requirement at all that this must be the case. The print subsystem will probably not remove the spool file on its own. So whatever command you specify on your own -- you should think to delete the spool file when it has been processed.

Using your own customized print commands is no problem with the traditional printing systems. However, if you don't bother to "roll your own", you should know which are the default built-in commands Samba uses for each printing subsystem (see table above). In all the commands listed in the last paragraphs you see these funny "*%<LETTER>*" notions. These are *macros*, or shortcuts, used as place holders for the names of real objects. At the time of running a command with such a placeholder, Samba will insert the appropriate value automatically. Print commands can handle all Samba macro substitutions. In regard to printing, the following ones do have special relevance:

- `%s, %f` - the path to the spool file name
- `%p` - the appropriate printer name
- `%J` - the job name as transmitted by the client.
- `%c` - the number of printed pages of the spooled job (if known).
- `%z` - the size of the spooled print job (in bytes)

The print command MUST contain at least one occurrence of `%s` or `%f`. -- The `%p` is optional. If no printer name is supplied, the `%p` will be silently removed from the print command. In this case the job is sent to the default printer.

If specified in the [global] section, the print command given will be used for any printable service that does not have its own print command specified. If there is neither a specified print command for a printable service nor a global print command, spool files will be created but not processed! And (most importantly): print files will not be removed, so they will start filling your Samba hard disk.

Note that printing may fail on some UNIXes from the "nobody" account. If this happens: create an alternative guest account and supply it with the privilege to print; set up this guest account in the [global] section with the "***guest account***" parameter.

You can form quite complex print commands. You need to realize that print commands are just passed to a UNIX shell. The shell is able to expand the included environment variables as usual. (The syntax to include a UNIX environment variable *$variable* in smb.conf or in the Samba print command is "*%$variable* ".) To give you a working "print command" example, the following will log a print job to "/tmp/print.log", print the file, then remove it. Note that ';' is the usual separator for commands in shell scripts:

```
print command = echo Printing %s >> /tmp/print.log; lpr -P %p %s; rm %s
```

You may have to vary your own command considerably from this example depending on how you normally print files on your system. The default for the "print command" parameter varies depending on the setting of the "printing" parameter. −− Another example

```
print command = /usr/local/samba/bin/myprintscript %p %s
```

6.6. Innovations in Samba Printing since 2.2

Before version 2.2.0, Samba's print server support for Windows clients was limited to the level of "*LanMan*" printing calls. This is the same protocol level as Windows 9x PCs offer when they share printers. Beginning with the 2.2.0 release, Samba started to support the native Windows NT printing mechanisms. These are implemented via *MS−RPC* (RPC = *Remote Procedure Calls*). MS−RPCs use the *SPOOLSS* named pipe for all printing.

The additional functionality provided by the new SPOOLSS support includes:

- Support for downloading printer driver files to Windows 95/98/NT/2000 clients upon demand ("*Point'n'Print*");
- Uploading of printer drivers via the Windows NT *Add Printer Wizard* (APW) or the *Imprints* tool set (refer to *http://imprints.sourceforge.net*);
- Support for the native MS−RPC printing calls such as StartDocPrinter, EnumJobs(), etc... (See the MSDN documentation at *http://msdn.microsoft.com/* for more information on the Win32 printing API);
- Support for NT *Access Control Lists* (ACL) on printer objects;
- Improved support for printer queue manipulation through the use of internal databases for spooled job information (implemented by various **.tdb* files).

One other benefit of an update is this: Samba 3.0 is able to publish all its printers in Active Directory (or LDAP)!

One slight difference is here: it is possible on a Windows NT print server to have printers listed in the Printers folder which are *not* shared. Samba does not make this distinction. By definition, the only printers of which Samba is aware are those which are specified as shares in *smb.conf*. The reason is that Windows NT/2k/XPprof clients do normally not need and use the standard SMB printer share; they rather print directly to any printer on another Windows NT host using MS−RPC. This of course assumes that the printing client has the necessary privileges on the remote host serving the printer. The default permissions assigned by Windows NT to a printer gives the "Print" permissions to the well−known "*Everyone*" group. (The older clients of type Win9x can only print to "shared" printers).

6.6.1. Client Drivers on Samba Server for *Point'n'Print*

There is still confusion about what all this means: *Is it or is it not a requirement for printer drivers to be installed on a Samba host in order to support printing from Windows clients?* The answer is this: *No, it is not a *requirement**. Windows NT/2000 clients can, of course, also run their APW to installing drivers *locally* (which then connect to a Samba served print queue). This is the same method as used by Windows 9x clients. (However, a *bug* existed in Samba 2.2.0 which made Windows NT/2000 clients require that the Samba server possess a valid driver for the printer. This was fixed in Samba 2.2.1...).

But it is a new **option** to install the printer drivers into the [print$] share of the Samba server, and a big convenience too. Then *all* clients (including 95/98/ME) get the driver installed when they first connect to this printer share. The *uploading* or *depositing* of the driver into this [print$] share, and the following binding of this driver to an existing Samba printer share can be achieved by different means:

- running the *APW* on an NT/2k/XPprof client (this doesn't work from 95/98/ME clients);

- using the *Imprints* toolset;
- using the *smbclient* and *rpcclient* commandline tools;
- using *cupsaddsmb*"(only works for the CUPS printing system, not for LPR/LPD, LPRng etc.).

Please take additional note of the following fact: *Samba does not use these uploaded drivers in any way to process spooled files*. Drivers are utilized entirely by the clients, who download and install them via the "Point 'n'Print" mechanism supported by Samba. The clients use these driver to generate print files in the format the printer (or the Unix print system) requires. Print files received by Samba are handed over to the Unix printing system, which is responsible for all further processing, if needed.

6.6.2. [printer$] Section is removed

[print$] vs. [printer$]

Versions of Samba prior to 2.2 made it possible to use a share named *[printer$]*. This name was taken from the printer$ service created by Windows 9x clients when a printer was shared by them. Windows 9x printer servers always have a printer$ service which provides read−only access (with no password required) in order to support printer driver downloads.

However, Samba's initial implementation allowed for a parameter named "*printer driver location*" to be used on a per share basis. This specified the location of the driver files associated with that printer. Another parameter named "*printer driver*" provided a means of defining the printer driver name to be sent to the client.

These parameters, including the "*printer driver file*" parameter, are now removed and can not be used in installations of Samba−3.0.

Now the share name *[print$]* is used for the location of downloadable printer drivers. It is taken from the print$ service created by Windows NT PCs when a printer is shared by them. Windows NT print servers always have a print$ service which provides read−write access (in the context of its ACLs) in order to support printer driver down− and uploads.

Don't fear −− this does not mean Windows 9x clients are thrown aside now. They can use Samba's [print$] share support just fine.

6.6.3. Creating [print$]

In order to support the up− and downloading of printer driver files, you must first configure a file share named [print$]. The "public" name of this share is hard coded in Samba's internals (because it is hardcoded in the MS Windows clients too). It can't be renamed since Windows clients are programmed to search for a service of exactly this name if they want to retrieve printer driver files.

You should modify the server's smb.conf file to add the global parameters and create the [print$] file share (of course, some of the parameter values, such as 'path' are arbitrary and should be replaced with appropriate values for your site):

```
[global]
     ; members of the ntadmin group should be able to add drivers and set
     ; printer properties. root is implicitly always a 'printer admin'.
     printer admin = @ntadmin
     [....]

[printers]
     [....]

[print$]
     comment = Printer Driver Download Area
     path = /etc/samba/drivers
     browseable = yes
     guest ok = yes
     read only = yes
     write list = @ntadmin, root
```

Of course, you also need to ensure that the directory named by the "***path***" parameter exists on the Unix file system.

6.6.4. Parameters in the [print$] Section

[print$] is a special section in smb.conf. It contains settings relevant to potential printer driver download and local installation by clients.

comment = Printer Driver Download Area

the comment appears next to the share name if it is listed in a share list. (Usual Windows clients won't see it often −− but it also shows up in a "*smbclient −L sambaserver*" output.)

path = /etc/samba/printers

this is the path to the location of the Windows driver file deposit from the UNIX point of view.

browseable = no

this makes the [print$] share "invisible" in Network Neighbourhood to clients. However, you can still "mount" it from any client using the "*net use g: \\sambaserver\print$"* command in a "DOS box" or the "Connect network drive" menu from Windows Explorer.

guest ok = yes

this allows all guest users read only access to this share. Access may be used to download and install printer drivers on clients. The requirement for "*guest ok = yes*" depends upon how your site is configured. If users will be guaranteed to have an account on the Samba host, then this is a non−issue. (**Note:** The non−issue is this: if all your Windows NT users are guaranteed to be authenticated by the Samba server (such as a domain member server and the NT user has already been validated by the Domain Controller in order to logon to the Windows NT session), then guest access is not necessary. Of course, in a workgroup environment where you just want to be able to print without worrying about silly accounts and security, then configure the share for guest access. You'll probably want to add *map to guest = Bad User* in the [global] section as well. Make sure you understand what this parameter does before using it.)

read only = yes

as we don't want everybody to upload driver files (or even change driver settings) we tagged this share as not writeable.

write list = @ntadmin, root

since the [print$] was made read only by the previous setting, we need to create a "write list" also. UNIX users and groups (denoted with a leading "@" character) listed here are allowed write access (as an exception to the general public's "read−only" access), which they need to update files on the share. Normally you will want to only name administrative level user accounts in this setting. Check the file system permissions to make sure these accounts can copy files to the share. If this is setup to a non−root account, then it should also exist as a "*printer admin*". See the *smb.conf* man page for more information on configuring file shares.

6.6.5. Subdirectory Structure in [print$]

In order for a Windows NT print server to support the downloading of driver files by multiple client architectures, you must create several subdirectories within the [print$] service (i.e. the Unix directory named by the "***path***" parameter). These correspond to each of the supported client architectures. Samba follows this model as well. Just like the name of the [print$] share itself, the subdirectories *must* be exactly the names listed below. (You may leave out the subdirs of architectures you don't want to support).

Therefore, create a directory tree below the [print$] share for each architecture you wish to support.

```
[print$]--+--
          |--W32X86           # serves drivers to "Windows NT x86"
          |--WIN40            # serves drivers to "Windows 95/98"
          |--W32ALPHA         # serves drivers to "Windows NT Alpha_AXP"
          |--W32MIPS          # serves drivers to "Windows NT R4000"
          |--W32PPC           # serves drivers to "Windows NT PowerPC"
```

ATTENTION! REQUIRED PERMISSIONS

In order to currently add a new driver to you Samba host, one of two conditions must hold true:

- The account used to connect to the Samba host must have a UID of 0 (i.e. a root account)
- The account used to connect to the Samba host must be named in the *printer admin* list.

Of course, the connected account must still possess access to add files to the subdirectories beneath [print$]. Remember that all file shares are set to 'read only' by default.

Once you have created the required [print$] service and associated subdirectories, go to a Windows NT 4.0/2k/XP client workstation. Open "Network Neighbourhood" or "My Network Places" and browse for the Samba host. Once you have located the server, navigate to its "*Printers and Faxes*" folder. You should see an initial listing of printers that matches the printer shares defined on your Samba host.

6.7. Installing Drivers into [print$]

You have successfully created the [print$] share in smb.conf? And Samba has re−read its configuration? Good. But you are not yet ready to take off. The *driver files* need to be *present* in this share, too! So far it is still an empty share. Unfortunatly,

it is not enough to just copy the driver files over. They need to be *set up* too. And that is a bit tricky, to say the least. We will now discuss two alternative ways to install the drivers into [print$]:

- using the Samba commandline utility *rpcclient* with its various subcommands (here: *adddriver* and *setdriver*) from any UNIX workstation;
- runing a GUI ("*Printer Properties*" and "*Add Printer Wizard*") from any Windows NT/2k/XP client workstation.

The latter option is probably the easier one (even if the only entrance to this realm seems a little bit weird at first).

6.7.1. Setting Drivers for existing Printers with a Client GUI

The initial listing of printers in the Samba host's " *Printers*" folder accessed from a client's Explorer will have no real printer driver assigned to them. By default, in Samba 3.0 (as in 2.2.1 and later) this driver name is set to a NULL string. This must be changed now. the local *Add Printer Wizard,* run from NT/2000/XP clients, will help us in this task.

However, the job to set a valid driver for the printer is not a straightforward one: You must attempt to view the printer properties for the printer you want the driver assign to. Open the Windows Explorer, open Network Neighbourhood, browse to the Samba host, open Samba's "Printers" folder, right−click the printer icon and select "Properties...". You are now trying to view printer and driver properties for a queue which has this default "NULL" driver assigned. This will result in an error message (this is "normal" here):

Device settings cannot be displayed. The driver for the specified printer is not installed, only spooler properties will be displayed. Do you want to install the driver now?

Don't click "Yes"! Instead, ***click "No"*** in the error dialog. Only now you will be presented with the printer properties window. From here, the way to assign a driver to a printer is open to us. You have now the choice. Either

- select a driver from the popup list of installed drivers. *Initially this list will be empty.* Or
- use the *"New Driver..."* button to install a new printer driver (which will in fact start up the APW).

Once the APW is started, the procedure is exactly the same you are used to from Wiindows. We assume here that you are familiar with the printer driver installations procedure on Windows NT. Make sure your connection is setup in fact as a user with *printer admin* privileges (if in doubt, use "smbstatus" to check for this). If you wish to install printer drivers for client operating systems other than "Windows NT x86", you will need to use the *"Sharing"* tab of the printer properties dialog.

Assuming you have connected with an administrative (or root) account (as named by the "***printer admin***" parameter), you will also be able to modify other printer properties such as ACLs and default device settings using this dialog. (For the default device settings, please consider the advice given further below). [FIXME: This section should sometime in the near future be updated and enhanced and include some screenshots from the process. FIXME]

6.7.2. Setting Drivers for existing Printers with *rpcclient*

The second way to install printer drivers into [print$] and set them up in a valid way can be done from the UNIX command line. This involves three (or four, if you count "zero" too ;−) distinct steps:

- 0. −− gathering the info about the required driver files and collecting the files together;
- 1. −− deposit the driver files into the [print$] share's correct subdirectories (possibly by using *smbclient*);
- 2. −− running the *rpcclient* commandline utility once with the *addriver* subcommand,
- 3. −− running *rpcclient* a second time with the *setdriver* subcommand.

We will provide detailed hints for each of these steps in the next few paragraphs.

6.7.2.1. Identifying the Driver Files

To find out about the driver files, you have two options: you could investigate the driver CD which comes with your printer. Study the **.inf* file on the CD, if it is contained. This may not be the possible, since the *.inf file might be missing. Unfortunately, many vendors have now started to use their own installation programs. These installations packages are often some sort of Windows platform archive format, plus, the files may get re−named during the installation process. This makes it extremely difficult to identify the driver files you need.

Then you only have the second option: install the driver first on a Windows client *locally* and investigate which file names and paths it uses after they are installed. (Note, that you need to repeat this procedure for every client platform you want to support. We are going to show it here for the "*W32X86*" platform only, a name used by Microsoft for all WinNT/2k/XP clients...)

A good method to recognize the driver files this is to print the test page from the driver's *Properties* Dialog (*General* tab). Then look at the list of driver files named on the printout. You'll need to recognize what Windows (and Samba) are calling the "*Driver File*", the "*Data File*", the "*Config File*", the "*Help File*" and (optionally) the "*Dependent Driver Files*" (this may vary slightly for Windows NT). (FIXME: Attention −− I've translated this back from German: please someone verify the exact names of these file categories on an English version of Windows...FIXME). You need to remember all names (or better take a note) for the next steps.

Another method to quickly test the driver filenames and related paths is provided by the *rpcclient* utility. Run it with the "*enumdrivers*" or with the "*getdriver*" subcommand, each in the "*3*" level. In the following example, *TURBO_XP* is the name of the Windows PC (in this case it was a Windows XP Professional laptop, BTW). I had installed the driver locally to TURBO_XP. *kde−bitshop* is the name of the Linux host from which I am working. We could run an *interactive* rpcclient session; then we'd get an *rpcclient>* prompt and would type the subcommands at this prompt. This is left as a good exercise to the reader. For now we use rpcclient with the *"−c*" parameter to execute a single subcommand line and exit again. This is the method you would use if you want to create scripts to automate the procedure for a large number of printers and drivers. Note the different quotes used to overcome the different spaces in between words:

```
kde−bitshop:~# rpcclient −U'Danka%xxxx' −c 'getdriver "Heidelberg Digimaster 9110 (PS)" 3' TURBO_XP
cmd = getdriver "Heidelberg Digimaster 9110 (PS)" 3

[Windows NT x86]
Printer Driver Info 3:
        Version: [2]
        Driver Name: [Heidelberg Digimaster 9110 (PS)]
        Architecture: [Windows NT x86]
        Driver Path: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\HDNIS01_de.DLL]
        Datafile: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\Hddm91c1_de.ppd]
        Configfile: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\HDNIS01U_de.DLL]
        Helpfile: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\HDNIS01U_de.HLP]

        Dependentfiles: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\Hddm91c1_de.DLL]
        Dependentfiles: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\Hddm91c1_de.INI]
        Dependentfiles: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\Hddm91c1KMMin.DLL]
        Dependentfiles: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\Hddm91c1_de.dat]
        Dependentfiles: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\Hddm91c1_de.cat]
        Dependentfiles: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\Hddm91c1_de.def]
        Dependentfiles: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\Hddm91c1_de.hre]
        Dependentfiles: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\Hddm91c1_de.vnd]
        Dependentfiles: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\Hddm91c1_de.hlp]
        Dependentfiles: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\Hddm91c1_de_reg.HLP]
        Dependentfiles: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\HDNIS01Aux.dll]
        Dependentfiles: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\HDNIS01_de.NTF]

        Monitorname: []
        Defaultdatatype: []
```

You may notice, that this driver has quite a big number of "Dependentfiles" (I know worse cases however). Also, strangely, the "*Driver File*" is here tagged as "*Driver Path*".... oh, well. Here we don't have yet support for the so−called "WIN40" architecture installed. This name is used by Microsoft for the Win95/98/ME platforms. If we want to support these, we need to install the Win95/98/ME driver files in addition to those for W32X86 (i.e. the WinNT72000/XP clients) onto a Windows PC. This PC can also host the Win9x drivers, even if itself runs on Windows NT, 2000 or XP.

Since the "print$" share is usually accessible through the Network Neighbourhood, you can also use the UNC notation from Windows Explorer to poke at it. The Win9x driver files will end up in subdirectory "0" of the "WIN40" directory. The full path to access them will be "*\\WINDOWSHOST\print$\WIN40\0*".

Note that more recent drivers on Windows 2000 and Wndows XP are installed into the "3" subdirectory instead of the "2". The version 2 of drivers, as used in Windows NT, were running in Kernel Mode. Windows 2000 changed this. While it still can use the Kernel Mode drivers (if this is enabled by the Admin), its native mode for printer drivers is User Mode execution. This requires drivers designed for this. These type of drivers install into the "3" subdirectory.

6.7.2.2. Collecting the Driver Files from a Windows Host's [print$] Share

Now we need to collect all the driver files we identified. in our previous step. Where do we get them from? Well, why not retrieve them from the very PC and the same [print$] share which we investigated in our last step to identify the files? We can use *smbclient* to do this. We will use the paths and names which were leaked to us by "*getdriver*". The listing is edited to include linebreaks for readability:

```
kde−bitshop:~# smbclient //TURBO_XP/print\$ −U'Danka%xxxx'                          \
                                            −c 'cd W32X86/2;mget HD*_de.*          \
```

```
                                              hd*ppd Hd*_de.* Hddm*dll HDN*Aux.DLL'
added interface ip=10.160.51.60 bcast=10.160.51.255 nmask=255.255.252.0
Got a positive name query response from 10.160.50.8 ( 10.160.50.8 )
Domain=[DEVELOPMENT] OS=[Windows 5.1] Server=[Windows 2000 LAN Manager]
Get file Hddm91c1_de.ABD? n
Get file Hddm91c1_de.def? y
getting file \W32X86\2\Hddm91c1_de.def of size 428 as Hddm91c1_de.def (22.0 kb/s)     \
                                                                  (average 22.0 kb/s)
Get file Hddm91c1_de.DLL? y
getting file \W32X86\2\Hddm91c1_de.DLL of size 876544 as Hddm91c1_de.DLL (737.3 kb/s) \
                                                                        (average 737.3 kb/s)
[...,]
```

After this command is complete, the files are in our current local directory. You probably have noticed that this time we passed several commands to the "−c" parameter, separated by semi−colons. This effects that all commands are executed in sequence on the remote Windows server before smbclient exits again.

Don't forget to repeat the procedure for the "WIN40" architecture should you need to support Win95/98/XP clients. Remember, the files for these architectures are in the WIN40/0/ subdir. Once we are complete, we can run "*smbclient ... put*" to store the collected files on the Samba server's [print$] share.

6.7.2.3. Depositing the Driver Files into [print$]

So, now we are going to put the driver files into the [print$] share. Remember, the UNIX path to this share has been defined previously in your smb.conf. You also have created subdirectories for the different Windows client types you want to support. Supposing your [print$] share maps to the UNIX path */etc/samba/drivers/*, your driver files should now go here:

- for all Windows NT, 2000 and XP clients into */etc/samba/drivers/W32X86/* −− ***but *not*(yet) into the "2" subdir***!
- for all Windows 95, 98 and ME clients into */etc/samba/drivers/WIN40/* −− ***but *not* (yet) into the "0" subdir***!

We again use smbclient to transfer the driver files across the network. We specify the same files and paths as were leaked to us by running "*getdriver*" against the original *Windows* install. However, now we are going to store the files into a *Samba /UNIX* print server's [print$] share...

```
kde−bitshop:~# smbclient //SAMBA−CUPS/print\$ −U'root%xxxx' −c 'cd W32X86; put HDNIS01_de.DLL; \
                                              put Hddm91c1_de.ppd; put HDNIS01U_de.DLL;        \
                                              put HDNIS01U_de.HLP; put Hddm91c1_de.DLL;        \
                                              put Hddm91c1_de.INI; put Hddm91c1KMMin.DLL;      \
                                              put Hddm91c1_de.dat; put Hddm91c1_de.dat;        \
                                              put Hddm91c1_de.def; put Hddm91c1_de.hre;        \
                                              put Hddm91c1_de.vnd; put Hddm91c1_de.hlp;        \
                                              put Hddm91c1_de_r.HLP; put HDNIS01Aux.dll;       \
                                              put HDNIS01_de.NTF'
added interface ip=10.160.51.60 bcast=10.160.51.255 nmask=255.255.252.0
Got a positive name query response from 10.160.51.162 ( 10.160.51.162 )
Domain=[CUPS−PRINT] OS=[Unix] Server=[Samba 2.2.7a]
putting file HDNIS01_de.DLL as \W32X86\HDNIS01_de.DLL (4465.5 kb/s) (average 4465.5 kb/s)
putting file Hddm91c1_de.ppd as \W32X86\Hddm91c1_de.ppd (12876.8 kb/s) (average 4638.9 kb/s)
putting file HDNIS01U_de.DLL as \W32X86\HDNIS01U_de.DLL (20249.8 kb/s) (average 5828.3 kb/s)
putting file HDNIS01U_de.HLP as \W32X86\HDNIS01U_de.HLP (9652.8 kb/s) (average 5899.8 kb/s)
putting file Hddm91c1_de.DLL as \W32X86\Hddm91c1_de.DLL (23777.7 kb/s) (average 10400.6 kb/s)
putting file Hddm91c1_de.INI as \W32X86\Hddm91c1_de.INI (98.6 kb/s) (average 10329.0 kb/s)
putting file Hddm91c1KMMin.DLL as \W32X86\Hddm91c1KMMin.DLL (22931.5 kb/s) (average 10501.7 kb/s)
putting file Hddm91c1_de.dat as \W32X86\Hddm91c1_de.dat (2462.8 kb/s) (average 10393.0 kb/s)
putting file Hddm91c1_de.dat as \W32X86\Hddm91c1_de.dat (4925.3 kb/s) (average 10356.3 kb/s)
putting file Hddm91c1_de.def as \W32X86\Hddm91c1_de.def (417.9 kb/s) (average 10290.1 kb/s)
putting file Hddm91c1_de.hre as \W32X86\Hddm91c1_de.hre (22571.3 kb/s) (average 11338.5 kb/s)
putting file Hddm91c1_de.vnd as \W32X86\Hddm91c1_de.vnd (3384.6 kb/s) (average 10754.3 kb/s)
putting file Hddm91c1_de.hlp as \W32X86\Hddm91c1_de.hlp (18406.8 kb/s) (average 10839.8 kb/s)
putting file Hddm91c1_de_r.HLP as \W32X86\Hddm91c1_de_r.HLP (20278.3 kb/s) (average 11386.3 kb/s)
putting file HDNIS01Aux.dll as \W32X86\HDNIS01Aux.dll (14994.6 kb/s) (average 11405.2 kb/s)
putting file HDNIS01_de.NTF as \W32X86\HDNIS01_de.NTF (23390.2 kb/s) (average 13170.8 kb/s)
```

Phewww −− that was a lot of typing! Most drivers are a lot smaller −− many only having 3 generic PostScript driver files plus 1 PPD. Note, that while we did retrieve the files from the "2" subdirectory of the "W32X86" directory from the Windows box, we *don't* put them (for now) in this same subdirectory of the Samba box! This re−location will automatically be done by the "*adddriver*" command which we will run shortly. (And don't forget to also put the files for the Win95/98/ME architecture into the *WIN40/* subdirectory should you need them....).

6.7.2.4. Check if the Driver Files are there (with smbclient)

For now we verify that our files are there. This can be done with smbclient too (but of course you can log in via SSH also and do this through a standard UNIX shell access too):

```
kde-bitshop:~# smbclient //SAMBA-CUPS/print\$ -U 'root%xxxx' -c 'cd W32X86; pwd; dir; cd 2; pwd; dir'
added interface ip=10.160.51.60 bcast=10.160.51.255 nmask=255.255.252.0
Got a positive name query response from 10.160.51.162 ( 10.160.51.162 )
Domain=[CUPS-PRINT] OS=[Unix] Server=[Samba 2.2.7a]

 Current directory is \\SAMBA-CUPS\print$\W32X86\
 .                                   D        0  Sun May  4 03:56:35 2003
 ..                                  D        0  Thu Apr 10 23:47:40 2003
 2                                   D        0  Sun May  4 03:56:18 2003
 HDNIS01Aux.dll                      A    15356  Sun May  4 03:58:59 2003
 Hddm91c1KMMin.DLL                   A    46966  Sun May  4 03:58:59 2003
 HDNIS01_de.DLL                      A   434400  Sun May  4 03:58:59 2003
 HDNIS01_de.NTF                      A   790404  Sun May  4 03:56:35 2003
 Hddm91c1_de.DLL                     A   876544  Sun May  4 03:58:59 2003
 Hddm91c1_de.INI                     A      101  Sun May  4 03:58:59 2003
 Hddm91c1_de.dat                     A     5044  Sun May  4 03:58:59 2003
 Hddm91c1_de.def                     A      428  Sun May  4 03:58:59 2003
 Hddm91c1_de.hlp                     A    37699  Sun May  4 03:58:59 2003
 Hddm91c1_de.hre                     A   323584  Sun May  4 03:58:59 2003
 Hddm91c1_de.ppd                     A    26373  Sun May  4 03:58:59 2003
 Hddm91c1_de.vnd                     A    45056  Sun May  4 03:58:59 2003
 HDNIS01U_de.DLL                     A   165888  Sun May  4 03:58:59 2003
 HDNIS01U_de.HLP                     A    19770  Sun May  4 03:58:59 2003
 Hddm91c1_de_reg.HLP                 A   228417  Sun May  4 03:58:59 2003
               40976 blocks of size 262144. 709 blocks available

 Current directory is \\SAMBA-CUPS\print$\W32X86\2\
 .                                   D        0  Sun May  4 03:56:18 2003
 ..                                  D        0  Sun May  4 03:56:35 2003
 ADOBEPS5.DLL                        A   434400  Sat May  3 23:18:45 2003
 laserjet4.ppd                       A     9639  Thu Apr 24 01:05:32 2003
 ADOBEPSU.DLL                        A   109568  Sat May  3 23:18:45 2003
 ADOBEPSU.HLP                        A    18082  Sat May  3 23:18:45 2003
 PDFcreator2.PPD                     A    15746  Sun Apr 20 22:24:07 2003
               40976 blocks of size 262144. 709 blocks available
```

Notice that there are already driver files present in the "2" subdir (probably from a previous installation). Once the files for the new driver are there too, you are still a few steps away from being able to use them on the clients. The only thing you could do *now* is to retrieve them from a client just like you retrieve ordinary files from a file share, by opening print$ in Windows Explorer. But that wouldn't install them per Point'n'Print. The reason is: Samba doesn't know yet that these files are something special, namely *printer driver files* and it doesn't know yet to which print queue(s) these driver files belong....

6.7.2.5. Running *rpcclient* with *adddriver*

So, next you must tell Samba about the special category of the files you just uploaded into the [print$] share. This is done by the "*adddriver*" command. It will prompt Samba to register the driver files into its internal TDB database files. The following command and its output has been edited, again, for readability:

```
kde-bitshop:~# rpcclient -Uroot%xxxx -c 'adddriver "Windows NT x86" "dm9110:HDNIS01_de.DLL: \
                                         Hddm91c1_de.ppd:HDNIS01U_de.DLL:HDNIS01U_de.HLP:   \
                                         NULL:RAW:Hddm91c1_de.DLL,Hddm91c1_de.INI,          \
                                         Hddm91c1_de.dat,Hddm91c1_de.def,Hddm91c1_de.hre,   \
                                         Hddm91c1_de.vnd,Hddm91c1_de.hlp,Hddm91c1KMMin.DLL, \
                                         HDNIS01Aux.dll,HDNIS01_de.NTF,                     \
                                         Hddm91c1_de_reg.HLP'                       SAMBA-CUPS

cmd = adddriver "Windows NT x86" "dm9110:HDNIS01_de.DLL:Hddm91c1_de.ppd:HDNIS01U_de.DLL:    \
                                  HDNIS01U_de.HLP:NULL:RAW:Hddm91c1_de.DLL,Hddm91c1_de.INI, \
                                  Hddm91c1_de.dat,Hddm91c1_de.def,Hddm91c1_de.hre,          \
                                  Hddm91c1_de.vnd,Hddm91c1_de.hlp,Hddm91c1KMMin.DLL,        \
                                  HDNIS01Aux.dll,HDNIS01_de.NTF,Hddm91c1_de_reg.HLP"

Printer Driver dm9110 successfully installed.
```

After this step the driver should be recognized by Samba on the print server. You need to be very carefull when typing the command. Don't exchange the order of the fields. Some changes would lead to a "*NT_STATUS_UNSUCCESSFUL*" error message. These become obvious. Other changes might install the driver files successfully, but render the driver unworkable.

So take care! Hints about the syntax of the adddriver command are in the man page. The CUPS printing chapter of this HOWTO collection provides a more detailed description, if you should need it.

6.7.2.6. Check how Driver Files have been moved after adddriver finished

One indication for Samba's recognition of the files as driver files is the "*successfully installed*" message. Another one is the fact, that our files have been moved by the *adddriver* command into the "*2*" subdirectory. You can check this again with *smbclient*:

```
kde-bitshop:~# smbclient //SAMBA-CUPS/print\$ -Uroot%xxxx -c 'cd W32X86;dir;pwd;cd 2;dir;pwd'
added interface ip=10.160.51.162 bcast=10.160.51.255 nmask=255.255.252.0
Domain=[CUPS-PRINT] OS=[Unix] Server=[Samba 2.2.7a]

 Current directory is \\SAMBA-CUPS\print$\W32X86\
 .                                   D        0  Sun May  4 04:32:48 2003
 ..                                  D        0  Thu Apr 10 23:47:40 2003
 2                                   D        0  Sun May  4 04:32:48 2003
               40976 blocks of size 262144. 731 blocks available

 Current directory is \\SAMBA-CUPS\print$\W32X86\2\
 .                                   D        0  Sun May  4 04:32:48 2003
 ..                                  D        0  Sun May  4 04:32:48 2003
 DigiMaster.PPD                      A   148336  Thu Apr 24 01:07:00 2003
 ADOBEPS5.DLL                        A   434400  Sat May  3 23:18:45 2003
 laserjet4.ppd                       A     9639  Thu Apr 24 01:05:32 2003
 ADOBEPSU.DLL                        A   109568  Sat May  3 23:18:45 2003
 ADOBEPSU.HLP                        A    18082  Sat May  3 23:18:45 2003
 PDFcreator2.PPD                     A    15746  Sun Apr 20 22:24:07 2003
 HDNIS01Aux.dll                      A    15356  Sun May  4 04:32:18 2003
 Hddm91c1KMMin.DLL                   A    46966  Sun May  4 04:32:18 2003
 HDNIS01_de.DLL                      A   434400  Sun May  4 04:32:18 2003
 HDNIS01_de.NTF                      A   790404  Sun May  4 04:32:18 2003
 Hddm91c1_de.DLL                     A   876544  Sun May  4 04:32:18 2003
 Hddm91c1_de.INI                     A      101  Sun May  4 04:32:18 2003
 Hddm91c1_de.dat                     A     5044  Sun May  4 04:32:18 2003
 Hddm91c1_de.def                     A      428  Sun May  4 04:32:18 2003
 Hddm91c1_de.hlp                     A    37699  Sun May  4 04:32:18 2003
 Hddm91c1_de.hre                     A   323584  Sun May  4 04:32:18 2003
 Hddm91c1_de.ppd                     A    26373  Sun May  4 04:32:18 2003
 Hddm91c1_de.vnd                     A    45056  Sun May  4 04:32:18 2003
 HDNIS01U_de.DLL                     A   165888  Sun May  4 04:32:18 2003
 HDNIS01U_de.HLP                     A    19770  Sun May  4 04:32:18 2003
 Hddm91c1_de_reg.HLP                 A   228417  Sun May  4 04:32:18 2003
               40976 blocks of size 262144. 731 blocks available
```

Another verification is that the timestamp of the printing TDB files is now updated (and possibly their filesize has increased).

6.7.2.7. Check if the Driver is recognized by Samba

Now the driver should be registered with Samba. We can easily verify this, and will do so in a moment. However, this *driver* is *not yet* associated with a particular *printer*. We may check the *driver* status of the files by at least three methods:

- from any Windows client browse Network Neighbourhood, finde the Samba host and open the Samba "*Printers and Faxes*" folder. Select any printer icon, right-click and select the printer " *Properties*". Click on the "*Advanced*" tab. Here is a field indicating the driver for that printer. A drop down menu allows you to change that driver. (Be carefull to not do this unwittingly.). You can use this list to view all drivers know to Samba. Your new one should be amongst them. (Each type of client will only see his own architecture's list. If you don't have every driver installed for each platform, the list will differ if you look at it from Windows95/98/ME or WindowsNT/2000/XP.)
- from a Windows 2000 or XP client (not WinNT) browse *Network Neighbourhood*, search for the Samba server and open the server's *Printers* folder, right-click the white background (with no printer highlighted). Select "*Server Properties*". On the "*Drivers*" tab you will see the new driver listed now. This view enables you to also inspect the list of files belonging to that driver. *(This doesn't work on Windows NT, but only on Windows 2000 and Windows XP. WinNT doesn't provide the mentioned "Drivers" tab).* -- An alternative, much quicker method for Windows 2000/XP to start this dialog is by typing into a DOS box (you must of course adapt the name to your Samba server instead of SAMBA-CUPS):
  ```
  rundll32 printui.dll,PrintUIEntry /s /t2 /n\\SAMBA-CUPS
  ```
- from a UNIX prompt run this command (or a variant thereof), where "SAMBA-CUPS" is the name of the Samba host and "xxxx" represents the actual Samba password assigned to root:
  ```
  kde-bitshop:~# rpcclient -U'root%xxxx' -c 'enumdrivers' SAMBA-CUPS
  ```

You will see a listing of all drivers Samba knows about. Your new one should be amongst them. But it is only listed under the [Windows NT x86] heading, not under [Windows 4.0], since we didn't install that part. Or did *you*? −− You will see a listing of all drivers Samba knows about. Your new one should be amongst them. In our example it is named *dm9110*. Note that the 3rd column shows the other installed drivers twice, for each supported architecture one time. Our new driver only shows up for "Windows NT 4.0 or 2000". To have it present for "Windows 95, 98 and ME" you'll have to repeat the whole procedure with the WIN40 architecture and subdirectory. (FIXME: Should I include a screenshot here?? FIXME:)

6.7.2.8. A Sidenote: you are not bound to specific Driver Names

You can name the driver as you like. If you repeat the "*adddriver*" step, with the same files as before, but with a different driver name, it will work the same:

```
kde−bitshop:~# rpcclient −Uroot%xxxx                                    \
                         −c 'adddriver "Windows NT x86"                 \
                         "myphantasydrivername:HDNIS01_de.DLL:          \
                         Hddm91c1_de.ppd:HDNIS01U_de.DLL:HDNIS01U_de.HLP:   \
                         NULL:RAW:Hddm91c1_de.DLL,Hddm91c1_de.INI,      \
                         Hddm91c1_de.dat,Hddm91c1_de.def,Hddm91c1_de.hre,   \
                         Hddm91c1_de.vnd,Hddm91c1_de.hlp,Hddm91c1KMMin.DLL, \
                         HDNIS01Aux.dll,HDNIS01_de.NTF,Hddm91c1_de_reg.HLP' SAMBA−CUPS

cmd = adddriver "Windows NT x86"
                "myphantasydrivername:HDNIS01_de.DLL:Hddm91c1_de.ppd:HDNIS01U_de.DLL:\
                 HDNIS01U_de.HLP:NULL:RAW:Hddm91c1_de.DLL,Hddm91c1_de.INI,      \
                 Hddm91c1_de.dat,Hddm91c1_de.def,Hddm91c1_de.hre,               \
                 Hddm91c1_de.vnd,Hddm91c1_de.hlp,Hddm91c1KMMin.DLL,             \
                 HDNIS01Aux.dll,HDNIS01_de.NTF,Hddm91c1_de_reg.HLP"

Printer Driver myphantasydrivername successfully installed.
```

You will also be able to bind that driver to any print queue. (However, you are responsible yourself that you associate drivers to queues which make sense to the target printer...) Note, that you can't run the rpcclient adddriver command repeatedly. Each run "consumes" the files you had put into the [print$] share by moving them into the respective subdirectories. So you *must* preceed an "*smbclient ... put"* command before each "*rpcclient ... addriver*" command.

6.7.2.9. Le Finale Grande: Running *rpcclient* with *setdriver*

Samba still needs to know *which* printer's driver this is. It needs to create a mapping of the driver to a printer, and store this info in its "memory", the TDB files. The *rpcclient setdriver* command achieves exactly this:

```
kde−bitshop:~# rpcclient −U'root%xxxx' −c 'setdriver dm9110 myphantasydrivername' SAMBA−CUPS
cmd = setdriver dm9110 myphantasydrivername
Succesfully set dm9110 to driver myphantasydrivername.
```

Ahhhhh −− na, didn't want to do that. Repeat, this time with the name I intended:

```
kde−bitshop:~# rpcclient −U'root%xxxx' −c 'setdriver dm9110 dm9110' SAMBA−CUPS
cmd = setdriver dm9110 dm9110
Succesfully set dm9110 to driver dm9110.
```

The syntax of the command is *rpcclient −U'root%sambapassword' −c 'setdriver "printername" "drivername' SAMBA−Hostname* . −− Now we have done *most* of the work. But not yet all....

Note that the "setdriver" command will only succeed, if the printer is known to Samba already. A bug in 2.2.x prevented Samba from recognizing freshly installed printers. You had to restart Samba, or at least send a HUP signal to all running smbd processes to work around this: "*kill −HUP `pidof smbd`*"

6.8. "The Proof for the Pudding lies in the Eating" (Client Driver Insta Procedure)

A famous philosopher said once: "The Proof for the Pudding lies in the Eating". The proof for our setup lies in the printing. So let's install the printer driver onto the client PCs. This is not as straightforward as it may seem. Read on.

6.8.1. The first Client Driver Installation

Especially important is the installation onto the first client PC (for each architectural platform separately). Once this is done right, all further clients are easy and shouldn't need further attention. What follows is a description for the recommended first procedure. You work now from a client workstation. First you should guarantee that your connection is not unwittingly mapped to *bad user* "nobody". In a DOS box type:

```
net use \\SAMBA−SERVER\print$ /user:root
```

Replace root, if needed, by another valid 'printer admin' user as of smb.conf definition. Should you already be connected as a different user, you'll get an error message. There is no easy way to get rid of that connection, because Windows doesn't seem to know a concept of "logging off" from a share connection (don't confuse this with logging off from the local workstation; that is a different matter). You can try to close *all* Windows file explorer and Internet Explorer windows. As a last resort, you may have to reboot. Make sure there is no automatic re−connection set up. (It may be easier to go to a different workstation and try from there...). After you made sure you are connected as a printer admin user (you can check this with the "smbstatus" command on Samba), do this from the Windows workstation:

- Open *Network Neighbourhood*
- Browse to Samba server
- Open its "*Printers and Faxes*" folder
- Highlight and right−click the printer
- Select "*Connect...*" (maybe on WinNT4/2K it is " *Install...* "?)

A new printer (named "*printername on samba−server* ") should now have appeared in your *local* Printer folder (*Start −−> Settings −−> Control Panel −−> Printers and Faxes*). (FIXME: this was translated from German. Someone please tell me the correct words. FIXME) −−

Most likely you are now tempted to try and print a test page. After all, you now can open the printer properties and on the "General" tab, there is a button offering to do just that. But chances are that you get an error message saying "Unable to print Test Page." The reason might be that there is not yet a valid Device Mode set for the driver, or that the "Printer Driver Data" set is still incomplete.

You must now make sure that a valid "Device Mode" is set for the driver. Don't fear −− we will explain now what that means.

6.8.2. IMPORTANT! Setting Device Modes on new Printers

In order for a printer to be truly usable by a Windows NT/2K/XP client, it must possess:

- a valid *Device Mode* generated by the driver for the printer (defining things like paper size, orientation and duplex settings), and
- a complete set of *PrinterDriverData* generated by the driver.

If either one of these is incomplete, the clients can produce less than optimal output at best. In the worst cases, unreadable garbage or nothing at all comes from the printer or they only harvest error messages when attempting to print. Samba stores the named values and all printing related info in its internal TDB database files *(ntprinters.tdb, ntdrivers.tdb, printing.tdb* and *ntforms.tdb*).

What do these two words stand for? Basically, the Device Mode and the set of Printer Driver Data is a collection of settings for all print queue properties, initialized in a sensible way. Device Modes and PrinterDriverData should initially be set on the print server (that is here: the Samba host) to healthy values so that the clients can start to use them immediately. How do we set these initial healthy values? This can be achieved by accessing the drivers remotely from an NT (or 2k/XP) client, as is discussed in the next paragraphs.

Be aware, that a valid Device Mode can only be initiated by a "*printer admin*", or root (the reason should be obvious). Device Modes can only correctly be set by executing the printer driver program itself. Since Samba can not execute this Win32 platform driver code, it sets this field initially to NULL (which is not a valid setting for clients to use). Fortunately, most drivers generate themselves the PrinterDriverData that is needed, when they are uploaded to the [print$] share with the help of the APW or rpcclient.

The generation and setting of a first valid Device Mode however requires some "tickling" from a client, to set it on the Samba server. The easiest means of doing so is to simply change the page orientation on the server's printer. This "executes" enough of the printer driver program on the client for the desired effect to happen, and feeds back the new DeviceMode to our Sanba server. You can use the native Windows NT/2K/XP printer properties page from a Window client for this:

- Browse the "Network Neighbourhood"
- Find the Samba server
- Open the Samba server's "Printers and Faxes" folder
- Highlight the shared printer in question
- Right−click the printer (You may already be here, if you followed the last section's description)
- At the low end of the context menu select "Properties...." (if the menu still offers the "Connect..." entry further above, you need to click that one first to achieve the driver installation as shown in the last section)
- Go to the "Advanced" tab; click on "Printing Defaults..."
- Change the "Portrait" page setting to "Landscape" (and back)
- (Oh, and make sure to *apply* changes between swapping the page orientation to cause the change to actually take effect...).
- While you're at it, you may optionally also want to set the desired printing defaults here, which then apply to all future client driver installations on the remaining from now on.

This procedure has executed the printer driver program on the client platform and fed back the correct Device Mode to Samba, which now stored it in its TDB files. (Once the driver is installed on the client, you can follow the analogous steps by accessing the *local* "Printers" folder too if you are a Samba printer admin user.) From now on printing should work as expected.

Samba also includes a service level parameter name ***default devmode*** for generating a default Device Mode for a printer. Some driver will function fine with Samba's default set of properties. Others may crash the client's spooler service. So use this parameter with caution. It is always better to have the client generate a valid device mode for the printer and store it on the server for you.

6.8.3. Further Client Driver Insta Procedures

Every further driver may be done by any user, along the lines described above: Browse network, open printers folder on Samba server, right−click printer and choose "Connect...". Once this completes (should be not more than a few seconds, but could also take a minute, depending on network conditions), you should find the new printer in your client workstation local "*Printers and Faxes*" folder.

You can also open your local "Printers and Faxes" folder by using this command on Windows 2000 and Windows XP Professional workstations:

```
rundll32 shell32.dll,SHHelpShortcuts_RunDLL PrintersFolder
```

or this command on Windows NT 4.0 workstations:

```
rundll32 shell32.dll,Control_RunDLL MAIN.CPL @2
```

You can enter the commands either inside a "DOS box" window or in the "Run command..." field from the "Start" menu.

6.8.4. Always make first Client Connection as root or "printer admin"

After you installed the driver on the Samba server (in its [print$] share, you should always make sure that your first client installation goes OK. Make it a habbit for yourself to build that the very first connection from a client as "printer admin". This is to make sure that

- a) a first valid "*Device Mode*" is really initialized (see above for more explanation details), and that
- b) the default print settings of your printer for all further client installations are as you want them

Do this by changing the orientation to landscape, " *Apply*" and change it back. Then set up the other things. (You don't want the default media size set to *Letter*, when you are all using *A4*, right? You may want to set the thing do *duplex* as the default; etc.). −−

To connect as root to a Samba printer, try this command from a Windows 2K/XP DOS box command prompt:

```
runas /netonly /user:root "rundll32 printui.dll,PrintUIEntry /p /t3 /n \\SAMBA−SERVER\printername"
```

You are prompted for root's Samba−password; type it, wait a few seconds and click on "*Printing Defaults*..." and proceed to set the job options as should be used as defaults by all clients. −− Alternatively, instead of root you can name one other member of the ***printer admins*** from the smb.conf setting.

Now all the other users downloading and installing the driver the same way (called "*Point'n'Print*") will have the same defaults set for them. If you miss this step you'll get a lot of helpdesk calls from your users. But maybe you like to talk to people.... ;−)

6.9. Other Gotchas

Your driver is installed. It is ready for " *Point'n'Print*" installation by the clients now. You *may* have tried to download and use it onto your first client machine now. But wait... let's make you acquainted first with a few tips and tricks you may find useful. For example, you didn't manage to "set the defaults" on the printer, as adviced in the preceeding paragraphs? And your users complain about various issues ? (like *"We need to set the paper size for each job from Letter to A4 and it won't store it!"*)

6.9.1. Setting default Print Options for the Client Drivers

The last sentence might by seen with mixed feelings by some users and admins. They have struggled for hours and hours and couldn't arrive at a point were their settings seemed to be saved. It is not their fault. The confusing thing is this: in the multi−tabbed dialog popping up when you right−click the printer name and select " *Properties...*", you can arrive at two identically looking dialogs, pretending that they help you to set printer options, in three different ways. Here is the definite answer to the "Samba Default Driver Setting FAQ":

"I can't set and save default print options for all users on Win2K/XP! Why not?"

How are you doing it? I bet the wrong way.... (It is not very easy to find out, though). There are 3 different ways to bring you to a dialog that *seems* to set everything. All three dialogs *look* the same. Only one of them *does* what you intend. (You need to be Administrator or Print Administrator to do it for all users.)
Here is how I reproduce it in on XP Prof (my interface is German −− my re−translation into English might not be accurate):

A. the first "wrong" way:

- ◊ 1. Open the "*Printers*" folder.
- ◊ 2. Right−click on the printer ("*remoteprinter on cupshost*") and select in context menu "*Printing Preferences..."*
- ◊ 3. Look at this dialog closely and remember how it looks like.

B. the second, another "wrong" way:

- ◊ 1. Open the "*Printers*" folder.
- ◊ 2. Right−click on the printer ("*remoteprinter on cupshost*") and select in context menu "*Properties*"
- ◊ 3. Click on the "*General*" tab
- ◊ 4. Click on the button "*Printing Preferences...*"
- ◊ 5. A new dialog opens. Keep this dialog open and go back to the parent dialog.

C. the third, the "correct" way:

(Should you do it from the beginning, just do steps 1. + 2. from above "B.")

- ◊ 3. Click on the "*Advanced*" tab. (Hmmm... everything "Grayed Out"? You are not logged in as a user with enough privileges then.)
- ◊ 4. Click on the "*Printing Defaults...*" button. ***(*)***
- ◊ 5. On any of the two new tabs, click on the "*Advanced...*" button. ***(*)***
- ◊ 6. A new dialog opens. Compare this one to the other, identical looking one from "B.5" or A.3".

Do you see any difference in the two settings dialogs? I don't either... However, only the last one, which you arrived at with steps "C.1.−6." will safe any settings to become permanent and be the defaults for new users. If you want all clients to get the same defaults, you need to conduct these steps *as administrator* (*printer admin* in smb.conf) *before* a client downloads the driver. (The clients can later set their own *per−user defaults* by following the procedures ***A.*** or ***B.*** above...). (This is new: Windows 2000 and Windows XP are allowing *per−user* default settings and the ones the admin gives them, before they set up their own.)

The "parents" of the identically looking dialogs have a slight difference in their window names: one is called "*Default Print Values for Printer Foo on Server Bar"* (the one you was looking for) and the other is called "*Print Settings for Printer Foo on Server Bar*". The last one is also the one you arrive at when you right−click the printer and select "*Print Settings...*" And exactly this is tje one what you was taught to use back in the days of Windows NT! So it is only natural to try the same way with Win2k or WinXP. You wouldn't dream of thinking that there is now a different "clicking path" to arrive at an identically looking, but functionally different dialog to set defaults for all users!

One final tip. try (on Win2000 and WinXP) to run this command (as a user with the right privileges):

- ◊ ***rundll32 printui.dll,PrintUIEntry /p /t3 /n\\SAMBA−SERVER\printersharename***

to see the tab with the "*Printing Defaults...*" button (the one you need). Run this command

◊ ***rundll32 printui.dll,PrintUIEntry /p /t0 /n\\SAMBA−SERVER\printersharename***

to see the tab with the "*Printing Preferences...*" button (the one which doesn't set system−wide defaults). You can start the commands from inside a DOS box" or from the "*Start −−> Run...*" menu.

6.9.2. Installing large Numbers of Printers

One issue that has arisen during the recent development phase of Samba is the need to support driver downloads for 100's of printers. Using Windows NT APW here is somewhat awkward (to say the least). If you don't want to acquire RSS pains from such the printer installation clicking orgy alone, you need to think about a non−interactive script.

If more than one printer are using the same driver, the ***rpcclient setdriver*** command can be used to set the driver associated with an installed queue. If the driver is uploaded to [print$] once and registred with the printing TDBs, it can be used by multiple print queues. In this case you just need to repeat the setprinter subcommand of rpcclient for every queue (without the need to conduct the *adddriver* again and again). The following is an example of how this could be accomplished. Of course you could mould this into a complete script, automating everything:

```
kde−bitshop:~$> rpcclient SAMBA−CUPS −U root%secret −c 'enumprinters' | grep −i Name

       name:[\\kde−bitshop\ir85wm]
       description:[\\kde−bitshop\ir85wm,,makes pamphlets, folds, saddlestiches]
−−
       name:[\\kde−bitshop\DigiMaster]
       description:[\\kde−bitshop\DigiMaster,,DigiMaster]
−−
       name:[\\kde−bitshop\dm9110]
       description:[\\kde−bitshop\dm9110,,dm9110]
```

The 'enumprinters' command lists all shared printers known to Samba (and tagged as "browseable" in smb.conf!). Note that the description field would show the driver name, in the field where we see here only two commas...

```
kde−bitshop:~$> rpcclient SAMBA−CUPS −U root%secret −c 'enumdrivers'
cmd = enumdrivers

[Windows NT x86]
Printer Driver Info 1:
  Driver Name: [infotec IS 2075 PCL 6]

Printer Driver Info 1:
  Driver Name: [DANKA InfoStream]

Printer Driver Info 1:
  Driver Name: [Heidelberg DigiM 9110 (PS)]

Printer Driver Info 1:
  Driver Name: [myphantasydrivername]
```

The 'enumdrivers' command lists all drivers installed in the [print$] share. These drivers are there, left over from previous installations.

```
kde−bitshop:~$> rpcclient −U root%secret −c 'setdriver dm9110 "Heidelberg DigiM 9110 (PS)"' SaMbA−cUpS
cmd = setdriver dm9110 Heidelberg DigiM 9110 (PS)
Successfully set dm9110 to driver Heidelberg DigiM 9110 (PS).
```

The 'setdriver' command sets one of the previously listed drivers to be associated with the "dm9110" printer. Note how upper− or lower case is irrelevant to the host name.

```
 kde−bitshop:~$> rpcclient samba−cups −U root%secret −c 'enumprinters' | grep descr | grep dm9110
  description:[\\SAMBA−CUPS\dm9110,Heidelberg DigiM 9110 (PS),110ppm HiVolume DANKA Stuttgart]
```

Now the 'enumprinters' command lists a driver name for printer "dm9110".

```
kde−bitshop:~$> rpcclient SaMbA−cUpS −U root%secret −c 'setdriver dm9110 myphantasydrivername'
cmd = setdriver dm9110 myphantasydrivername
Successfully set dm9110 to myphantasydrivername.
```

We try to set the very same printer to a different driver...

```
kde-bitshop:~$> rpcclient samba-cups -U root%secret -c 'enumprinters' | grep desc | grep dm9110
  description:[\\SAMBA-CUPS\dm9110,myphantasydrivername,110ppm HiVolume DANKA Stuttgart]
```

Again, the 'setdriver' command has set the printqueue with the new driver, as is indicated in the "description" field....

```
samba-cups:~#> for `seq 1 5` do; lpadmin -p my_printer-$i -E ; done
samba-cups:~#> rcsamba stop
samba-cups:~#> rcsamba start
```

This command installs 5 printers into CUPS. Note that these are "raw" printers for CUPS, since we don't specify a PPD.... (The stopping and starting of Samba should not be necessary -- but an insistent bug in recent versions of Samba prevented the automatic update of the printer list. The 3.0 release should see this bug finally squashed...)

```
kde-bitshop:~#> rpcclient -U 'root%secret' -c 'enumprinters' samba-cups | grep descr | grep my_p
       description:[\\SAMBA-CUPS\my_printer-1,,my_printer-1]
       description:[\\SAMBA-CUPS\my_printer-2,,my_printer-2]
       description:[\\SAMBA-CUPS\my_printer-3,,my_printer-3]
       description:[\\SAMBA-CUPS\my_printer-4,,my_printer-4]
       description:[\\SAMBA-CUPS\my_printer-5,,my_printer-5]
```

Here we use 'enumprinters' to query for Samba's sight of things. Note that Samba doesn't list a driver for these newly installed CUPS printers.

```
kde-bitshop:~$> for i in `seq 1 5`; do \
                    rpcclient -d 0 -Uroot%secret -c "setdriver my_printer-$i \
                    myphantasydrivername" kde-bitshop;\
                done

Succesfully set my_printer-1 to driver myphantasydrivername.
Succesfully set my_printer-2 to driver myphantasydrivername.
Succesfully set my_printer-3 to driver myphantasydrivername.
Succesfully set my_printer-4 to driver myphantasydrivername.
Succesfully set my_printer-5 to driver myphantasydrivername.
```

Here we use 'setdriver' to associate the 5 new print queues with a previously installed driver (as listed previously by 'enumdrivers'.) Of course, this driver needs to be choosen so that it is fitting the model for the Windows clients to download and use..

```
kde-bitshop:~> rpcclient -U 'root%secret -c 'enumprinters' SAMBA-CUPS | grep descr |grep my_p
       description:[\\SAMBA-CUPS\my_printer-1,myphantasydrivername,my_printer-1]
       description:[\\SAMBA-CUPS\my_printer-2,myphantasydrivername,my_printer-2]
       description:[\\SAMBA-CUPS\my_printer-3,myphantasydrivername,my_printer-3]
       description:[\\SAMBA-CUPS\my_printer-4,myphantasydrivername,my_printer-4]
       description:[\\SAMBA-CUPS\my_printer-5,myphantasydrivername,my_printer-5]
```

Now the 'enumprinters' command shows the drivername in the "description" field. Note, that the printers to CUPS still represent "raw" printqueues -- the driver listed here (the same one for each queue) is only usable for Samba's Windows clients to download, install and format the printjobs, which will be spooled as "raw" by CUPS....

It may be not easy to recognize: but the first call to "*enumprinters*" showed the "dm9110" printer with an empty string where the driver should have been listed (between the 2 commas in the "description" field). After the "*setdriver*" command succeeded, all is well. (The CUPS Printing chapter has more info about the installation of printer drivers with the help of rpccclient).

6.9.3. Adding new Printers with the Windows NT APW

By default, Samba exhibits all printer shares defined in *smb.conf* in the "*Printers...*" folder. Also located in this folder is the Windows NT Add Printer Wizard icon. The APW will be shown only if...

- ...the connected user is able to successfully execute an OpenPrinterEx(\\server) with administrative privileges (i.e. root or "***printer admin***").

  (Try this from a Windows 2K/XP DOS box command prompt:

  runas /netonly /user:root rundll32 printui.dll,PrintUIEntry /p /t0 /n \\SAMBA-SERVER\printersharename

and click on " *Printing Preferences...*")
- ...smb.conf contains the setting "***show add printer wizard = yes***" (the default).

The APW can do various things:

- upload a new driver to the Samba [print$] share;
- associate an uploaded driver with an existing (but still "driverless") print queue;
- exchange the currently used driver for an existing print queue with one that has been uploaded before;
- add an entirely new printer to the Samba host (only in conjunction with a working "***add printer command***" −− a complementing "***delete printer command***" for removing entries from the "*Printers...*" folder may be provided too)

The last one (add a new printer) requires more effort than the previous ones. In order to use the APW to successfully add a printer to a Samba server, the *add printer command* must have a defined value. The program hook must successfully add the printer to the Unix print system (i.e. to */etc/printcap*, */etc/cups/printers.conf* or other appropriate files) and to *smb.conf* if necessary.

When using the APW from a client, if the named printer share does not exist, **smbd** will execute the " ***add printer command***" and reparse to the *smb.conf* to attempt to locate the new printer share. If the share is still not defined, an error of "*Access Denied*" is returned to the client. Note that the "***add printer command***" is executed under the context of the connected user, not necessarily a root account. A "***map to guest = bad user***" may have connected you unwittingly under the wrong privilege −− check it by using the "***smbstatus***" command.

6.9.4. Weird Error Message "Can not connect under a different Name"

Once you are connected with the wrong credentials, there is no chance to correct that other than close all Explorer windows, and maybe reboot.

- The "***net use \\SAMBA−SERVER\sharename /user:root***" gives you an error message: "*Multiple connections to a server or a shared resource by the same user utilizing the several user names are not allowed. Disconnect all previous connections to the server, resp. the shared resource, and try again.*"
- Every attempt to "connect a network drive" to "\\SAMBASERVER\print$" to z: is countered by the pertinacious message. *"This network folder is currently connected under different credentials (username and password). Disconnect first any existing connection to this network share in order to connect again under a different username and password*."

So you close all connections. You try again. You get the same message. You check from the Samba side, using "smbstatus". Yes, there are some more connections. You kill them all. The client still gives you the same error message. You watch the smbd.log file on a very high debug level and try re−connect. Same error message, but not a single line in the log. You start to wonder if there was a connection attempt at all. You run ethereal and tcpdump while you try to connect. Result: not a single byte goes on the wire. Windows still gives the errror message. You close all Explorere Windows and start it again. You try to connect −− and this times it works! Windows seems to cache connection info somewhere and doesn't keep it up to date. (In unlucky cases you might need to reboot to get rid of the error message.)

6.9.5. Be careful when assembling Driver Files

You need to be very careful when you take notes about the files and belonging to a particular driver. Don't confuse the files for driver version "0" (for Win95/98/ME, going into [print$]/WIN/0/), driver version "2" (Kernel Mode driver for WinNT, going into [print$]/W32X86/2/ −− *may* be used on Win2K/XP too), and driver version "3" (non−Kernel Mode driver going into [print$]/W32X86/3/ −− *can not* be used on WinNT). Very often these different driver versions contain files carrying the same name −− but still the files are very different! Also, if you look at them from the Windows Explorer (they reside in "*%WINDOWS%\system32\spool\drivers\W32X86*") you will probably see names in capital letters, while an "enumdrivers" command from Samba would show mixed or lower case letters. So it is easily possible to confuse them. If you install them manually using "rpcclient" and subcommands, you may even succeed without an error message. Only later, when you try install on a client, you will encounter error messages like "This server has no appropriate driver for the printer..."

Here is an example. You are invited to look very closely at the various files, compare their names and their spelling, and discover the differences in the composition of the version−2 and −3 sets (BTW, the version−0 set contained 40 (!) Dependentfiles, so I left it out for space reasons):

```
kde4@kde−bitshop:~> rpcclient −U 'Administrator%xxxx' −c 'enumdrivers 3' 10.160.50.8

Printer Driver Info 3:
        Version: [3]
        Driver Name: [Canon iR8500 PS3]
        Architecture: [Windows NT x86]
        Driver Path: [\\10.160.50.8\print$\W32X86\3\cns3g.dll]
```

```
        Datafile: [\\10.160.50.8\print$\W32X86\3\iR8500sg.xpd]
        Configfile: [\\10.160.50.8\print$\W32X86\3\cns3gui.dll]
        Helpfile: [\\10.160.50.8\print$\W32X86\3\cns3g.hlp]

        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\aucplmNT.dll]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\ucs32p.dll]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\tnl32.dll]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\aussdrv.dll]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\cnspdc.dll]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\aussapi.dat]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\cns3407.dll]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\CnS3G.cnt]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\NBAPI.DLL]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\NBIPC.DLL]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\cpcview.exe]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\cpcdspl.exe]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\cpcedit.dll]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\cpcqm.exe]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\cpcspl.dll]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\cfine32.dll]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\cpcr407.dll]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\Cpcqm407.hlp]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\cpcqm407.cnt]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\3\cns3ggr.dll]

        Monitorname: []
        Defaultdatatype: []

Printer Driver Info 3:
        Version: [2]
        Driver Name: [Canon iR5000-6000 PS3]
        Architecture: [Windows NT x86]
        Driver Path: [\\10.160.50.8\print$\W32X86\2\cns3g.dll]
        Datafile: [\\10.160.50.8\print$\W32X86\2\IR5000sg.xpd]
        Configfile: [\\10.160.50.8\print$\W32X86\2\cns3gui.dll]
        Helpfile: [\\10.160.50.8\print$\W32X86\2\cns3g.hlp]

        Dependentfiles: [\\10.160.50.8\print$\W32X86\2\AUCPLMNT.DLL]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\2\aussdrv.dll]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\2\cnspdc.dll]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\2\aussapi.dat]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\2\cns3407.dll]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\2\CnS3G.cnt]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\2\NBAPI.DLL]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\2\NBIPC.DLL]
        Dependentfiles: [\\10.160.50.8\print$\W32X86\2\cns3gum.dll]

        Monitorname: [CPCA Language Monitor2]
        Defaultdatatype: []
```

If we write the "version 2" files and the "version 3" files into different text files and compare the result, we see this picture:

```
kde4@kde-bitshop:~> sdiff 2-files 3-files

cns3g.dll                     cns3g.dll
iR8500sg.xpd                  iR8500sg.xpd
cns3gui.dll                   cns3gui.dll
cns3g.hlp                     cns3g.hlp
AUCPLMNT.DLL                | aucplmNT.dll
                            > ucs32p.dll
                            > tnl32.dll
aussdrv.dll                   aussdrv.dll
cnspdc.dll                    cnspdc.dll
aussapi.dat                   aussapi.dat
cns3407.dll                   cns3407.dll
CnS3G.cnt                     CnS3G.cnt
NBAPI.DLL                     NBAPI.DLL
NBIPC.DLL                     NBIPC.DLL
cns3gum.dll                 | cpcview.exe
                            > cpcdspl.exe
                            > cpcqm.exe
                            > cpcspl.dll
                            > cfine32.dll
                            > cpcr407.dll
                            > Cpcqm407.hlp
                            > cpcqm407.cnt
                            > cns3ggr.dll
```

Don't get fooled though! Driver files for each version with identical names may be different in their content, as you can see from this size comparison:

```
kde4@kde-bitshop:~>  for i in cns3g.hlp cns3gui.dll cns3g.dll; do                \
                       smbclient //10.160.50.8/print\$ -U 'Administrator%xxxx' \
                          -c "cd W32X86/3; dir $i; cd .. ; cd 2; dir $i";      \
                       done

  CNS3G.HLP               A   122981  Thu May 30 02:31:00 2002
  CNS3G.HLP               A    99948  Thu May 30 02:31:00 2002

  CNS3GUI.DLL             A  1805824  Thu May 30 02:31:00 2002
  CNS3GUI.DLL             A  1785344  Thu May 30 02:31:00 2002

  CNS3G.DLL               A  1145088  Thu May 30 02:31:00 2002
  CNS3G.DLL               A    15872  Thu May 30 02:31:00 2002
```

In my example were even more differences than shown here. Conclusion: you must be very careful to select the correct driver files for each driver version. Don't rely on the names alone. Don't interchange files belonging to different driver versions.

6.9.6. Samba and Printer Ports

Windows NT/2000 print servers associate a port with each printer. These normally take the form of LPT1:, COM1:, FILE:, etc... Samba must also support the concept of ports associated with a printer. By default, only one printer port, named "Samba Printer Port", exists on a system. Samba does not really need such a "port" in order to print; it rather is a requirement of Windows clients. They insist to get told about an available port when they request this info, otherwise theyw and error message at you. So Samba fakes the port info to keep the Windows clients happy..

Note that Samba does not support the concept of "Printer Pooling" internally either. Printer Pooling assigns a logical printer to multiple ports as a form of load balancing or fail over.

If you require that multiple ports be defined for some reason or another ("My users and my Boss should not know that they are working with Samba"), *smb.conf* possesses a *enumports command* which can be used to define an external program that generates a listing of ports on a system.

6.9.7. Avoiding the most common Misconfigurations of the Client Driver

Soooo −− printing works, but there are still problems. Most jobs print well, some don't at all. Some jobs have problems with fonts, which don't look good at all. Some jobs print fast, and some are dead−slow. We can't cover it all −− but we want to encourage you to read the little paragraph about "Avoiding the wrong PostScript Driver Settings" in the CUPS Printing part of this document.

6.10. The Imprints Toolset

The Imprints tool set provides a UNIX equivalent of the Windows NT Add Printer Wizard. For complete information, please refer to the Imprints web site at *http://imprints.sourceforge.net/* as well as the documentation included with the imprints source distribution. This section will only provide a brief introduction to the features of Imprints.

ATTENTION! MAINTAINER REQUIRED

Unfortunately, the Imprints toolset is no longer maintained. As of December, 2000, the project is in need of a new maintainer. The most important skill is decent perl coding and an interest in MS−RPC based printing using Samba. If you wish to volunteer, please coordinate your efforts on the samba−technical mailing list. The toolset is still in usable form −− but only for a series of older printer models, where there are prepared packages to use. Packages for more up to date print devices are needed if Imprints should have a future.

6.10.1. What is Imprints?

Imprints is a collection of tools for supporting these goals:

- Providing a central repository information regarding Windows NT and 95/98 printer driver packages
- Providing the tools necessary for creating the Imprints printer driver packages.

- Providing an installation client which will obtain printer drivers from a central internet (or intranet) Imprints Server repository and install them on remote Samba and Windows NT 4 print servers.

6.11.2. Creating Printer Driver Packages

The process of creating printer driver packages is beyond the scope of this document (refer to Imprints.txt also included with the Samba distribution for more information). In short, an Imprints driver package is a gzipped tarball containing the driver files, related INF files, and a control file needed by the installation client.

6.12.3. The Imprints Server

The Imprints server is really a database server that may be queried via standard HTTP mechanisms. Each printer entry in the database has an associated URL for the actual downloading of the package. Each package is digitally signed via GnuPG which can be used to verify that package downloaded is actually the one referred in the Imprints database. It is *not* recommended that this security check be disabled.

6.13.4. The Installation Client

More information regarding the Imprints installation client is available in the Imprints−Client−HOWTO.ps file included with the imprints source package.

The Imprints installation client comes in two forms.

- a set of command line Perl scripts
- a GTK+ based graphical interface to the command line perl scripts

The installation client (in both forms) provides a means of querying the Imprints database server for a matching list of known printer model names as well as a means to download and install the drivers on remote Samba and Windows NT print servers.

The basic installation process is in four steps and perl code is wrapped around **smbclient** and **rpcclient**

```
foreach (supported architecture for a given driver)
{
    1.  rpcclient: Get the appropriate upload directory
        on the remote server
    2.  smbclient: Upload the driver files
    3.  rpcclient: Issues an AddPrinterDriver() MS−RPC
}

4.  rpcclient: Issue an AddPrinterEx() MS−RPC to actually
    create the printer
```

One of the problems encountered when implementing the Imprints tool set was the name space issues between various supported client architectures. For example, Windows NT includes a driver named "Apple LaserWriter II NTX v51.8" and Windows 95 calls its version of this driver "Apple LaserWriter II NTX"

The problem is how to know what client drivers have been uploaded for a printer. An astute reader will remember that the Windows NT Printer Properties dialog only includes space for one printer driver name. A quick look in the Windows NT 4.0 system registry at

```
HKLM\System\CurrentControlSet\Control\Print\Environment
```

will reveal that Windows NT always uses the NT driver name. This is ok as Windows NT always requires that at least the Windows NT version of the printer driver is present. However, Samba does not have the requirement internally. Therefore, how can you use the NT driver name if is has not already been installed?

The way of sidestepping this limitation is to require that all Imprints printer driver packages include both the Intel Windows NT and 95/98 printer drivers and that NT driver is installed first.

6.11. Add Network Printers at Logon without User Interaction

The following MS Knowledge Base article may be of some help if you need to handle Windows 2000 clients: *How to Add Printers with No User Interaction in Windows 2000.* (*http://support.microsoft.com/default.aspx?scid=kb;en−us;189105*). It also applies to Windows XP Professional clients.

The ideas sketched out below are inspired by this article. It describes a commandline method which can be applied to install network and local printers and their drivers. This is most useful if integrated in Logon Scripts. You can see what options are available by typing in a command prompt ("DOS box") this:

```
rundll32 printui.dll,PrintUIEntry /?
```

A window pops up which shows you all of the commandline switches available. An extensive list of examples is also provided. This is only for Win 2k/XP. (It doesn't work on WinNT. WinNT has probably some other tools in the respective Resource Kit.) Here's a suggestion what a possible client logon script could contain, with a short explanation of what the lines actually do (it works if 2k/XP Windows clients access printers via Samba, but works for Windows−based print servers too):

```
rundll32 printui.dll,PrintUIEntry /dn /n "\\sambacupsserver\infotec2105−IPDS" /q
rundll32 printui.dll,PrintUIEntry /in /n "\\sambacupsserver\infotec2105−PS"
rundll32 printui.dll,PrintUIEntry /y /n "\\sambacupsserver\infotec2105−PS"
```

Here is a list of the used commandline parameters:

```
/dn −− deletes a network printer
/q  −− quiet modus
/n  −− names a printer
/in −− adds a networ printer connection
/y  −− sets printer as default printer
```

I have tested this with a Samba 2.2.7a and a Samba 3.0alpha24 installation and Windows XP Professional clients. Note that this specific command set works with network print queues. (Installing local print queues requires different parameters, but this is of no interest here.)

- Line 1 deletes a possibly existing previous network printer " *infotec2105−IPDS*" (which had used native Windows drivers with LPRng that were removed from the server which was converted to CUPS). The "/q" at the end eliminates "Confirm" or error dialog boxes popping up. They shouldn't be presented to the user logging on....
- Line 2 adds the new printer "*infotec2105−PS*" (which actually is same physical device but is now run by the new CUPS printing system and associated with the CUPS/Adobe PS drivers). The printer and his driver *must* have been added to Samba prior to the user logging in (f.e. by a procedure as discussed earlier in this chapter, or by running *cupsaddsmb*). The driver is now auto−downloaded to the client PC where the user is about to log in.
- Line 3 sets the default printer to this new network printer (there might be several other printers installed with this same method and some may be local as well −− so we deside for a default printer). The default printer selection may of course be different for different users.

Note that the second line only works if the printer "infotec2105−PS" is an already working printqueue on "sambacupsserver", and if the printer drivers have sucessfully been uploaded (via *APW* , *smbclient/rpcclient* or *cupsaddsmb*) into the [print$] driver repository of Samba. Also, some previous Samba versions required a re−start of smbd after the printer install and the driver upload, otherwise the script (or any other client driver download) would fail.

Since there no easy way to test for the existence of an installed network printer from the logon script, the suggestion is: don't bother checking and just allow the deinstallation/reinstallation every time a user logs in −− it's really quick anyway (1 to 2 seconds).

The additional benefits for this are:

- It puts in place any printer default setup changes automatically at every user logon.
- It allows for "roaming" users' login into the domain from different workstations.

Since network printers are installed per user this makes for an easy up−to−date keeping. The extra few seconds at logon time will not really be noticeable. Printers can be centrally added, changed, and deleted at will on the server with no user intervention required on the clients. (You just need to keep the logon scripts up to date.).

6.12. The "*addprinter command*"

The "***addprinter command***" can be configured to be a shell script or program executed by Samba. It is triggered by running the APW from a client against the Samba print server. The APW asks the user to fill in several fields (like printer name,

driver to be used, comment, port monitor, etc.). These parameters are passed on to Samba by the APW. If the addprinter command is designed in a way that it can create a new printer (through writing correct printcap entries on legacy systems, or execute the *lpadmin* command on more modern systems) and create the associated share in smb.conf, then the APW will in effect really create a new printer on Samba and the UNIX print subsystem!

(FIXME: this has to be more elaborated. A working expample script should be provided. It should be tested too. FIXME],

6.13. Migration of "Classical" printing to Samba 3.0

"NT−style" printer driver management has not changed considerably in 3.0 over the 2.2.x releases (apart from many improvements). Here migration should be quite easy, especially if you followed previous advice to stop using deprecated parameters in your setup. For migrations from an existing 2.0.x setup, or if you continued "Win9x−style" printing in your Samba 2.2 installations, it is more of an effort. Please read the appropriate release notes and the HOWTO Collection for 2.2. You can follow several paths. Here are possible scenarios for migration:

- You need to study and apply the new Windows NT printer and driver support. Previously used parameters "*printer driver file*", " *printer driver*" and "*printer driver location*" are no longer supported.
- If you want to take advantage of WinNT printer driver support you also need to migrate theWin9x/ME drivers to the new setup.
- An existing printers.def file (the one specified in the now removed parameter "printer driver file = ...") will work no longer with Samba−3.0. In 3.0, smbd attempts to locate a Win9x/ME driver files for the printer in [print$] and additional settings in the TDB and only there; if it fails it will *not* (as 2.2.x used to do) drop down to using a printers.def (and all associated parameters). The **make_printerdef** tool is removed and there is no backwards compatibility for this.
- You need to install a Windows 9x driver into the [print$] share for a printer on your Samba host. The driver files will be stored in the "WIN40/0" subdirectory of [print$], and some other settings and info go into the printing−related TDBs.
- If you want to migrate an existing printers.def file into the new setup, the current only solution is to use the Windows NT APW to install the NT drivers and the 9x drivers. This can be scripted using **smbclient** and **rpcclient**. See the Imprints installation client at *http://imprints.sourceforge.net/* for an example. See also the discussion of rpcclient usage in the "CUPS Printing" section.

6.14. Publishing Printer Information in Active Directory or LDAP

We will pubish an update to this section shortly.

6.15. Common Errors and Problems

Here are a few typical errors and problems people have encountered. You can avoid them. Read on.

6.15.1. I give my root password but I don't get access

Don't confuse the root password which is valid for the Unix system (and in most cases stored in the form of a one−way hash in a file named "/etc/shadow") with the password used to authenticate against Samba!. Samba doesn't know the UNIX password; for root to access Samba resources via Samba−type access, a Samba account for root must be created first. This is often done with the "smbpasswd" command.

6.15.2. My printjobs get spooled into the spooling directory, but then get lost

Don't use the existing Unix print system directory for Samba too. It may seem convenient and saving space, but it only leads to problems. The two must be separate.

Chapter 7: CUPS Printing Support in Samba 3.0

Abstract

A great deal of uncertainty regarding CUPS and how it works is seen in a large number of postings on the samba mailing lists. Many express frustration when MS Windows printing appears not to work with a CUPS/Samba back−end.

Here is a good place to point out how CUPS can be used and what it does. This document is a draft for the new "Samba HOWTO Collection", to be published with the upcoming Samba−3.0 release. Your suggestions and comments should go to <kpfeifle at danka dot de>. Especially welcome are bits and items for the "Troubleshooting Tips" section at the end of the document.

This is "work in progress". Currently I consider it 90% complete. It consists mainly of copied'n'pasted bits and pieces I have contributed in the last 15 month to various mailinglists, with some minor editing work done so far.

7.1. Features and Benefits

The Common Unix Print System (CUPS) has become very popular. All big Linux distributions now ship it as their default printing system. But to many it is still a very mystical tool. Normally "It Just Works" (TM). People tend to regard it as a sort of "black box", which they don't want to look into, as long as it works OK. But once there is a little problem, they are in trouble to find out where to start debugging it. Also, even the most recent and otherwise excellent printed Samba documentation deals quite negligent with CUPS printing, leaving out important pieces or even writing plain wrong things about it. This demands rectification. But before you dive into this chapter, make sure that you don't forget to refer to the "Classical Printing" chapter also −− it contains a lot of stuff that is relevant for CUPS too.

CUPS sports quite a few unique and powerful features. While their basic functions may be grasped quite easily, they are also new. Because they are different from other, more traditional printing systems, it is best to try and not apply any prior knowledge about printing upon this new system. Rather try to start understand CUPS from the beginning. This documentation will lead you here to a complete understanding of CUPS, if you study all of the material contained. But lets start with the most basic things first. Maybe this is all you need for now. Then you can skip most of the other paragraphs.

7.1.1. Overview

CUPS is more than just a print spooling system. It is a complete printer management system that complies with the new IPP (*Internet Printing Protocol*). IPP is an industry and IETF (*Internet Engineering Task Force*) standard for network printing. Many of its functions can be managed remotely (or locally) via a web browser (giving you a platform−independent access to the CUPS print server). In addition it has the traditional commandline and several more modern GUI interfaces (GUI interfaces developed by 3rd parties, like KDE's overwhelming *KDEPrint*).

CUPS allows creation of "raw" printers (ie: NO print file format translation) as well as "smart" printers (i.e. CUPS does file format conversion as required for the printer). In many ways this gives CUPS similar capabilities to the MS Windows print monitoring system. Of course, if you are a CUPS advocate, you would argue that CUPS is better! In any case, let us now move on to explore how one may configure CUPS for interfacing with MS Windows print clients via Samba.

7.2. Basic Configuration of CUPS support

Printing with CUPS in the most basic *smb.conf* setup in Samba 3.0 (as was true for 2.2.x) only needs two settings: ***printing*** = ***cups*** and ***printcap*** = ***cups*** . CUPS itself doesn't need a printcap file anymore. However, the*cupsd.conf* configuration file knows two related directives: they control if such a file should be automatically created and maintained by CUPS for the convenience of third party applications (example: *Printcap /etc/printcap* and *PrintcapFormat BSD*). These legacy programs often insist to find a "printcap" file containing printernames −− or they will refuse to print. Make sure CUPS is set to generate and maintain a printcap! For details see "*man cupsd.conf*" and other CUPS−related documentation, like the wealth of documents on your CUPS server itself: *http://localhost:631/documentation.html*.

7.2.1. Linking of smbd with libcups.so

Samba has a very special relationship to CUPS. The reason is: Samba can be compiled with CUPS library support. Most recent installations have this support enabled, and per default CUPS linking is compiled into smbd and other Samba

binaries. Of course, you can use CUPS even if Samba is not linked against *libcups.so* −− but there are some differences in required or supported configuration then.

If SAMBA is compiled against libcups, then printcap = cups uses the CUPS API to list printers, submit jobs, query queues, etc. Otherwise it maps to the System V commands with an additional −*oraw* option for printing. On a Linux system, you can use the ***ldd*** utility to find out details (ldd may not be present on other OS platforms, or its function may be embodied by a different command):

```
transmeta:/home/kurt # ldd `which smbd`
       libssl.so.0.9.6 => /usr/lib/libssl.so.0.9.6 (0x4002d000)
       libcrypto.so.0.9.6 => /usr/lib/libcrypto.so.0.9.6 (0x4005a000)
       libcups.so.2 => /usr/lib/libcups.so.2 (0x40123000)
       [....]
```

The line "*libcups.so.2 => /usr/lib/libcups.so.2 (0x40123000)*" shows there is CUPS support compiled into this version of Samba. If this is the case, and printing = cups is set, then *any otherwise manually set print command in smb.conf is ignored*. This is an important point to remember!

> ***Tip:*** Should you require −− for any reason −− to set your own print commands, you can still do this by setting ***printing = sysv***. However, you'll loose all the benefits from the close CUPS/Samba integration. You are on your own then to manually configure the rest of the printing system commands (most important: *print command*; other commands are *lppause command, lpresume command, lpq command, lprm command, queuepause command* and *queue resume command*).

7.2.2. Simple smb.conf Settings for CUPS

To summarize, here is the simplest printing−related setup for ***smb.conf*** to enable basic CUPS support:

```
[global]
        printing = cups
        printcap name = cups
        load printers = yes

[printers]
        comment = All Printers
        path = /var/spool/samba
        public = yes
        guest ok = yes
        writable = no
        printable = yes
        printer admin = root, @ntadmins
```

This is all you need for basic printing setup for CUPS. It will print all Graphic, Text, PDF and PostScript file submitted from Windows clients. However, nost of your Windows users would not know how to send this kind of files to print without opening a GUI application. Windows clients tend to have local printer drivers installed. And the GUI applecation's print buttons start a printer driver... Your users also very rarely send files from the command line. Unlike UNIX clients, they hardly submit graphic, text or PDF formatted files directly to the spooler. They nearly exclusively print from GUI applications, with a "printer driver" hooked in between the applications native format and the print data stream. If the backend printer is not a PostScript device, the print data stream is "binary", sensible only for the target printer. Read on to learn which problem this may cause and how to avoid it.

7.2.3. More complex smb.conf Settings for CUPS

Here is a slightly more complex printing−related setup for ***smb.conf***. It enables general CUPS printing support for all printers, but defines one printer share which is set up differently.

```
[global]
        printing = cups
        printcap name = cups
        load printers = yes

[printers]
        comment = All Printers
        path = /var/spool/samba
        public = yes
        guest ok = yes
        writable = no
        printable = yes
```

```
        printer admin = root, @ntadmins

[special_printer]
        comment = A special printer with his own settings
        path = /var/spool/samba-special
        printing = sysv
        printcap = lpstat
        print command = echo "NEW: `date`: printfile %f" >> /tmp/smbprn.log ;\
                        echo "     `date`: p-%p s-%s f-%f" >> /tmp/smbprn.log ;\
                        echo "     `date`: j-%j J-%J z-%z c-%c" >> /tmp/smbprn.log :\
                        rm %f
        public = no
        guest ok = no
        writeable = no
        printable = yes
        printer admin = kurt
        hosts deny = 0.0.0.0
        hosts allow = turbo_xp, 10.160.50.23, 10.160.51.60
```

This special share is only there for my testing purposes. It doesn't even write the printjob to a file. It just logs the job parameters known to Samba into the "***/tmp/smbprn.log***" file and deletes the jobfile. Moreover, the "***printer admin***" of this share is "kurt" (not the "@ntadmins" group); guest access is not allowed; the share isn't announced in Network Neighbourhood (so you need to know it is there), and it is only allowing access from three hosts. To prevent CUPS kicking in and taking over the print jobs for that share, we need to set "***printing = sysv***" and "***printcap = lpstat***".

7.3. Advanced Configuration

Before we dive into further configuration options and setup details, let's clarify one point: *Network printing needs to be organized and setup properly!*. Often this is not done right. Legacy systems or small LANs in business environments often lack a clear design and good housekeeping.

7.3.1. Central spooling vs. "Peer-to-Peer" printing

Many small office or home networks, as well as badly organized larger environments, allow each client a direct access to available network printers. Generally, this is a bad idea. It often blocks one client's access to the printer when another client's job is printing. It also might freeze the first client's application while it is waiting to get rid of the job. Also, there are frequent complaints about various jobs being printed with their pages mixed with each other. A better concept is the usage of a "print server": it routes all jobs through one central system, which responds immediately, takes jobs from multiple concurrent clients at the same time and in turn transfers them to the printer(s) in the correct order.

7.3.2. CUPS/Samba as a "spooling-only" Print Server -- "raw" printing with Vendor Drivers on Windows Clients

Most traditionally configured Unix print servers acting on behalf of Samba's Windows clients represented a really simple setup. Their only task was to manage the "raw" spooling of all jobs handed to them by Samba. This approach meant that the Windows clients were expected to prepare the printjob file in such a way that it beame fit to be fed to the printing device. Here a native (vendor-supplied) Windows printer driver for the target device needed to be installed on each and every client.

Of course you can setup CUPS, Samba and your Windows clients in the same, traditional and simple way. When CUPS printers are configured for RAW print-gh mode operation it is the responsibility of the Samba client to fully render the print job (file). The file must be sent in a format that is suitable for direct delivery to the printer. Clients need to run the vendor-provided drivers to do this. In this case CUPS will NOT do any print file format conversion work.

7.3.3. Driver Installation Methods on Windows Clients

The printer drivers on the Windows clients may be installed in two functionally different ways:

- manually install the drivers locally on each client, one by one; this yields the old "*LanMan*" style printing; it uses a "*\\sambaserver\printershare*" type of connection.
- deposit and prepare the drivers (for later download) on the print server (Samba); this enables the clients to use "Point'n'Print" to get drivers semi-automatically installed the first time they access the printer; with this method NT/2K/XP clients use the *SPOOLSS/MS-RPC* type printing calls...

The second method is recommended for use over the first.

7.3.4. Explicitly enable "raw" printing for *application/octet−stream*!

If you use the first option (drivers are installed on the client side), there is one setting to take care of: CUPS needs to be told that it should allow "raw" printing of deliberate (binary) file formats. The CUPS files that need to be correctly set for RAW mode printers to work are:

- /etc/cups/mime.types
- /etc/cups/mime.convs

Both contain entries (at the end of the respective files) which must be uncommented to allow *RAW* mode operation. In ***/etc/cups/mime.types*** make sure this line is present:

```
application/octet−stream
```

In ***/etc/cups/mime.convs***, have this line:

```
application/octet−stream   application/vnd.cups−raw   0   −
```

If these two files are not set up correctly for raw Windows client printing, you may encounter the dreaded "*Unable to convert file 0"* in your CUPS error_log file.

Note: editing the "mime.convs" and the "mime.types" file does not ****enforce**** "raw" printing, it only ****allows**** it.

Background: CUPS being a more security−aware printing system than traditional ones does not by default allow to send deliberate (possibly binary) data to printing devices. (This could be easily abused to launch a "Denial of Service" attack on your printer(s), causing at least the loss of a lot of paper and ink... "Unknown" data are tagged by CUPS as ***MIME type*** "*application/octet−stream*" and not allowed to go to the printer. By default, you can only send other (known) MIME types "raw". Sending data "raw" means: CUPS does not try to convert them and passes them to the printer untouched (see next chapter for even more background explanations...)

This is all you need to know to get the CUPS/Samba combo printing "raw" files prepared by Windows clients, which have vendor drivers locally installed. If you are not interested in background info about more advanced CUPS/Samba printing, simply skip the remaining sections of this chapter.

7.3.5. Three familiar Methods for Driver Upload plus a new one

If you want to use the MS−RPC type printing, you must upload the drivers onto the Samba server first ([print$] share). For a discussion on how to deposit printer drivers on the Samba host (so that the Windows clients can download and use them via "Point'n'Print") please also refer to the previous chapter of this HOWTO Collection. There you find a description or reference to three methods of preparing the client drivers on the Samba server:

- the GUI, "Add Printer Wizard" *upload−from−a−Windows−client* method;
- the commandline, "smbclient / rpcclient" *upload−from−a−UNIX−workstation* method;
- the *Imprints* Toolset method.

These 3 methods apply to CUPS all the same. A new and more convenient way to load the Windows drivers into Samba is provided provided if you use CUPS:

- the "*cupsaddsmb*" utility.

cupsaddsmb is discussed in much detail further below. But we will first explore the CUPS filtering system and compare the Windows and UNIX printing architectures.

7.4. Using CUPS/Samba in an advanced Way −− intelligent printing with PostScript Driver Download

Still reading on? Good. Let's go into more detail then. We now know how to set up a "dump" printserver −− a server which is spooling printjobs "raw", leaving the print data untouched.

Possibly you need to setup CUPS in a more smart way. The reasons could be manifold:

- Maybe your boss wants to get monthly statistics: Which printer did how many pages? What was the average data size of a job? What was the average print run per day? What are the typical hourly peaks in printing? Which departments prints how much?
- Maybe you are asked to setup a print quota system: users should not be able to print more jobs, once they have surpassed a given limit per period?

- Maybe your previous network printing setup is a mess and shall be re−organized from a clean beginning?
- Maybe you have experiencing too many 'Blue Screens, originating from poorly debugged printer drivers running in NT "kernel mode"?

These goals can not be achieved by a raw print server. To build a server meeting these requirements, you'll first need to learn about how CUPS works and how you can enable its features.

What follows is the comparison of some fundamental concepts for Windows and Unix printing; then is the time for a description of the CUPS filtering system, how it works and how you can tweak it.

7.4.1. GDI on Windows −− PostScript on Unix

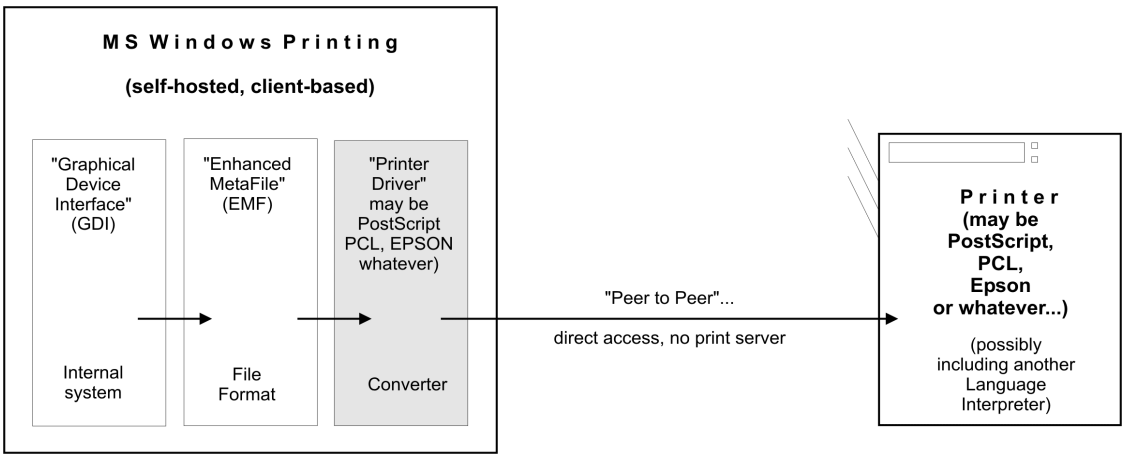
Network printing is one of the most complicated and error−prone day−to−day tasks any user or an administrator may encounter. This is true for all OS platforms. And there are reasons for this.

You can't expect for most file formats to justw them towards printers and they get printed. There needs to be a file format conversion in between. The mess is: there is no common standard for print file formats across all manufacturers and printer types. While ***PostScript*** (trademark held by Adobe), and to an extend ***PCL*** (trademark held by HP), have developed into semi−official "standards", by being the most widely used PDLs (*Page Description Languages*), there are still many manufacturers who "roll there own" (their reasons may be unacceptable license fees for using printer−embedded PostScript interpreters, etc.).

7.4.2. Windows Drivers, GDI and EMF

In Windows OS, the format conversion job is done by the printers drivers. On MS Windows OS platforms all application programmers have at their disposal a built−in API, the GDI (*Graphical Device Interface*), as part and parcel of the OS itself, to base themselves on. This GDI core is used as one common unified ground, for all Windows programs, to draw pictures, fonts and documents *on screen* as well as *on paper* (=print). Therefor printer driver developers can standardize on a well−defined GDI output for their own driver input. Achieving WYSIWYG ("What You See Is What You Get") is relatively easy, because both, the on−screen graphic primitives, as well as the on−paper drawn objects, come from one common source. This source, the GDI, produces often a file format called EMF (*Enhanced MetaFile*). The EMF is processed by the printer driver and converted to the printer−specific file format.

Remark: similar to the GDI foundation in MS Windows, Apple has chosen to put paper and screen output on a common foundation for their (BSD−Unix−based, did you know??) Mac OS X and Darwin Operating Systems.Their *Core Graphic Engine* uses a *PDF* derivate for all display work.



7.4.3. Unix Printfile Conversion and GUI Basics

In Unix and Linux, there is no comparable layer built into the OS kernel(s) or the X (screen display) server. Every application is responsible for itself to create its print output. Fortunately, most use PostScript. That gives at least some common ground. Unfortunately, there are many different levels of quality for this PostScript. And worse: there is a huge difference (and no common root) in the way how the same document is displayed on screen and how it is presented on paper. WYSIWYG is more difficult to achieve. [This goes back to the time decades ago, when the predecessors of *X.org* , designing the UNIX foundations and protocols for Graphical User Interfaces refused to take over responsibility for "paper output" also −− as some had demanded at the time −− and restricted itself to "on−screen only". (There is now, already for a few years, the "Xprint" project going on; it tries to build printing support into the X framework, including a PostScript and a PCL driver, but it is not yet ready for prime time.) You can see this unfavorable inheritance up to the present day by looking into the various "font" directories on your system −− there are separate ones for fonts used for X display and fonts to be

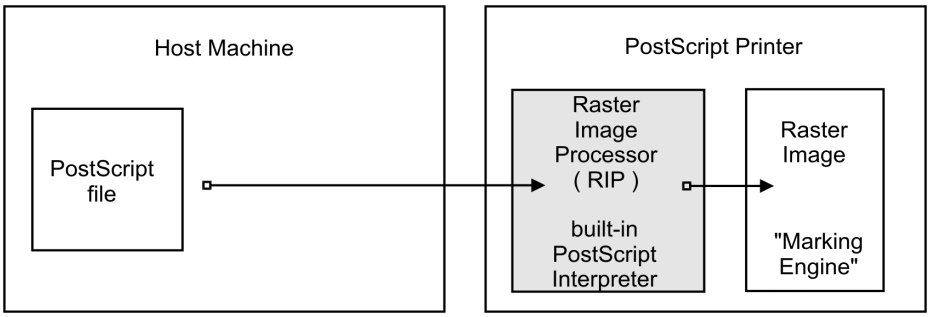
used on paper.]

Background: The PostScript programming language is an "invention" by Adobe Inc. −− but its specifications have been published to the full. Its strength lies in its powerful abilities to describe graphical objects (fonts, shapes, patterns, lines, curves, dots...), their attributes (color, linewidth...) and the way to manipulate (scale, distort, rotate, shift...) them. Because of its open specification, anybody with the skill can start writing his own implementation of a PostScript interpreter and use it to display PostScript files on screen or on paper. Most graphical output devices are based on the concept of "raster images" or "pixels" (one notable exception are pen plotters). Of course, you can look at a PostScript file in its textual form −− and will be reading its PostScript code, the language instructions which need to be interpreted by a rasterizer. Rasterizers produce pixel images, which may be displayed on screen by a viewer program or on paper by a printer...

7.4.4. PostScript and Ghostscript

So, Unix is lacking one common ground for printing on paper and displaying on screen. Despite of this unfavorable legacy for Unix, basic printing is fairly easy ...if you have PostScript printers at your disposal! The reason is: these devices have a PostScript language "interpreter", also called a *Raster Image Processor* (RIP), built−in (which makes them more expensive than other types of printers);w PostScript towards them, and they will spit out your prints. Their RIP is doing all the hard work of converting the PostScript drawing commands into a bitmap picture as you see it on paper, in a resolution as done by your printer. This is no difference to PostScript printing of a file from a Windows origin.

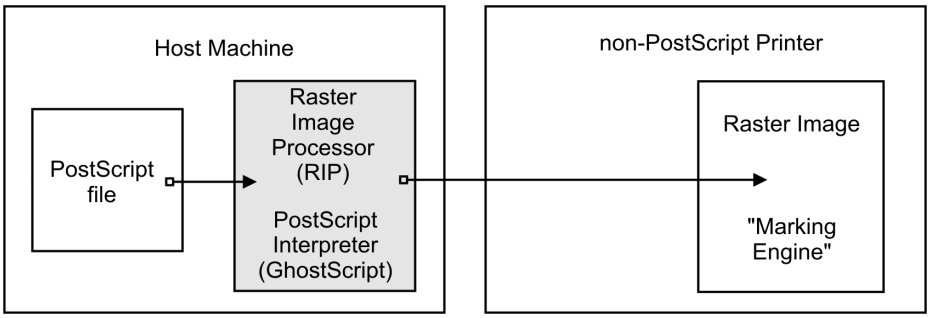
Remark: Traditional Unix programs and printing systems −− while using PostScript −− are largely not PPD−aware. PPDs are "PostScript Printer Description" files. They enable you to specify and control all options a printer supports: duplexing, stapling, punching... Therefore Unix users for a long time couldn't choose many of the supported device and job options, unlike Windows or Apple users. But now there is CUPS.... ;−)



However, there are other types of printers out there. These don't know how to print PostScript. They use their own *Page Description Language* (PDL, often proprietary). To print to them is much more demanding. Since your Unix applications mostly produce PostScript, and since these devices don't understand PostScript, you need to convert the printfiles to a format suitable for your printer on the host, before you can send it away.

7.4.5. Ghostscript −− the Software RIP for non−PostScript Printers

Here is where ***Ghostscript*** kicks in. Ghostscript is the traditional (and quite powerful) PostScript interpreter used on Unix platforms. It is a RIP in software, capable to do a *lot* of file format conversions, for a very broad spectrum of hardware devices as well as software file formats. Ghostscript technology and drivers is what enables PostScript printing to non−PostScript hardware.



Hint: Use the "gs −h" command to check for all built−in "devices" of your Ghostscript version. If you specify f.e. a parameter of "*−sDEVICE=png256*" on your Ghostscript command line, you are asking Ghostscript to convert the input into a PNG file. Naming a "device" on the commandline is the most important single parameter to tell Ghostscript how exactly it should render the input. New Ghostscript versions are released at fairly regular intervals, now by artofcode LLC. They are initially put under the "AFPL" license, but re−released under the GNU GPL as soon as the next AFPL version appears. GNU Ghostscript is probably the version installed on most Samba systems. But it has got some deficiencies. Therefore ESP

Ghostscript was developed as an enhancement over GNU Ghostscript, with lots of bug−fixes, additional devices and improvements. It is jointly maintained by developers from CUPS, Gimp−Print, MandrakeSoft, SuSE, RedHat and Debian. It includes the "cups" device (essential to print to non−PS printers from CUPS).

7.4.6. PostScript Printer Description (PPD) Specification

While PostScript in essence is a *Page Description Language* (PDL) to represent the page layout in a *device independent* way, real world print jobs are always ending up to be output on a hardware with device−specific features. To take care of all the differences in hardware, and to allow for innovations, Adobe has specified a syntax and file format for *PostScript Printer Description* (PPD) files. Every PostScript printer ships with one of these files.

PPDs contain all information about general and special features of the given printer model: Which different resolutions can it handle? Does it have a Duplexing Unit? How many paper trays are there? What media types and sizes does it take? For each item it also names the special command string to be sent to the printer (mostly inside the PostScript file) in order to enable it.

Info from these PPDs is meant to be taken into account by the printer drivers. Therefor, installed as part of the Windows PostScript driver for a given printer is the printer's PPD. Where it makes sense, the PPD features are presented in the drivers' UI dialogs to display to the user as choice of print options. In the end, the user selections are somehow written (in the form of special PostScript, PJL, JCL or vendor−dependent commands) into the PostScript file created by the driver

Warning: A PostScript file that was created to contain device−specific commands for achieving a certain print job output (f.e duplexed, stapled and punched) on a specific target machine, may not print as expected, or may not be printable at all on other models; it also may not be fit for further processing by software (f.e. by a PDF distilling program).

7.4.7. CUPS can use all Windows−formatted Vendor PPDs

CUPS can handle all spec−compliant PPDs as supplied by the manufacturers for their PostScript models. Even if a Unix/Linux−illiterate vendor might not have mentioned our favorable OS in his manuals and brochures −− you can safely trust on this: *if you get hold of the Windows NT version of the PPD, you can use it unchanged in CUPS* and thusly access the full power of your printer just like a Windows NT user could!

Hint: To check the spec compliance of any PPD online, go to *http://www.cups.org/testppd.php* and upload your PPD. You will see the results displayed immediately. (CUPS in all versions after 1.1.19 has a much more strict internal PPD parsing and checking code enabled; in case of printing trouble this online resource should be one of your first pitstops!)

Warning: For real PostScript printers ***don't*** use the "*Foomatic*" or "*cupsomatic*" PPDs from Linuxprinting.org. With these devices the original vendor−provided PPDs are always the first choice!

Tip: If you are looking for an original vendor−provided PPD of a specific device, and you know this odd NT4 box (or any other Windows box) on your LAN has the PostScript driver installed, just use "*smbclient //NT4−box/print\$ −U username*" to access the Windows directory where all printer driver files are stored. First look in the "*W32X86/2*" subdir for the PPD you are searching...

7.4.8. CUPS also uses PPDs for non−PostScript Printers...

CUPS uses specially crafted PPDs to handle non−PostScript printers too. These PPDs are usually not available from the vendors (and "No!", you can't just take the PPD of a Postscript printer with the same model name and hope it works for the non−PostScript version too!). To understand how these PPDs work for non−PS printers we first need to dive deeply into the CUPS filtering and file format conversion architecture. Stay tuned.

7.5. The CUPS Filtering Architecture

The core of the CUPS filtering system is based on *Ghostscript*. In addition to Ghostscript, CUPS uses some other filters of its own. You (or your OS vendor) may have plugged in even more filters. CUPS handles all data file formats under the label of various "MIME types". Every incoming printfile is subjected to an initial "*auto−typing*". The auto−typing determines its given *MIME type*. A given MIME type implies one or more (or none) possible filtering chain regarding the selected target printer. This section discusses how MIME types recognition and conversion rules interact. They are used by CUPS to automatically setup a working filtering chain for any given input data format.

If CUPS rasterizes a PostScript file ***natively*** to a bitmap, this is done in 2 stages:

- the first stage uses a Ghostscript device named "cups" (this is since version 1.1.15) and produces a generic raster format called "CUPS raster".

- the second stage uses a "raster driver" which converts the generic CUPS raster to a device specific raster.

Make sure your Ghostscript version has the "cups" device compiled in (check with "`gs -h | grep cups`"). Otherwise you may encounter the dreaded "*Unable to convert file 0*" in your CUPS error_log file. To have "cups" as a device in your Ghostscript, you either need to *patch GNU Ghostscript* and re−compile, or use "*ESP Ghostscript*" . The superior alternative is ESP Ghostscript: it supports not just CUPS, but also 300 other devices (while GNU Ghostscript supports only ~180). Because of this broad output device support, ESP Ghostscript is the first choice for non−CUPS spoolers too. It is now recommended by Linuxprinting.org for all spoolers.

CUPS printers may be setup to use ***external*** rendering paths. One of the most common ones is provided by the "*Foomatic/cupsomatic*" concept, from Linuxprinting.org. This uses the classical Ghostscript approach, doing everything in one step. It doesn't use the "cups" device, but one of the many others. However, even for Foomatic/cupsomatic usage, best results and broadest printer model support is provided by ESP Ghostscript (more about cupsomatic/Foomatic −− especially the new version called now "foomatic−rip" −− follows below).

7.5.1. MIME types and CUPS Filters

CUPS reads the file `/etc/cups/mime.types` (and all other files carrying a **.types* suffix in the same directory) upon startup. These files contain the MIME type recognition rules which are applied when CUPS runs its auto−typing routines. The rule syntax is explained in the man page for "mime.types" and in the comments section of the *mime.types* file itself. A simple rule reads like this:

```
application/pdf         pdf string(0,%PDF)
```

It means: if a filename has either a "*.pdf*" suffix, or if the magic string "*%PDF*" is right at the beginning of the file itself (offset 0 from the start), then it is a PDF file (*application/pdf*). Another rule is this:

```
application/postscript  ai eps ps string(0,%!) string(0,<04>%!)
```

Its meaning: if the filename has one of the suffixes ".ai", ".eps", ".ps" or if the file itself starts with one of the strings "%!" or "<04>%!", it is a generic PostScript file (*application/postscript*).

Note: there is a very important difference between two similar MIME type in CUPS: one is "*application/postscript*", the other is "*application/vnd.cups−postscript*". While "application/postscript" is meant to be device independent (job options for the file are still outside the PS file content, embedded in commandline or environment variables by CUPS), "application/vnd.cups−postscript" may have the job options inserted into the PostScript data itself (were applicable). The transformation of the generic PostScript (application/postscript) to the device−specific version (application/vnd.cups−postscript) is the responsibility of the CUPS ***pstops*** filter. pstops uses information contained in the PPD to do the transformation.

Hint: Don't confuse any other "mime.types" file your system might be using with the one in the "/etc/cups/" directory.

CUPS can handle ASCII text, HP/GL, PDF, PostScript, DVI and a lot of image formats (GIF, PNG, TIFF, JPEG, Photo−CD, SUN−Raster, PNM, PBM, SGI−RGB and some more) and their associated MIME types with its filters.

7.5.2. MIME type Conversion Rules

CUPS reads the file `/etc/cups/mime.convs` (and all other files named with a **.convs* suffix in the same directory) upon startup. These files contain lines naming an input MIME type, an output MIME type, a format conversion filter which can produce the output from the input type and virtual costs associated with this conversion. One example line reads like this:

```
application/pdf         application/postscript   33   pdftops
```

It means that the "*pdftops*" filter will take "application/pdf" as input and produce "application/postscript" as output, the virtual cost of this operation is 33 CUPS−$. The next filter is more expensive, costing 66 CUPS−$:

```
application/vnd.hp−HPGL application/postscript   66   hpgltops
```

It is the "*hpgltops*", which processes HP−GL plotter files to PostScript.

```
application/octet−stream
```

Here are two more examples:

```
application/x-shell     application/postscript   33    texttops
text/plain              application/postscript   33    texttops
```

The last two examples name the "*texttops*" filter to work on "text/plain" as well as on "application/x-shell". (Hint: this differentiation is needed for the syntax highlighting feature of "texttops").

7.5.3. Filter Requirements

There are many more combinations named in mime.convs. However, you are not limited to use the ones pre-defined there. You can plug in any filter you like into the CUPS framework. It must meet, or must be made to meet some minimal requirements. If you find (or write) a cool conversion filter of some kind, make sure it complies to what CUPS needs, and put in the right lines in *mime.types* and *mime.convs* -- then it will work seamlessly inside CUPS!

Hint: the mentioned "CUPS requirements" for filters are simple. Take filenames or *stdin* as input and write to *stdout*. They should take these 5 or 6 arguments: `printer job user title copies options [filename]`

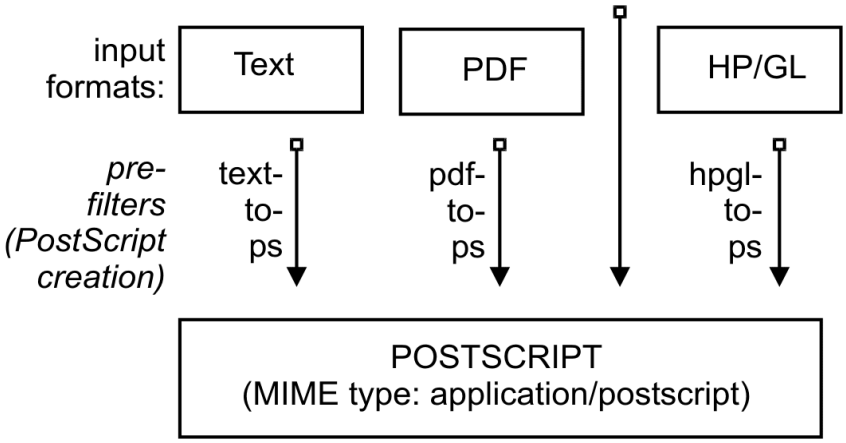
* `printer` - The name of the printer queue (normally this is the name of the filter being run)
* `job` - The numeric job ID for the job being printed
* `user` - The string from the originating-user-name attribute
* `title` - The string from the job-name attribute
* `copies` - The numeric value from the number-copies attribute
* `options` - The job options
* `filename` - (Optionally) The print request file (if missing, filters expected data fed thru *stdin*)

In most cases it is very easy to write a simple wrapper script around existing filters to make them work with CUPS.

7.5.4. Prefilters

As was said, PostScript is the central file format to any Unix based printing system. From PostScript, CUPS generates raster data to feed non-PostScript printers.

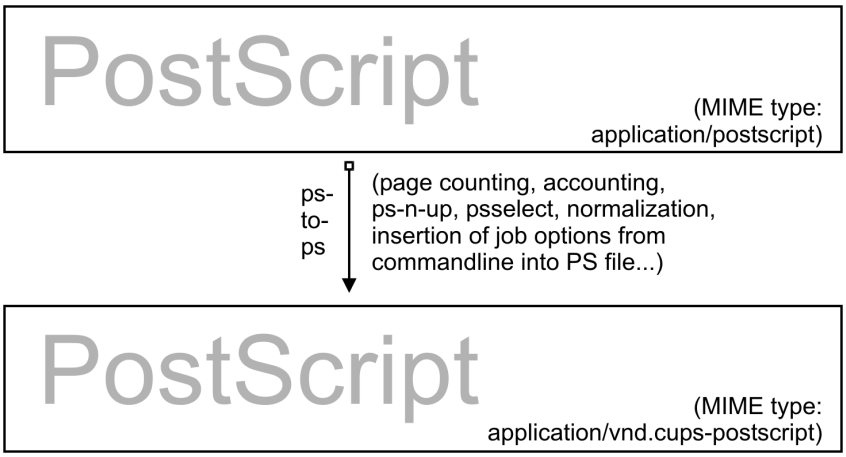
But what is happening if you send one of the supported non-PS formats to print? -- Then CUPS runs "pre-filters" on these input formats to generate PostScript first. There are pre-filters to create PS from ASCII text, PDF, DVI or HP-GL. The outcome of these filters is always of MIME type "application/postscript" (meaning that any device-specific print options are not yet embedded into the PostScript by CUPS, and that the next filter to be called is pstops...). Another pre-filter is running on all supported image formats, the *imagetops* filter. Its outcome is always of MIME type "application/vnd.cups-postscript" (***not*** "application/postscript"), meaning it has the print options already embedded into the file.



-->

7.5.5. pstops

*pstops*is the filter to convert *application/postscript* to *application/vnd.cups-postscript* . It was said above that this filter inserts all device-specific print options (commands to the printer to ask for the duplexing of output, or stapling an punching it, etc.) into the PostScript file.

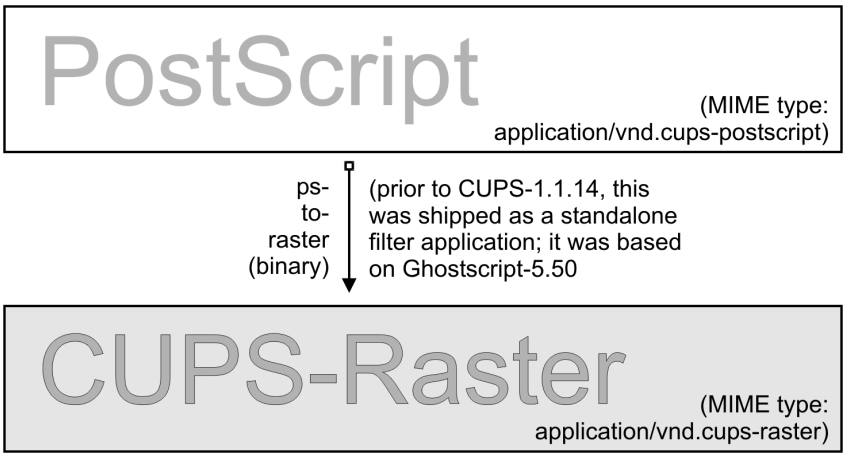


This is not all: other tasks performed by it are

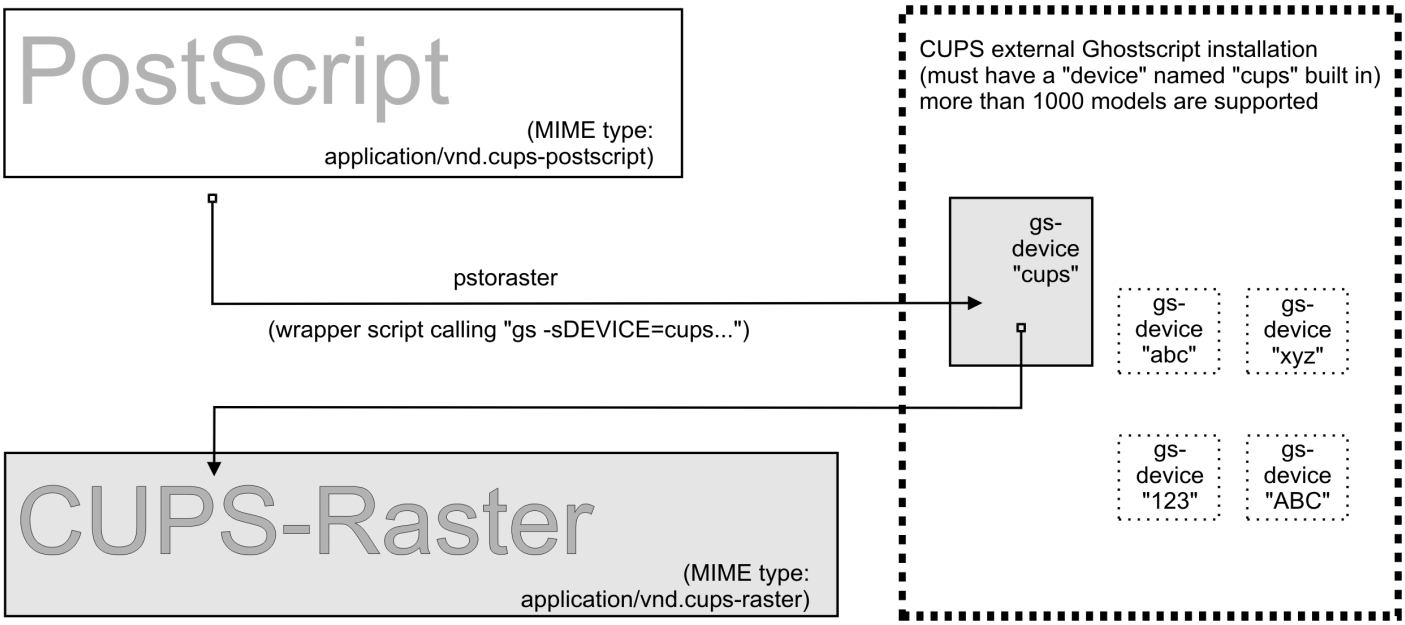
- selecting the range of pages to be printed (if you choose to print only pages "3, 6, 8−11, 16, 19−21", or only the odd numbered ones)
- putting 2 or more logical pages on one sheet of paper (the so−called "number−up" function)
- counting the pages of the job to insert the accounting information into the `/var/log/cups/page_log`

7.5.6. pstoraster

pstoraster is at the core of the CUPS filtering system. It is responsible for the first stage of the rasterization process. Its input is of MIME type application/vnd.cups−postscript; its output is application/vnd.cups−raster. This output format is not yet meant to be printable. Its aim is to serve as a general purpose input format for more specialized *raster drivers*, who are able to generate device−specific printer data.



CUPS raster is a generic raster format with powerful features. It is able to include per−page information, color profiles and more to be used by the following downstream raster drivers. Its MIME type is registered with IANA and its specification is of course completely open. It is designed to make it very easy and inexpensive for manufacturers to develop Linux and Unix raster drivers for their printer models, should they choose to do so. CUPS always takes care for the 1st stage of rasterization so these vendors don't need to care about Ghostscript complications. (In fact, there is currently more than one vendor financing the development of CUPS raster drivers).

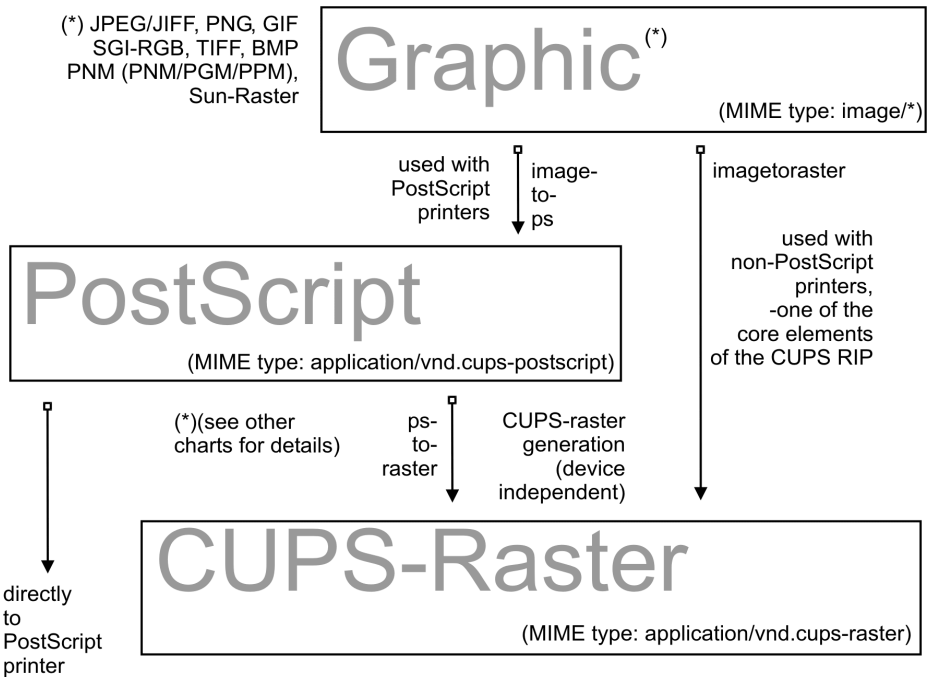


CUPS versions before 1.1.15 were shipping a binary (or source code) standalone filter, named "pstoraster". pstoraster was derived from GNU Ghostscript 5.50, and could be installed besides and in addition to any GNU or AFPL Ghostscript package without conflicting.

From 1.1.15, this has changed. The functions for this has been integrated back into Ghostscript (now based on GNU Ghostscript 7.05). The "pstoraster" filter is now a simple shell script calling "`gs`" with the "`-sDEVICE=cups`" parameter. If your Ghostscript doesn't show a success on asking for "`gs -h |grep cups`", you might not be able to print. Update your Ghostscript then!

7.5.7. imagetops and imagetoraster

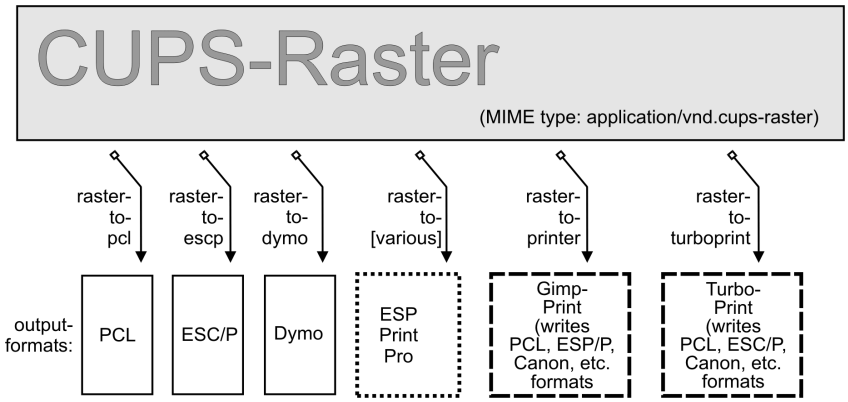
Above, in the section about prefilters we mentioned the prefilter which generates PostScript from image formats. The imagetoraster filter is used to directly go from image to raster, without the intermediate PostScript stage. It is used more often than the above mentioned prefilters. Here is a summarizing flowchart of image file filtering:



7.5.8. rasterto[printerspecific]

CUPS ships with quite some different raster drivers processing CUPS raster. On my system I find in /usr/lib/cups/filter/ these: *rastertoalps*, *rastertobj*, *rastertoepson*, *rastertoescp*, *rastertopcl*, *rastertoturboprint*, *rastertoapdk*, *rastertodymo*, *rastertoescp*, *rastertohp* and *rastertoprinter*. Don't worry if you have less: some of these are installed by commercial add−ons to CUPS (like rastertoturboprint), others (like rastertoprinter) by 3rd party driver development projects (such as

Gimp−Print) wanting to cooperate as closely as possible with CUPS.



7.5.9. CUPS Backends

The last part of any CUPS filtering chain is a "backend". Backends are special programs that send the print−ready file to the final device. There is a separate backend program for any transfer "protocol" of sending printjobs over the network, or for every local interface. Every CUPS printqueue needs to have a CUPS "device−URI" associated with it. The device URI is the way to encode the backend used to send the job to its destination. Network device−URIs are using two slashes in their syntax, local device URIs only one, as you can see from the following list. Keep in mind that local interface names may vary much from my examples, if your OS is not Linux:

usb
: This backend sends printfiles to USB−connected printers. An exampel for the CUPS device−URI to use is: ***usb:/dev/usb/lp0***

serial
: This backend sends printfiles to serially connected printers. An exampel for the CUPS device−URI to use is: ***serial:/dev/ttyS0?baud=11500***

parallel
: This backend sends printfiles to printers connected to the parallel port. An exampel for the CUPS device−URI to use is: ***parallel:/dev/lp0***

scsi
: This backend sends printfiles to printers attached to the SCSI interface. An exampel for the CUPS device−URI to use is: ***scsi:/dev/sr1***

lpd
: This backend sends printfiles to LPR/LPD connected network printers. An exampel for the CUPS device−URI to use is: ***lpd://remote_host_name/remote_queue_name***

: This backend sends printfiles to AppSocket (a.k.a. "HP JetDirect") connected network printers. An exampel for the CUPS device−URI to use is: ***socket://10.11.12.13:9100***

ipp
: This backend sends printfiles to IPP connected network printers (or to other CUPS servers). Exampels for CUPS device−URIs to use are: ***ipp://192.193.194.195/ipp*** (for many HP printers) or ***ipp://remote_cups_server/printers/remote_printer_name***

http
: This backend sends printfiles to HTTP connected printers. (The http:// CUPS backend is only a symlink to the ipp:// backend.) Exampels for the CUPS device−URIs to use are: ***http://192.193.194.195:631/ipp*** (for many HP printers) or ***http://remote_cups_server:631/printers/remote_printer_name***

smb
: This backend sends printfiles to printers shared by a Windows host. An exampel for CUPS device−URIs to use are: ***smb://workgroup/server/printersharename*** or ***smb://server/printersharename*** or ***smb://username:password@workgroup/server/printersharename*** or ***smb://username:password@server/printersharename***. The smb:// backend is a symlink to the Samba utility "*smbspool*" (doesn't ship with CUPS). If the symlink is not present in your CUPS backend directory, have your root user create it: *ln −s `which smbspool` /usr/lib/cups/backend/smb*.

It is easy to write your own backends as Shell or Perl scripts, if you need any modification or extension to the CUPS print system. One reason could be that you want to create "special" printers which send the printjobs as email (through a

"mailto:/" backend), convert them to PDF (through a "pdfgen:/" backend) or dump them to "/dev/null" (In fact I have the system−wide default printer set up to be connected to a "devnull:/" backend: there are just too many people sending jobs without specifying a printer, or scripts and programs which don't name a printer. The system−wided default deletes the job and sends a polite mail back to the $USER asking him to alsways specifiy a correct printername...).

Not all of the mentioned backends may be present on your system or usable (depending on your hardware configuration). One test for all available CUPS backends is provided by the *lpinfo* utility. Used with the −v paralmeter, it lists all available backends:

```
kde−bitshop:~# lpinfo −v
```

7.5.10. cupsomatic/Foomatic −− how do they fit into the Picture?

"cupsomatic" filters may be the most widely used on CUPS installations. You must be clear about the fact that these were not developed by the CUPS people. They are a "Third Party" add−on to CUPS. They utilize the traditional Ghostscript devices to render jobs for CUPS. When troubleshooting, you should know about the difference. Here the whole rendering process is done in one stage, inside Ghostscript, using an appropriate "device" for the target printer. cupsomatic uses PPDs which are generated from the "Foomatic" Printer & Driver Database at Linuxprinting.org.

You can recognize these PPDs from the line calling the "***cupsomatic***" filter:

```
*cupsFilter: "application/vnd.cups−postscript  0  cupsomatic"
```

This line you may find amongst the first 40 or so lines of the PPD file. If you have such a PPD installed, the printer shows up in the CUPS web interface with a "foomatic" namepart for the driver description. cupsomatic is a Perlscript that runs Ghostscript, with all the complicated commandline options auto−constructed from the selected PPD and commandline options give to the printjob.

However, cupsomatic is now deprecated. Its PPDs (especially the first generation of them, still in heavy use out there) are not meeting the Adobe specifications. You might also suffer difficulties when you try to download them with "Point and Print" to Windows clients. A better, and more powerful successor is now in a very stable Beta−version available: it is called " ***foomatic−rip*** ". To use foomatic−rip as a filter with CUPS, you need the new−type PPDs. These have a similar, but different line:

```
*cupsFilter: "application/vnd.cups−postscript  0  foomatic−rip
```

The PPD generating engine at Linuxprinting.org has been revamped. The new PPDs comply to the Adobe spec. On top, they also provide a new way to specify different quality levels (hi−res photo, normal color, grayscale, draft...) with a single click (whereas before you could have required 5 or more different selections (media type, resolution, inktype, dithering algorithm...). There is support for custom−size media built in. There is support to switch print−options from page to page, in the middle of a job. And the best thing is: the new foomatic−rip now works seamlessly with all legacy spoolers too (like LPRng, BSD−LPD, PDQ, PPR etc.), providing for them access to use PPDs for their printing!

7.5.11. The complete Picture

If you want to see an overview over all the filters and how they relate to each other, the complete picture of the puzzle is at the end of this document.

7.5.12. mime.convs

CUPS auto−constructs all possible filtering chain paths for any given MIME type, and every printer installed. But how does it decide in favor or against a specific alternative? (There may often be cases, where there is a choice of two or more possible filtering chains for the same target printer). Simple: you may have noticed the figures in the 3rd column of the mime.convs file. They represent virtual costs assigned to this filter. Every possible filtering chain will sum up to a total "filter cost". CUPS decides for the most "inexpensive" route.

Hint: The setting of " *FilterLimit 1000*" in cupsd.conf will not allow more filters to concurrently run than to consume a total of 1000 virtual filter cost. This is a very efficient way to limit the load of any CUPS server by setting an appropriate "FilterLimit" value. A FilterLimit of 200 allows roughly 1 job at a time, while a FilterLimit of 1000 allows approximately 5 jobs maximum at a time.

7.5.13. "Raw" printing

You can tell CUPS to print (nearly) any file "raw". "Raw" means it will not be filtered. CUPS will send the file to the printer "as is" without bothering if the printer is able to digest it. Users need to take care themselves that they send sensible data formats only. Raw printing can happen on any queue if the "−o raw" option is specified on the command line. You can also set up raw−only queues by simply not associating any PPD with it. This command

```
lpadmin −P rawprinter −v socket://11.12.13.14:9100 −E
```

sets up a queue named "rawprinter", connected via the "socket" protocol (a.k.a. "HP JetDirect") to the device at IP address 11.12.1.3.14, using port 9100. (If you had added a PPD with `−P /path/to/PPD` to this command line, you would have installed a "normal" printqueue.

CUPS will automatically treat each job sent to a queue as a "raw" one, if it can't find a PPD associated with the queue. However, CUPS will only send known MIME types (as defined in its own mime.types file) and refuse others.

7.5.14. "application/octet−stream" printing

Any MIME type with no rule in the */etc/cups/mime.types* file is regarded as unknown or "application/octet−stream" and will not be sent. Because CUPS refuses to print unknown MIME types per default, you will probably have experienced the fact that printjobs originating from Windows clients were not printed. You may have found an error message in your CUPS logs like:

```
Unable to convert file 0 to printable format for job
```

To enable the printing of "application/octet−stream" files, edit these two files:

- /etc/cups/mime.convs
- /etc/cups/mime.types

Both contain entries (at the end of the respective files) which must be uncommented to allow *RAW* mode operation for application/octet−stream. In */etc/cups/mime.types* make sure this line is present:

```
application/octet−stream
```

This line (with no specific auto−typing rule set) makes all files not otherwise auto−typed a member of application/octet−stream. In */etc/cups/mime.convs*, have this line:

```
application/octet−stream   application/vnd.cups−raw   0   −
```

This line tells CUPS to use the *Null Filter* (denoted as "−", doing... nothing at all) on "*application/octet−stream*", and tag the result as "*application/vnd.cups−raw*". This last one is always a green light to the CUPS scheduler to now hand the file over to the "backend" connecting to the printer and sending it over.

Note: editing the "mime.convs" and the "mime.types" file does not *enforce* "raw" printing, it only *allows* it.

Background: CUPS being a more security−aware printing system than traditional ones does not by default allow to send deliberate (possibly binary) data to printing devices. (This could be easily abused to launch a Denial of Service attack on your printer(s), causing at least the loss of a lot of paper and ink...) "Unknown" data are regarded by CUPS as ***MIME type*** "*application/octet−stream*". While you ***can*** send data "raw", the MIME type for these must be one that is known to CUPS and an allowed one. The file `/etc/cups/mime.types` defines the "rules" how CUPS recognizes MIME types. The file `/etc/cups/mime.convs` decides which file conversion filter(s) may be applied to which MIME types.

7.5.15. PostScript Printer Descriptions (PPDs) for non−PS Printers

Originally PPDs were meant to be used for PostScript printers only. Here, they help to send device−specific commands and settings to the RIP which processes the jobfile. CUPS has extended this scope for PPDs to cover non−PostScript printers too. This was not very difficult, because it is a standardized file format. In a way it was logical too: CUPS handles PostScript and uses a PostScript RIP (=Ghostscript) to process the jobfiles. The only difference is: a PostScript printer has the RIP built−in, for other types of printers the Ghostscript RIP runs on the host computer.

PPDs for a non−PS printer have a few lines that are unique to CUPS. The most important one looks similar to this:

```
*cupsFilter: application/vnd.cups−raster  66   rastertoprinter
```

It is the last piece in the CUPS filtering puzzle. This line tells the CUPS daemon to use as a last filter "rastertoprinter". This filter should be served as input an "application/vnd.cups−raster" MIME type file. Therefor CUPS should auto−construct a

filtering chain, which delivers as its last output the specified MIME type. This is then taken as input to the specified "rastertoprinter" filter. After this the last filter has done its work ("rastertoprinter" is a Gimp−Print filter), the file should go to the backend, which sends it to the output device.

CUPS by default ships only a few generic PPDs −− but they are good for several hundred printer models. You may not be able to control different paper trays, or you may get larger margins than your specific model supports):
* `deskjet.ppd`, older HP inkjet printers and compatible
* `deskjet2.ppd`, newer HP inkjet printers and compatible
* `dymo.ppd`, label printers
* `epson9.ppd`, Epson 24pin impact printers and compatible
* `epson24.ppd`, Epson 24pin impact printers and compatible
* `okidata9.ppd`, Okidata 9pin impact printers and compatible
* `okidat24.ppd`, Okidata 24pin impact printers and compatible
* `stcolor.ppd`, older Epson Stylus Color printers
* `stcolor2.ppd`, newer Epson Stylus Color printers
* `stphoto.ppd`, older Epson Stylus Photo printers
* `stphoto2.ppd`, newer Epson Stylus Photo printers
* `laserjet.ppd`, all PCL printers

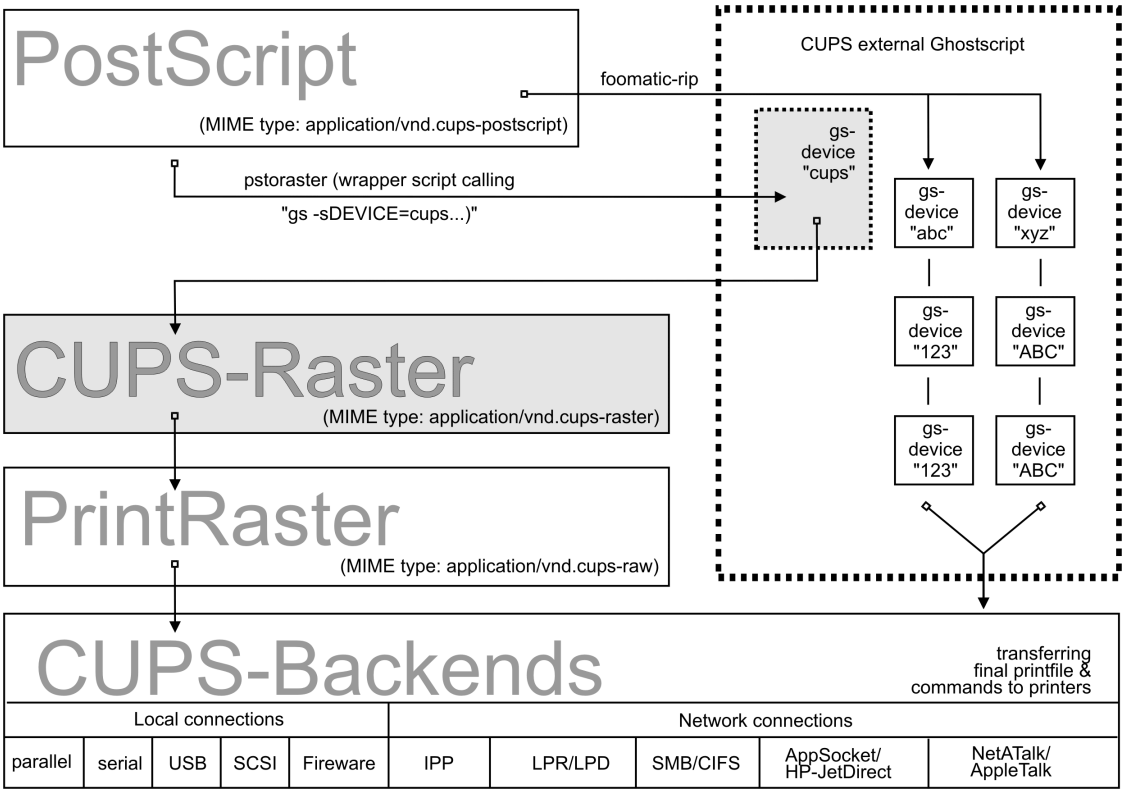
Further below is a discussion of several other driver/PPD−packages suitable fur use with CUPS.

7.5.16. Difference: *"cupsomatic/foomatic−rip"* and *"native CUPS"* printing

Native CUPS rasterization works in two steps.

- First is the "pstoraster" step. It uses the special "cups" device from ESP Ghostscript 7.05.x as its tool
- Second comes the "rasterdriver" step. It uses various device−specific filters; there are several vendors who provide good quality filters for this step, some are Free Software, some are Shareware/Non−Free, some are proprietary.

Often this produces better quality (and has several more advantages) than other methods.



One other method is the "***cupsomatic/foomatic−rip***" way. Note, that cupsomatic is ***not*** made by the CUPS developers. It is an independent contribution to printing development, made by people from Linuxprinting.org (see also http://www.cups.org/cups−help.html). cupsomatic is no longer developed and maintained and is no longer supported. It has now been replaced by "foomatic−rip". foomatic−rip is a complete re−write of the old cupsomatic idea, but very much improved and generalized to other (non−CUPS) spoolers. An upgrade to foomatic−rip is strongly adviced, especially if you are upgrading to a recent version of CUPS too.

Both, the cupsomatic (old) and the "***foomatic−rip***" (new) methods from Linuxprinting.org use the traditional Ghostscript print file processing, doing everything in a single step. It therefore relies on all the other devices built−in into Ghostscript. The quality is as good (or bad) as Ghostscript rendering is in other spoolers. The advantage is that this method supports many printer models not supported (yet) by the more modern CUPS method.

Of course, you can use both methods side by side on one system (and even for one printer, if you set up different queues), and find out which works best for you....

foomatic−rip "kidnaps" the printfile after the "application/vnd.cups−postscript" stage and deviates it through the CUPS−external, systemwide Ghostscript installation: Therefore the printfile bypasses the "pstoraster" filter (and thus also bypasses the CUPS−raster−drivers "rasterto*something* "). After Ghostscript finished its rasterization, foomatic−rip hands the rendered file directly to the CUPS backend... The flowchart above illustrates the difference between native CUPS rendering and the foomatic−rip method.

7.5.17. Examples for filtering Chains

Here are a few examples of commonly occurring filtering chains to illustrate the workings of CUPS.

Assume you want to print a PDF file to a HP JetDirect−connected PostScript printer, but you want to print the pages 3−5, 7, 11−13 only, and you want to print them "2−up" and "duplex":

- your print options (page selection as required, 2−up, duplex) are passed to CUPS on the commandline: "*lp −d printername −o number−up=2 −o page−ranges=3−5,7,11−13 −o sides=two−sided−long−edge /some/PDF.pdf*";
- the (complete, with all pages) PDF file is sent to CUPS and will be autotyped as "*application/pdf*";
- the file therefore first must pass the "*pdftops*" pre−filter, which produces PostScript MIME type "*application/postscript*" (a preview here would still show all pages of the original PDF);
- the file then passes the "*pstops*" filter which applies the commandline options: it selects the pages 2−5, 7 and 11−13, creates an imposed layout "2 pages on 1 sheet" and inserts the correct "duplex" command (as is defined in the printer's PPD) into the new PostScript file; the file now is of PostScript MIME type "*application/vnd.cups−postscript*";
- the file goes to the "*socket*" backend, which transfers the job to the printer.

The resulting filter chain therefore is *pdftops −−> pstops −−> socket*.

Assume you want to print the same file to an USB−connected Epson Stylus Photo printer, installed with the CUPS *stphoto2.ppd*. The first few filtering stages are nearly the same:

- your print options (page selection as required, 2−up, duplex) are passed to CUPS on the commandline (see above);
- the (complete, with all pages) PDF file is sent to CUPS and autotyped as "*application/pdf*";
- the file therefore first must pass the "*pdftops*" pre−filter, which produces PostScript MIME type "*application/postscript*" (a preview here would still show all pages of the original PDF);
- the file then passes the "pstops" filter which applies the commandline options: it selects the pages 2−5, 7 and 11−13, creates an imposed layout "2 pages on 1 sheet" and inserts the correct "duplex" command... (OOoops −− this printer and his PPD don't support duplex printing at all −− this option will be ignored then) into the new PostScript file; the file now is of PostScript MIME type " *application/vnd.cups−postscript*";
- the file then passes the "*pstoraster*" stage and becomes MIME type "***application/cups−raster***";
- finally, the "*rastertoepson*" filter does its work (as is indicated in the printer's PPD by the "*cupsFilter" line), creating the printer−specific raster data and embedding any user−selected print−options into the print data stream;
- the file goes to the "*usb*" backend, which transfers the job to the printer.

The resulting filter chain therefore is *pdftops −−> pstops −−> pstoraster −−> rastertoepson −−> usb*.

7.5.18. Sources of CUPS drivers / PPDs

On the internet you can find now many thousand CUPS−PPD files (with their companion filters), in many national languages, supporting more than 1.000 non−PostScript models.

- ESP PrintPro (http://wwwl.easysw.com/printpro/) (commercial, non−Free) is packaged with more than 3.000 PPDs, ready for successful usage "out of the box" on Linux, Mac OS X, IBM−AIX, HP−UX, Sun−Solaris, SGI−IRIX, Compaq Tru64, Digital Unix and some more commercial Unices. ESP Print Pro is written by the CUPS developers themselves and its sales help finance the further development of CUPS, as they feed their creators... It should be the first choice of anyone in need of supported drivers and CUPS setup in professional or enterprise environments.
- the Gimp−Print−Project (http://gimp−print.sourceforge.net/) (LPGL/GPL, Free Software) provides around 140 PPDs (supporting nearly 400 printers, many driven to photo quality output), to be used alongside the Gimp−Print raster drivers for CUPS;

- TurboPrint (http://www.turboprint.com/) (Shareware, non−Free) supports roughly the same amount of printers as Gimp−Print in excellent quality (with some advantages for Canon printers);
- OMNI (http://www−124.ibm.com/developerworks/oss/linux/projects/omni/) (LPGL, Free) is a package made by IBM, now containing support for more than 400 printers, stemming from the inheritance of IBM OS/2 KnowHow, ported over to Linux (CUPS support is in a Beta−stage at present);
- HPIJS (http://hpinkjet.sourceforge.net/) (BSD−style licenses, Free) supports around 150 of HP's own printers and is also providing excellent print quality now (currently only via the Foomatic path);
- Foomatic/foomatic−rip (http://www.linuxprinting.org/) (LPGL, Free) from Linuxprinting.org are providing PPDs for practically every Ghostscript filter known to the world (including Omni, Gimp−Print and HPIJS).

NOTE: the cupsomatic/foomatic−rip trick from Linuxprinting.org is working different from the other drivers. This is explained elsewhere in this document.

7.5.19. Printing with Interface Scripts

CUPS also supports the usage of "interface scripts" as known from System V AT&T printing systems. These are often used for PCL printers, from applications that generate PCL print jobs. Interface scripts are specific to printer models. They have a similar role as PPDs for PostScript printers. Interface scripts may inject the Escape sequences as required into the print data stream, if the user has chosen to select a certain paper tray, or print landscape, or use A3 paper, etc. Interfaces scripts are practically unknown in the Linux realm. On HP−UX platforms they are more often used. You can use any working interface script on CUPS too. Just install the printer with the "`−i`" option:

```
lpadmin −p pclprinter −v socket://11.12.13.14:9100 −i /path/to/interface−script
```

Interface scripts might be the "unknown animal" to many. However, with CUPS they provide the most easy way to plug in your own custom−written filtering script or program into one specific print queue. (Some info about the traditional usage of interface scripts is to be found at http://playground.sun.com/printing/documentation/interface.html.

7.6. Network printing (purely Windows)

Network printing covers a lot of ground. To understand what exactly goes on with Samba when it is printing on behalf of its Windows clients, let's first look at a "purely Windows" setup: Windows clients with a Windows NT print server.

7.6.1. From Windows Clients to an NT Print Server

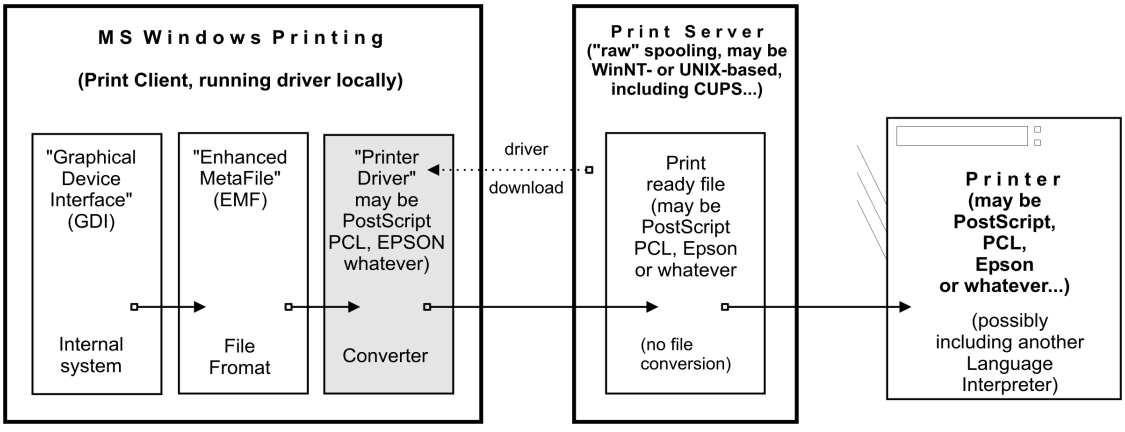
Windows clients printing to an NT−based print server have two options. They may

- execute the driver locally and render the GDI output (EMF) into the printer specific format on their own, or
- send the GDI output (EMF) to the server, where the driver is executed to render the printer specific output.

Both print paths are shown in the flowcharts below.

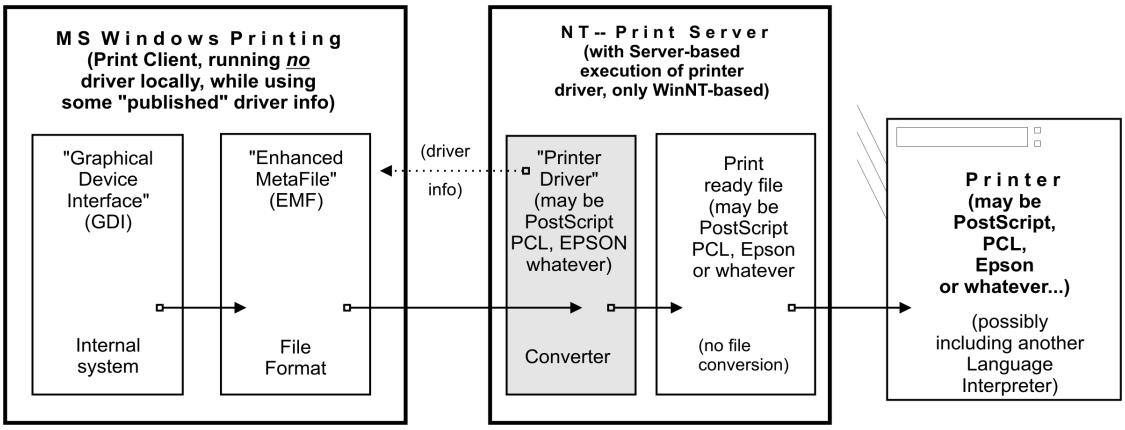
7.6.2. Driver Execution on the Client

In the first case the print server must spool the file as "raw", meaning it shouldn't touch the jobfile and try to convert it in any way. This is what traditional Unix−based print server can do too −− and at a better performance and more reliably than NT print server. This is what most Samba administrators probably are familiar with. One advantage of this setup is that this "spooling−only" print server may be used even if no driver(s) for Unix are available −− it is sufficient to have the Windows client drivers available and installed on the clients.



7.6.3. Driver Execution on the Server

The other path executes the printer driver on the server. The clients transfers print files in EMF format to the server. The server uses the PostScript, PCL, ESC/P or other driver to convert the EMF file into the printer−specific language. It is not possible for Unix to do the same. Currently there is no program or method to convert a Windows client's GDI output on a Unix server into something a printer could understand.



However, there is something similar possible with CUPS. Read on...

7.7. Network Printing (Windows clients −− UNIX/Samba Print Servers)

Since UNIX print servers can *not* execute the Win32 program code on their platform, the picture is somewhat different. However, this doesn't deprecate you in your options much. In the contrary, you may have a way here to implement printing features which are not possible otherwise.

7.7.1. From Windows Clients to a CUPS/Samba Print Server

Here is a simple recipe how you can take advantage of CUPS powerful features for the benefit of your Windows network printing clients.

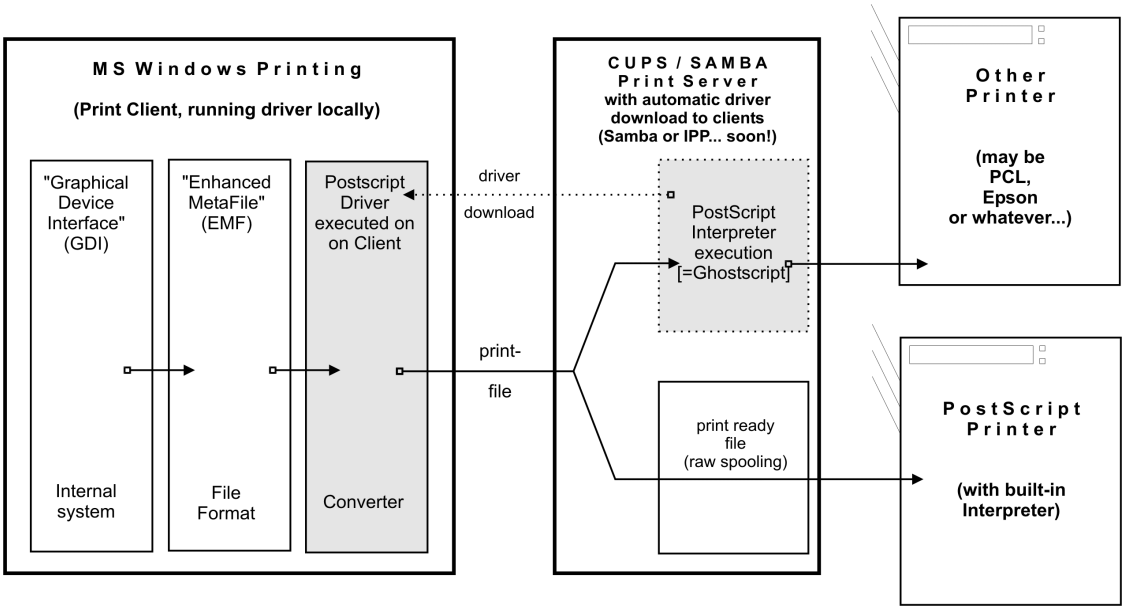
- Let the Windows clients send PostScript to the CUPS server.
- Let the CUPS server render the PostScript into device specific raster format.

This requires the clients to use a PostScript driver (even if the printer is a non−PostScript model. It also requires that you have a "driver" on the CUPS server.

Firstly, to enable CUPS based printing through Samba the following options should be set in your smb.conf file [globals] section:

- `printing = CUPS`
- `printcap = CUPS`

When these parameters are specified, all manually set print directives (like "`print command =...`", or "`lppause command =...`") in *smb.conf* (as well as in samba itself) will be ignored. Instead, Samba will directly interface with CUPS through it's application program interface (API) − as long as Samba has been compiled with CUPS library (libcups) support. If Samba has NOT been compiled with CUPS support, and if no other print commands are set up, then printing will use the *System V* AT&T command set, with the *−oraw* option automatically passing gh. (If you want your own defined print commands to work with a Samba that has CUPS support compiled in, simply use "***printing = sysv*** "...)



7.7.2. Samba receiving Jobfiles and passing them to CUPS

Samba ***must*** use its own spool directory. (It is set by a line similar to "*path = /var/spool/samba*", in the [printers] or [printername] section of smb.conf). Samba receives the job in its own spool space and passes it into the spool directory of CUPS (The CUPS spooling directory is set by the "*RequestRoot*" directive, in a line that defaults to " *RequestRoot /var/spool/cups*"). CUPS checks the access rights of its spool dir and resets it to healthy values with every re−start. We have seen quite some people who had used a common spooling space for Samba and CUPS, and were struggling for weeks with this "problem"..

A Windows user authenticates only to Samba (by whatever means is configured). If Samba runs on the same host as CUPS, you only need to allow "localhost" to print. If they run on different machines, you need to make sure the Samba host gets access to printing on CUPS.

7.8. Network PostScript RIP: CUPS Filters on Server −− clients use PostScript Driver with CUPS−PPDs

PPDs can control all print device options. They are usually provided by the manufacturer −− if you own a PostScript printer, that is. PPD files (PostScript Printer Descriptions) are always a component of PostScript printer drivers on MS Windows or Apple Mac OS systems. They are ASCII files containing user−selectable print options, mapped to appropriate PostScript, PCL or PJL commands for the target printer. Printer driver GUI dialogs translate these options "on−the−fly" into buttons and drop−down lists for the user to select.

CUPS can load, without any conversions, the PPD file from any Windows (NT is recommended) PostScript driver and handle the options. There is a web browser interface to the print options (select http://localhost:631/printers/ and click on one "*Configure Printer*" button to see it), a commandline interface (see **man lpoptions** or see if you have **lphelp** on your system) plus some different GUI frontends on Linux/UNIX, which can present PPD options to users. PPD options are normally meant to become evaluated by the PostScript RIP on the real PostScript printer.

7.8.1. PPDs for non−PS Printers on UNIX

CUPS doesn't limit itself to "real" PostScript printers in its usage of PPDs. The CUPS developers have extended the scope of the PPD concept, to also describe available device and driver options for non−PostScript printers gh CUPS−PPDs.

This is logical, as CUPS includes a fully featured PostScript interpreter (RIP). This RIP is based on Ghostscript. It can process all received PostScript (and additionally many other file formats) from clients. All CUPS−PPDs geared to non−PostScript printers contain an additional line, starting with the keyword **cupsFilter* . This line tells the CUPS print system which printer−specific filter to use for the interpretation of the supplied PostScript. Thus CUPS lets all its printers appear as PostScript devices to its clients, because it can act as a PostScript RIP for those printers, processing the received PostScript code into a proper raster print format.

7.8.2. PPDs for non−PS Printers on Windows

CUPS−PPDs can also be used on Windows−Clients, on top of a "core" PostScript driver (now recommended is the "CUPS PostScript Driver for WindowsNT/2K/XP" −− you can also use the Adobe one, with limitations). This feature enables CUPS to do a few tricks no other spooler can do:

- act as a networked PostScript RIP (Raster Image Processor), handling printfiles from all client platforms in a uniform way;
- act as a central accounting and billing server, since all files are passed through the **pstops** filter and are therefore logged in the CUPS *page_log* file. −− *NOTE:* this can not happen with "raw" print jobs, which always remain unfiltered per definition;
- enable clients to consolidate on a single PostScript driver, even for many different target printers.

Using CUPS PPDs on Windows clients enables these to control all print job settings just as a UNIX client can do too.

7.9. Windows Terminal Servers (WTS) as CUPS Clients

This setup may be of special interest to people experiencing major problems in WTS environments. WTS need often a multitude of non−PostScript drivers installed to run their clients' variety of different printer models. This often imposes the price of much increased instability.

7.9.1. Printer Drivers running in "Kernel Mode" cause many Problems

The reason is that in Win NT printer drivers run in "Kernel Mode". This introduces a high risk for the stability of the system, if the driver is not really stable and well−tested. And there are a lot of bad drivers out there! Especially notorious is the example of the PCL printer driver that had an additional sound module running, to notify users via soundcard of their finished jobs. Do I need to say that this one was also reliably causing "Blues Screens of Death" on a regular basis?

PostScript drivers generally are very well tested. They are not known to cause any problems, even though they run in Kernel Mode too. (This might be because there have so far only been 2 different PostScript drivers −− the ones from Adobe and the one from Microsoft. Both are very well tested and are as stable as you ever can imagine on Windows. The CUPS driver is derived from the Microsoft one.

7.9.2. Workarounds impose heavy Limitations

In many cases, in an attempt to work around this problem, site administrators have resorted to restrict the allowed drivers installed on their WTS to one generic PCL− and one PostScript driver. This however restricts the clients in the amount of printer options available for them −− often they can't get out more then simplex prints from one standard paper tray, while their devices could do much better, if driven by a different driver!)

7.9.3. CUPS −− a "Magical Stone"?

Using a PostScript driver, enabled with a CUPS−PPD, seems to be a very elegant way to overcome all these shortcomings. There are −− depending on the version of Windows OS you use −− up to 3 different PostScript drivers available: Adobe, Microsoft and CUPS PostScript drivers. None of them is known to cause major stability problems on WTS (even if used with many different PPDs). The clients will be able to (again) chose paper trays, duplex printing and other settings. However, there is a certain price for this too: a CUPS server acting as a PostScript RIP for its clients requires more CPU and RAM than when just acting as a "raw spooling" device. Plus, this setup is not yet widely tested, although the first feedbacks look very promising...

7.9.4. PostScript Drivers with no major problems −− even in Kernel Mode

More recent printer drivers on W2K and XP don't run in Kernel mode (unlike Win NT) any more. However, both operating systems can still use the NT drivers, running ini Kernel mode. (You can roughly tell the drivers in subdirectory "2" of "W32X86" are "old" ones). As was said before, the Adobe as well as the Microsoft PostScript drivers are not known to cause any stability problems. The CUPS driver is derived from the Microsoft one. There is a simple reason for this: The MS DDK (Device Development Kit) for Win NT (which used to be available at no cost to licensees of Visual Studio) includes the source code of the Microsoft driver, and licensees of Visual Studio are allowed to use and modify it for their own driver development efforts. This is what the CUPS people have done. The license doesn't allow them to publish the whole of the source code. However, they have released the "diff" under the GPL, and if you are owner of an "MS DDK for Win NT", you can check the driver yourself.

7.10. Setting up CUPS for driver Download

We have said before: all previously known methods to prepare client printer drivers on the Samba server for download and Point'n'Print convenience of Windows workstations are working with CUPS too. These methods were described in the previous chapter. (In reality, this is a pure Samba business, and only a matters the Samba/Win client relationship.)

7.10.1. *cupsaddsmb* −− the unknown Utility

The **cupsaddsmb** utility (shipped with all current CUPS versions) is an alternative method to transfer printer drivers into the Samba [print$] share. Remember, this share is where clients expect drivers deposited and setup for download and installation. It makes the sharing of any (or all) installed CUPS printers very easy. cupsaddsmb can use the Adobe PostScript driver as well as the newly developed "*CUPS PostScript Driver for WinNT/2K/XP* ". Note, that cupsaddsmb does *not* work with arbitrary vendor printer drivers, but only with the *exact* driver files that are named in its man page.

The CUPS printer driver is available from the CUPS download site. Its package name is "*cups−samba−[version].tar.gz* ". It is prefered over the Adobe drivers since it has a number of advantages:

- it supports a much more accurate page accounting;
- it supports banner pages, and page labels on all printers;
- it supports the setting of a number of job IPP attributes (such as job−priority, page−label and job−billing)

However, currently only Windows NT, 2000, and XP are supported by the CUPS drivers. You will have to get the respective part of Adobe driver too if you need to support Windows 95, 98, and ME clients.

7.10.2. Prepare your smb.conf for cupsaddsmb

Prior to running cupsaddsmb, you need the following settings in smb.conf:

```
[global]
        load printers = yes
        printing = cups
        printcap name = cups

[printers]
        comment = All Printers
        path = /var/spool/samba
        browseable = no
        public = yes
        guest ok = yes          # setting depends on your requirements
        writable = no
        printable = yes
        printer admin = root

[print$]
        comment = Printer Drivers
        path = /etc/samba/drivers
        browseable = yes
        guest ok = no
        read only = yes
        write list = root
```

7.10.3. CUPS Package of "PostScript Driver for WinNT/2k/XP"

CUPS users may get the exactly same packages from *http://www.cups.org/software.html*. It is a separate package from the CUPS base software files, tagged as "*CUPS 1.1.x Windows NT/2k/XP Printer Driver for SAMBA (tar.gz, 192k)*". The filename to download is "*cups−samba−1.1.x.tar.gz*". Upon untar−/unzip−ing, it will reveal these files:

```
kurt@kde−bitshop:~> tar xvzf cups−samba−1.1.19.tar.gz

   cups−samba.install
   cups−samba.license
   cups−samba.readme
   cups−samba.remove
   cups−samba.ss
```

These have been packaged with the ESP meta packager software "EPM". The *.*install* and *.*remove* files are simple shell scripts, which untars the *.*ss* (the *.*ss* is nothing else but a tar−archive, which can be untar−ed by "tar" too). Then it puts the content into */usr/share/cups/drivers/*. This content includes 3 files:

```
kurt@kde−bitshop:~> tar tv cups−samba.ss

   cupsdrvr.dll
   cupsui.dll
   cups.hlp
```

The *cups−samba.install* shell scripts is easy to handle:

```
kurt@kde−bitshop:/tmp> ./cups−samba.install

  [....]
  Installing software...
  Updating file permissions...
  Running post−install commands...
  Installation is complete.
```

The script should automatically put the driver files into the */usr/share/cups/drivers/* directory.

ATTENTION: due to a bug, one recent CUPS release puts the *cups.hlp* driver file into */usr/share/drivers/* instead of */usr/share/cups/drivers/*. To work around this, copy/move the file (after running the " *./cups−samba.install*" script) manually to the right place

```
   cp /usr/share/drivers/cups.hlp /usr/share/cups/drivers/
```

This new CUPS PostScript driver is currently binary−only, but free of charge. No complete source code is provided (yet). The reason is this: it has been developed with the help of the *Microsoft Driver Developer Kit* (DDK) and compiled with Microsoft Visual Studio 6. Driver developers are not allowed to distribute the whole of the source code as Free Software. However, CUPS developers released the "diff" in source code under the GPL, so anybody with a license of Visual Studio and a DDK will be able to compile for him/herself.

7.10.4. Recognize the different Driver Files

The CUPS drivers don't support the "older" Windows 95/98/ME, but only the Windows NT/2000/XP client:

```
[Windows NT, 2000, and XP are supported by:]
        cups.hlp
        cupsdrvr.dll
        cupsui.dll
```

Adobe drivers are available for the older Windows 95/98/ME as well as the Windows NT/2000/XP clients. The set of files is different for the different platforms.

```
[Windows 95, 98, and Me are supported by:]
        ADFONTS.MFM
        ADOBEPS4.DRV
        ADOBEPS4.HLP
        DEFPRTR2.PPD
        ICONLIB.DLL
        PSMON.DLL

[Windows NT, 2000, and XP are supported by:]
        ADOBEPS5.DLL
        ADOBEPSU.DLL
        ADOBEPSU.HLP
```

Note: if both, the Adobe driver files and the CUPS driver files for the support of WinNT/2k/XP are present in , the Adobe ones will be ignored and the CUPS ones will be used. If you prefer −− for whatever reason −− to use Adobe−only drivers, move away the 3 CUPS driver files. The Win95/98/ME clients use the Adobe drivers in any case.

7.10.5. Acquiring the Adobe Driver Files

Acquiring the Adobe driver files seems to be unexpectedly difficult for many users. They are not available on the Adobe website as single files and the self−extracting and/or self−installing Windows−exe is not easy to locate either. Probably you need to use the included native installer and run the installation process on one client once. This will install the drivers (and one Generic PostScript printer) locally on the client. When they are installed, share the Generic PostScript printer. After this, the client's [print$] share holds the Adobe files, from where you can get them with smbclient from the CUPS host. A more detailed description about this is in the next (the CUPS printing) chapter...

7.10.6. ESP Print Pro Package of "PostScript Driver for WinNT/2k/XP"

Users of the ESP Print Pro software are able to install their "Samba Drivers" package for this purpose with no problem. Retrieve the driver files from the normal download area of the ESP Print Pro software at *http://www.easysw.com/software.html*. You need to locate the link labelled "SAMBA" amongst the "*Download Printer Drivers for ESP Print Pro 4.x*" area and download the package. Once installed, you can prepare any driver by simply highlighting the printer in the Printer Manager GUI and select "*Export Driver...*" from the menu. (Of course you need to have prepared Samba beforehand too to handle the driver files; i.e. mainly setup the [print$] share, etc.). The ESP Print Pro package includes the CUPS driver files as well as a (licensed) set of Adobe drivers for the Windows 95/98/ME client family.

7.10.7. Caveats to be considered...

Once you have run the install script (and possibly manually moved the "*cups.hlp*" file to " */usr/share/cups/drivers/*"), the driver is ready to be put into Samba's [print$] share (which often maps to "*/etc/samba/drivers/*" and contains a subdir tree with *WIN40* and *W32X86* branches): You do this by running "cupsaddsmb" (see also "*man cupsaddsmb*" for CUPS since release 1.1.16). [***Hint***: You may need to put root into the smbpasswd file by running "*smbpasswd*"; this is especially important if you should run this whole procedure for the first time, and are not working in an environment where everything is configured for "*Single Sign On*" to a Windows Domain Controller.]

Once the driver files are in the [print$] share and are initialized, they are ready to be downloaded and installed by the Win NT/2k/XP clients.

NOTE 1: Win 9x/ME clients won't work with the CUPS PostScript driver. For these you'd still need to use the ADOBE*.* drivers as previously.

NOTE 2: It is not harming if you've still the ADOBE*.* driver files from previous installations in the "*/usr/share/cups/drivers/*" directory. The new *cupsaddsmb* (from 1.1.16) will automatically prefer "its own" drivers if it finds both.

NOTE 3: Should your Win clients have had the old ADOBE*.* files for the Adobe PostScript driver installed, the download and installation of the new CUPS PostScript driver for Windows NT/2k/XP will fail at first. You need to wipe the old driver from the clients first. It is not enough to "delete" the printer as the driver files will still be kept by the clients and re−used if you try to re−install the printer. To really get rid of the Adobe driver files on the clients, open the "Printers" folder (possibly via "*Start −−> Settings −−> Control Panel −−> Printers*"), right−click onto the folder background and select "*Server Properties*". A new dialog opens; select the "*Drivers*" tab; on the list select the driver you want to delete and click on the "*Delete*" button. (This will only work if there is not a single printer left which uses that particular driver −− you need to "delete" all printers using this driver in the "Printers" folder first.) You need Administrator privileges to do this.

NOTE 4: Once you have successfully downloaded the CUPS PostScript driver to a client, you can easily switch all printers to this one by proceeding as described elsewhere in the "Samba HOWTO Collection": either change a driver for an existing printer by running the "Printer Properties" dialog, or use "rpcclient" with the "setdriver" sub−command.

7.10.8. Which Benefits from "CUPS PostScript Driver for Windows NT/2k/XP" as compared to Adobe Driver?

You are interested in a comparison between the CUPS and the Adobe PostScript drivers? For our purposes these are the most important items which weigh in favor of the CUPS ones:

- no hassle with the Adobe EULA
- no hassle with the question "*Where do I get the ADOBE*.* driver files from?*"
- the Adobe drivers (on request of the printer PPD associated with them) often put a PJL header in front of the main PostScript part of the print file. Thus the printfile starts with "*<1B >%−12345X*" or "*<escape>%−12345X*" instead of "*%!PS*"). This leads to the CUPS daemon auto−typing the incoming file as a print−ready file, not

initiating a pass through the "pstops" filter (to speak more technical, it is not regarded as the generic MIME type "*application/postscript*", but as the more special MIME type "*application/cups.vnd−postscript*"), which therefore also leads to the page accounting in "*/var/log/cups/page_log*" not receiving the exact number of pages; instead the dummy page number of "1" is logged in a standard setup)

- the Adobe driver has more options to "mis−configure" the PostScript generated by it (like setting it inadvertedly to "*Optimize for Speed*", instead of "*Optimize for Portability*", which could lead to CUPS being unable to process it)
- the CUPS PostScript driver output sent by Windows clients to the CUPS server will be guaranteed to be auto−typed always as generic MIME type "*application/postscript*", thusly passingugh the CUPS "pstops" filter and logging the correct number of pages in the *page_log* for accounting and quota purposes
- the CUPS PostScript driver supports the sending of additional standard (IPP) print options by Win NT/2k/XP clients. Such additional print options are: naming the CUPS standard *banner pages* (or the custom ones, should they be installed at the time of driver download), using the CUPS "*page−label*" option, setting a *job−priority* and setting the *scheduled time of printing* (with the option to support additional useful IPP job attributes in the future).
- the CUPS PostScript driver supports the inclusion of the new "**cupsJobTicket"* comments at the beginning of the PostScript file (which could be used in the future for all sort of beneficial extensions on the CUPS side, but which will not disturb any other applications as they will regard it as a comment and simply ignore it).
- the CUPS PostScript driver will be the heart of the fully fledged CUPS IPP client for Windows NT/2K/XP to be released soon (probably alongside the first Beta release for CUPS 1.2).

7.10.9. Run "cupsaddsmb" (quiet Mode)

The **cupsaddsmb** command copies the needed files into your [print$] share. Additionally, the PPD associated with this printer is copied from /etc/cups/ppd/ to [print$]. There the files wait for convenient Windows client installations via "*Point and Print*". Before we can run the command successfully, we need to be sure that we can authenticate towards Samba. If you have a small network you are probably using user level security ("*security = user*"). Probably your root has already a Samba account. Otherwise, create it now, using "*smbpasswd*":

```
root#  smbpasswd −a root
New SMB password: [type in password 'secret']
Retype ew SMB password: [type in password 'secret']
```

Here is an example of a successfully run cupsaddsmb command.

```
root#  cupsaddsmb −U root infotec_IS2027
Password for root required to access localhost via SAMBA: [type in password 'secret']
```

To share ***all*** printers and drivers, use the *−a* parameter instead of a printer name. (Since cupsaddsmb "exports" the printer drivers to Samba, it should be obvious that it only works for queues with a CUPS driver associated.)

7.10.10. Run "cupsaddsmb" with verbose Output

Probably you want to see what's going on. Use the *−v* parameter to get a more verbose output (Attention −− you will see the root password for the Samba account printed on screen. If you use remote access, the password will go over the wire unencrypted!) The output below was edited for better readability −− all "\" at the end of a line indicate that I inserted an artificial line break plus some indentation here:

```
root # cupsaddsmb −U root −v infotec_2105
Password for root required to access localhost via SAMBA:
Running command: smbclient //localhost/print\$ −N −U'root%secret' −c 'mkdir W32X86;put   \
                 /var/spool/cups/tmp/3e98bf2d333b5 W32X86/infotec_2105.ppd;put           \
                 /usr/share/cups/drivers/cupsdrvr.dll W32X86/cupsdrvr.dll;put            \
                 /usr/share/cups/drivers/cupsui.dll W32X86/cupsui.dll;put                \
                 /usr/share/cups/drivers/cups.hlp W32X86/cups.hlp'
added interface ip=10.160.51.60 bcast=10.160.51.255 nmask=255.255.252.0
Domain=[CUPS−PRINT] OS=[Unix] Server=[Samba 2.2.7a]
NT_STATUS_OBJECT_NAME_COLLISION making remote directory \W32X86
putting file /var/spool/cups/tmp/3e98bf2d333b5 as \W32X86/infotec_2105.ppd (2328.8 kb/s) \
             (average 2328.8 kb/s)
putting file /usr/share/cups/drivers/cupsdrvr.dll as \W32X86/cupsdrvr.dll (9374.3 kb/s)  \
             (average 5206.6 kb/s)
putting file /usr/share/cups/drivers/cupsui.dll as \W32X86/cupsui.dll (8107.2 kb/s)      \
             (average 5984.1 kb/s)
putting file /usr/share/cups/drivers/cups.hlp as \W32X86/cups.hlp (3475.0 kb/s)          \
             (average 5884.7 kb/s)

Running command: rpcclient localhost −N −U'root%secret' −c 'adddriver "Windows NT x86"   \
                 "infotec_2105:cupsdrvr.dll:infotec_2105.ppd:cupsui.dll:cups.hlp:NULL:   \
                 RAW:NULL"'
cmd = adddriver "Windows NT x86" "infotec_2105:cupsdrvr.dll:infotec_2105.ppd:cupsui.dll: \
                 cups.hlp:NULL:RAW:NULL"
```

```
  Printer Driver infotec_2105 successfully installed.

  Running command: smbclient //localhost/print\$ −N −U'root%secret' −c 'mkdir WIN40;put    \
                   /var/spool/cups/tmp/3e98bf2d333b5 WIN40/infotec_2105.PPD; put          \
                   /usr/share/cups/drivers/ADFONTS.MFM WIN40/ADFONTS.MFM;put              \
                   /usr/share/cups/drivers/ADOBEPS4.DRV WIN40/ADOBEPS4.DRV;put            \
                   /usr/share/cups/drivers/ADOBEPS4.HLP WIN40/ADOBEPS4.HLP;put            \
                   /usr/share/cups/drivers/DEFPRTR2.PPD WIN40/DEFPRTR2.PPD;put            \
                   /usr/share/cups/drivers/ICONLIB.DLL
  WIN40/ICONLIB.DLL;put /usr/share/cups/drivers/PSMON.DLL WIN40/PSMON.DLL;'
  added interface ip=10.160.51.60 bcast=10.160.51.255 nmask=255.255.252.0
  Domain=[CUPS−PRINT] OS=[Unix] Server=[Samba 2.2.7a]
  NT_STATUS_OBJECT_NAME_COLLISION making remote directory \WIN40
  putting file /var/spool/cups/tmp/3e98bf2d333b5 as \WIN40/infotec_2105.PPD (2328.8 kb/s)  \
               (average 2328.8 kb/s)
  putting file /usr/share/cups/drivers/ADFONTS.MFM as \WIN40/ADFONTS.MFM (9368.0 kb/s)     \
               (average 6469.6 kb/s)
  putting file /usr/share/cups/drivers/ADOBEPS4.DRV as \WIN40/ADOBEPS4.DRV (9958.2 kb/s)   \
               (average 8404.3 kb/s)
  putting file /usr/share/cups/drivers/ADOBEPS4.HLP as \WIN40/ADOBEPS4.HLP (8341.5 kb/s)   \
               (average 8398.6 kb/s)
  putting file /usr/share/cups/drivers/DEFPRTR2.PPD as \WIN40/DEFPRTR2.PPD (2195.9 kb/s)   \
               (average 8254.3 kb/s)
  putting file /usr/share/cups/drivers/ICONLIB.DLL as \WIN40/ICONLIB.DLL (8239.9 kb/s)     \
               (average 8253.6 kb/s)
  putting file /usr/share/cups/drivers/PSMON.DLL as \WIN40/PSMON.DLL (6222.2 kb/s)         \
               (average 8188.5 kb/s)

  Running command: rpcclient localhost −N −U'root%secret' −c 'adddriver "Windows 4.0"      \
                   "infotec_2105:ADOBEPS4.DRV:infotec_2105.PPD:NULL:ADOBEPS4.HLP:          \
                   PSMON.DLL:RAW:ADOBEPS4.DRV,infotec_2105.PPD,ADOBEPS4.HLP,PSMON.DLL,     \
                   ADFONTS.MFM,DEFPRTR2.PPD,ICONLIB.DLL"'
  cmd = adddriver "Windows 4.0" "infotec_2105:ADOBEPS4.DRV:infotec_2105.PPD:NULL:          \
                   ADOBEPS4.HLP:PSMON.DLL:RAW:ADOBEPS4.DRV,infotec_2105.PPD,ADOBEPS4.HLP,  \
                   PSMON.DLL,ADFONTS.MFM,DEFPRTR2.PPD,ICONLIB.DLL"
  Printer Driver infotec_2105 successfully installed.

  Running command: rpcclient localhost −N −U'root%secret'                                  \
                             −c 'setdriver infotec_2105 infotec_2105'
  cmd = setdriver infotec_2105 infotec_2105
  Succesfully set infotec_2105 to driver infotec_2105.
```

If you look closely, you'll discover your root password was transfered unencrypted over the wire, so beware! Also, if you look further her, you'll discover error messages like NT_STATUS_OBJECT_NAME_COLLISION in between. They occur, because the directories WIN40 and W32X86 already existed in the [print$] driver download share (from a previous driver installation). They are harmless here.

7.10.11. Understanding cupsaddsmb

What has happened? What did cupsaddsmb do? −− There are five stages of the procedure

- FIRST: call the CUPS server via IPP and request the driver files and the PPD file for the named printer;
- SECOND: store the files temporarily in the local TEMPDIR (as defined in cupsd.conf);
- THIRD: connect via smbclient to the Samba server's [print$] share and put the files into the share's WIN40 (for Win95/98/ME) and W32X86/ (for WinNT/2k/XP) sub directories;
- FOURTH: connect via rpcclient to the Samba server and execute the "adddriver" command with the correct parameters;
- FIFTH: connect via rpcclient to the Samba server a second time and execute the "setdriver" command.

Note, that you can run the cupsaddsmb utility with parameters to specify one remote host as Samba host and a second remote host as CUPS host. Especially if you want to get a deeper understanding, it is a good idea try it and see more clearly what is going on (though in real life most people will have their CUPS and Samba servers run on the same host):

```
root:~# cupsaddsmb −H sambaserver −h cupsserver −v printername
```

7.10.12. How to recognize if cupsaddsm completed successfully

You *must* always check if the utility completed successfully in all fields. You need as a minimum these 3 messages amongst the output:

- "*Printer Driver infotec_2105 successfully installed.*" # (for the W32X86 == WinNT/2K/XP architecture...)

- "*Printer Driver infotec_2105 successfully installed.*" # (for the WIN40 == Win9x/ME architecture...)
- "*Succesfully set [printerXPZ] to driver [printerXYZ].*"

These messages probably not easily recognized in the general output. If you run cupsaddsmb with the "−a" parameter (which tries to prepare *all* active CUPS printer drivers for download), you might miss if individual printers drivers had problems to install properly. Here a redirection of the output will help you analyze the results in retrospective.

Note, that it is impossible to see any diagnostic output if you don't run cupsaddsmb in verbose mode. Therefore we strongly recommend to not use the default quiet mode. It will hide any problems from you which might occur.

7.10.13. cupsaddsmb with a Samba PDC

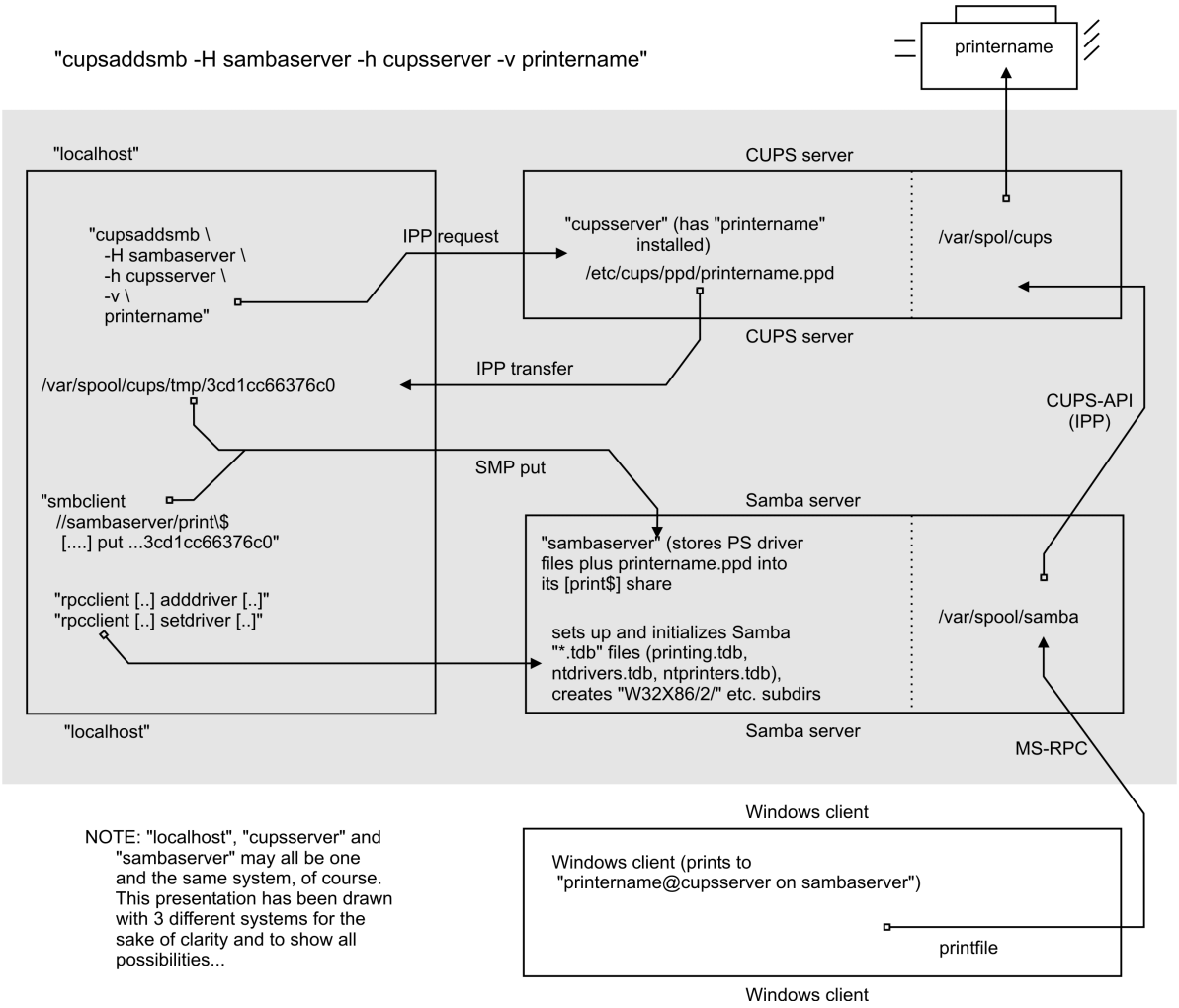
You can't get the standard cupsaddsmb command to run on a Samba PDC? You are asked for the password credential all over again and again and the command just will not take off at all? Try one of these variations:

```
root:~# cupsaddsmb −U DOMAINNAME\\root −v printername
root:~# cupsaddsmb −H SAMBA−PDC −U DOMAINNAME\\root −v printername
root:~# cupsaddsmb −H SAMBA−PDC −U DOMAINNAME\\root −h cups−server −v printername
```

(Note the two backslashes −− the first one is required to "escape" the second one).

7.10.14. cupsaddsmb Flowchart

Here is a chart about the procedures, commandflows and dataflows of the "cupaddsmb" command. (Note again: cupsaddsmb is not intended to and does not work with "raw" queues!)



7.10.15. Installing the PostScript Driver on a Client

After cupsaddsmb completed, your driver is prepared for the clients to use. Here are the steps you must perform to download and install it via "Point'n'Print". From a Windows client, browse to the CUPS/Samba server;

- open the "*Printers*" share of Samba in Network Neighbourhood;
- right−click on the printer in question;
- from the opening context−menu select "*Install..."* or " *Connect...*" (depending on the Windows version you use).

After a few seconds, there should be a new printer in your client's *local* "*Printers*" folder: On Windows XP it will follow a naming convention of "*PrinterName on SambaServer*". (In my current case it is "infotec_2105 on kde−bitshop"). If you want to test it and send your first job from an application like Winword, there the new printer appears in a "*\\SambaServer\PrinterName*" notation in the dropdown list of available printers.

NOTE: **cupsaddsmb** will only reliably work with CUPS version 1.1.15 or higher and Samba from 2.2.4. If it doesn't work, or if the automatic printer driver download to the clients doesn't succeed, you can still manually install the CUPS printer PPD on top of the Adobe PostScript driver on clients. Then point the client's printer queue to the Samba printer share for a UNC type of connection:

```
  net use lpt1: \\sambaserver\printershare /user:ntadmin
```

should you desire to use the CUPS networked PostScript RIP functions. (Note that user "ntadmin" needs to be a valid Samba user with the required privileges to access the printershare) This would set up the printer connection in the traditional "*LanMan*" way (not using MS−RPC).

7.10.16. Avoiding critical PostScript Driver Settings on the Client

Soooo −− printing works, but there are still problems. Most jobs print well, some don't at all. Some jobs have problems with fonts, which don't look very good. Some jobs print fast, and some are dead−slow Many of these problems can be greatly reduced or even completely eliminated if you follow a few guidelines. Remember, if your print device is not PostScript−enabled, you are treating your Ghostscript installation on your CUPS host with the output your client driver settings produce. Treat it well:

- Avoid the "*PostScript Output Option: Optimize for Speed*" settting −− use the "*Optimize for Portability*" instead (Adobe PostScript driver).
- Don't use the "*Page Independence: NO*" setting −− use " *Page Independence YES*" (CUPS PostScript Driver)
- Recommended is the "*True Type Font Downloading Option: Native True Type*" over "*Automatic*" and "*Outline*"; by all means avoid "*Bitmap*" (Adobe PostScript Driver)
- Choose "*True Type Font: Download as Softfont into Printer*" over the default "*Replace by Device Font*" (for exotic fonts you may need to change it back to get a printout at all) (Adobe)
- Sometimes you can choose "*PostScript Language Level*": in case of problems try "*2*" instead of "*3*" (the latest ESP Ghostscript package handels Level 3 PostScript very well) (Adobe).
- Say "*Yes*" to "*PostScript Error Handler*" (Adobe)

7.11. Installing PostScript Driver Files manually (using rpcclient)

Of course you can run all the commands which are embedded into the cupsaddsmb convenience utility yourself, one by one, and hereby upload and prepare the driver files for future client downloads.

1. prepare Samba (a CUPS printqueue with the name of the printer should be there −− we are providing the driver now);
2. copy all files to *[print$]:*
3. run *rpcclient adddriver* (for each client architecture you want to support):
4. run *rpcclient setdriver.*

We are going to do this now. First, read the man page on "rpcclient" to get a first idea. Look at all the printing related sub−commands. "*enumprinters*", "*enumdrivers*", " *enumports*", "*adddriver*", "setdriver" are amongst the most interesting ones. rpcclient implements an important part of the MS−RPC protocol. You can use it to query (and command) a Win NT (or 2K/XP) PC too. MS−RPC is used by Windows clients, amongst other things, to benefit from the "Point'n' Print" features. Samba can now mimic this too.

7.11.1. A Check for the man Page

First let's have a little check of the *rpcclient man page*. Here are two relevant passages:

```
adddriver <arch> <config>
  Execute an  AddPrinterDriver() RPC to install  the printer driver information on the server.
  Note that  the driver files should already exist in  the directory returned by getdriverdir.
```

```
  Possible  values for arch are  the same  as those for  the getdriverdir command.  The config
  parameter is defined as follows:
               Long Printer Name:\
               Driver File Name:\
               Data File Name:\
               Config File Name:\
               Help File Name:\
               Language Monitor Name:\
               Default Data Type:\
               Comma Separated list of Files

   Any empty fields should be entered as the string "NULL".

   Samba does not need to support the concept of  Print Monitors since these only apply to lo-
   cal printers  whose driver can make use of  a bi-directional link  for communication.  This
   field should be "NULL". On a remote NT print server,the Print Monitor for a driver must al-
   ready be installed prior to adding the driver or else the RPC will fail.

   [....]

setdriver <printername> <drivername>
  Execute  a SetPrinter() command  to update  the printer driver associated with  an installed
  printer. The printer driver must already be correctly installed on the print server.

  See also the enumprinters and enumdrivers commands for obtaining a list of installed printers
  and drivers.
```

7.11.2. Understanding the man Page

The *exact* format isn't made too clear by the man page, since you have to deal with some parameters containing spaces. Here is a better description for it. We have line-broken the command and indicated the breaks with "\". Usually you would type the command in one line without the linebreaks:

```
adddriver "Architecture" \
          "LongPrinterName:DriverFile:DataFile:ConfigFile:HelpFile:\
          LanguageMonitorFile:DataType:ListOfFiles,Comma-separated"
```

What the man pages denotes as a simple <config> keyword, does in reality consist of 8 colon-separated fields. The last field may take multiple (in some -- very insane! -- cases even 20 different additional files. This might sound confusing at first. Note, that what the man pages names the "LongPrinterName" in reality should rather be called the "Driver Name". You can name it anything you want, as long as you use this name later in the "*rpcclient ... setdriver*" command. For practical reasons, many name the driver the same as the printer.....

True -- it isn't simple at all. I hear you asking: *How doI know which files are "Driver File", "Data File", "Config File", "Help File" and "Language Monitor File" in each case?* -- For an answer you may want to have a look at how a Windows NT box with a shared printer presents the files to us. Remember, that this whole procedure has to be developed by the Samba Team by overhearing the traffic caused by Windows computers on the wire. We may as well turn to a Windows box now, and access it from a UNIX workstation. We will query it with "*rpcclient*" to see what it tells us and try to understand the man page more clearly which we've read just now.

7.11.3. Producing an Example by querying a Windows Box

We could run "*rpcclient*" with a " *getdriver*" or a "*getprinter*" subcommand (in level 3 verbosity) against it. Just sit down at UNIX or Linux workstation with the Samba utilities installed. Then type the following command:

```
rpcclient -U'USERNAME%PASSWORD' NT-SERVER-NAME -c 'getdriver printername 3'
```

From the result it should become clear which is which. Here is an example from my environment:

```
kde-bitshop:~# rpcclient -U'Danka%xxxx' W2KSERVER -c'getdriver "DANKA InfoStream Virtual Printer" 3'
cmd = getdriver "DANKA InfoStream Virtual Printer" 3

[Windows NT x86]
Printer Driver Info 3:
        Version: [2]
        Driver Name: [DANKA InfoStream]
        Architecture: [Windows NT x86]
        Driver Path: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\PSCRIPT.DLL]
        Datafile: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\INFOSTRM.PPD]
```

```
        Configfile: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\PSCRPTUI.DLL]
        Helpfile: [C:\WINNT\System32\spool\DRIVERS\W32X86\2\PSCRIPT.HLP]

        Dependentfiles: []
        Dependentfiles: []
        Dependentfiles: []
        Dependentfiles: []
        Dependentfiles: []
        Dependentfiles: []
        Dependentfiles: []

        Monitorname: []
        Defaultdatatype: []
```

Some printer drivers list additional files under the label "Dependentfiles" −− these would go into the last field "*ListOfFiles,Comma−separated*". For the CUPS PostScript drivers we don't need any (nor would we for the Adobe PostScript driver) −− therefore the field will get a "NULL" entry.

7.11.4. What is required for adddriver and setdriver to succeed

From the manpage (and from the quoted output of *cupsaddsmb*, above) it becomes clear, that you need to have the certain conditions in order to make the manual uploading and initializing of the driver files succeed. The two rpcclient subcommands (*adddriver* and *setdriver*) need to encounter the following pre−conditions to complete successfully:

- you are connected as "printer admin", or root (note, that this is ***not*** the "Printer Operators" group in NT, but the "*printer admin*" group, as defined in the [global] section of smb.conf);
- copy all required driver files to *\\sambaserver\print$\w32x86* and *\\sambaserver\print$\win40* as appropriate. They will end up in the "0" respective "2" subdirectories later −− for now ***don't*** put them there, they'll be automatically used by the *adddriver* subcommand.! (if you use "smbclient" to put the driver files into the share, note that you need to escape the "$": *smbclient //sambaserver/print\$ −U root*);
- the user you're connecting as must be able to write to the *print$* share and create subdirectories;
- the printer you are going to setup for the Windows clients, needs to be installed in CUPS already;
- the CUPS printer must be known to Samba, otherwise the " *setdriver*" subcommand fails with an NT_STATUS_UNSUCCESSFUL error. To check if the printer is known by Samba you may use the " *enumprinters*" subcommand to rpcclient. (A long−standing bug prevented a proper update of the printer list until every smbd process had received a SIGHUP or was restarted. Remember this in case you've created the CUPS printer just shortly ago and encounter problems −− try restart Samba...)

7.11.5. Manual Commandline Driver Installation in 15 little Steps

We are going to install a printer driver now by manually executing all required commands. As this may seem a rather complicated process at first, we go through the procedure step by step, explaining every single action item as it comes up.

7.11.6.1. First Step: Install the Printer on CUPS

```
kde−bitshop:~# lpadmin −p mysmbtstprn −v socket://10.160.51.131:9100 −E −P /home/kurt/canonIR85.ppd
```

This installs printer with the name "*mysmbtstprn*" to the CUPS system. The printer is accessed via a socket (a.k.a. JetDirect or Direct TCP/IP) connection. You need to be root for this step

7.11.6.2. Second Step (optional): Check if the Printer is recognized by Samba

```
kde4@kde−bitshop:~> rpcclient −Uroot%xxxx −c 'enumprinters' localhost | grep −C2 mysmbtstprn

       flags:[0x800000]
       name:[\\kde−bitshop\mysmbtstprn]
       description:[\\kde−bitshop\mysmbtstprn,,mysmbtstprn]
       comment:[mysmbtstprn]
```

This should show the printer in the list. If not, stop and re−start the Samba daemon (smbd), or send a HUP signal: " *kill −HUP `pidof smbd`*". Check again. Troubleshoot + repeat until success. (Note the "empty" field between the two commas in the "description" line. Here would the driver name appear if there was one already) You need to know root's Samba password (as set by the "smbpasswd" command) for this step and most of the following steps (alternatively you can authenticate as one of the users from the "write list" as defined in smb.conf for [print$].)

7.11.6.3. Third Step (optional): Check if Samba knows a Driver for the Printer

```
kde4@kde-bitshop:~> rpcclient -Uroot%xxxx -c 'getprinter mysmbtstprn 2' localhost | grep driver
        drivername:[]


kde4@kde-bitshop:~> rpcclient -Uroot%xxxx -c 'getprinter mysmbtstprn 2' localhost | grep -C4 driv
       servername:[\\kde-bitshop]
       printername:[\\kde-bitshop\mysmbtstprn]
       sharename:[mysmbtstprn]
       portname:[Samba Printer Port]
       drivername:[]
       comment:[mysmbtstprn]
       location:[]
       sepfile:[]
       printprocessor:[winprint]


kde-bitshop:~> rpcclient -U root%xxxx -c 'getdriver mysmbtstprn' localhost
result was WERR_UNKNOWN_PRINTER_DRIVER
```

Neither method of the 3 shown above should show a driver. This step is just to teach you the sight of that condition. An attempt to connect to the printer at this stage will prompt the message along the lines: "The server has not the required printer driver installed..."

7.11.6.4. Fourth Step: Put all required Driver Files into Samba's [print$]

```
kde4@kde-bitshop:~> smbclient //localhost/print\$ -U 'root%xxxx'                      \
                           -c 'cd W32X86;                                            \
                               put /etc/cups/ppd/mysmbtstprn.ppd mysmbtstprn.PPD;    \
                               put /usr/share/cups/drivers/cupsui.dll cupsui.dll;    \
                               put /usr/share/cups/drivers/cupsdrvr.dll cupsdrvr.dll; \
                               put /usr/share/cups/drivers/cups.hlp cups.hlp'
```

(Note that this command should be entered in one long single line. Line-breaks and the line-end indicating "\" has been inserted for readability reasons.) This step is *required* for the next one to succeed. It makes the driver files physically present in the [print$] share. However, clients would still not be able to install them, because Samba does not yet treat them as driver files. A client asking for the driver would still be presented with a "not installed here" message.

7.11.6.5. Fifth Step: Verify where the Driver Files are now

```
kde-bitshop:~#  ls -l /etc/samba/drivers/W32X86/
total 669
drwxr-sr-x    2 root     ntadmin       532 May 25 23:08 2
drwxr-sr-x    2 root     ntadmin       670 May 16 03:15 3
-rwxr--r--    1 root     ntadmin     14234 May 25 23:21 cups.hlp
-rwxr--r--    1 root     ntadmin    278380 May 25 23:21 cupsdrvr.dll
-rwxr--r--    1 root     ntadmin    215848 May 25 23:21 cupsui.dll
-rwxr--r--    1 root     ntadmin    169458 May 25 23:21 mysmbtstprn.PPD
```

The driver files now are in the W32X86 architecture "root" of [print$].

7.11.6.6. Sixth Step: Tell Samba that these are *Driver* Files ("adddriver")

```
kde4@kde-bitshop:~> rpcclient -Uroot%xxxx -c `adddriver "Windows NT x86" "mydrivername: \
                                       cupsdrvr.dll:mysmbtstprn.PPD:                  \
                                       cupsui.dll:cups.hlp:NULL:RAW:NULL"             \
                                       localhost

Printer Driver mydrivername successfully installed.
```

Note that your can not repeat this step if it fails. (It could fail because of a simple typo). It will most likely have moved a part of the driver files into the "2" subdirectory. If this step fails, you need to go back to the fourth step and repeat it, before you can try this one again. In this step you need to choose a name for your driver. It is normally a good idea to use the same name as is used for the printername; however, in big installations you may use this driver for a number of printers which have obviously different names. So the name of the driver is not fixed...

7.11.6.7. Seventh Step: Verify where the Driver Files are now

```
kde4@kde-bitshop:~> ls -l /etc/samba/drivers/W32X86/
total 1
```

```
drwxr-sr-x    2 root     ntadmin       532 May 25 23:22 2
drwxr-sr-x    2 root     ntadmin       670 May 16 03:15 3


kde4@kde-bitshop:~> ls -l /etc/samba/drivers/W32X86/2
total 5039
[....]
-rwxr--r--    1 root     ntadmin     14234 May 25 23:21 cups.hlp
-rwxr--r--    1 root     ntadmin    278380 May 13 13:53 cupsdrvr.dll
-rwxr--r--    1 root     ntadmin    215848 May 13 13:53 cupsui.dll
-rwxr--r--    1 root     ntadmin    169458 May 25 23:21 mysmbtstprn.PPD
```

Notice how step 6 did also move the driver files to the appropriate subdirectory. (Compare with the situation after step 5).

7.11.6.8. Eighth Step (optional): Verify if Samba now recognizes the Driver

```
kde-bitshop:~# rpcclient -Uroot%xxxx -c 'enumdrivers 3' localhost | grep -B2 -A5 mydrivername

Printer Driver Info 3:
       Version: [2]
       Driver Name: [mydrivername]
       Architecture: [Windows NT x86]
       Driver Path: [\\kde-bitshop\print$\W32X86\2\cupsdrvr.dll]
       Datafile: [\\kde-bitshop\print$\W32X86\2\mysmbtstprn.PPD]
       Configfile: [\\kde-bitshop\print$\W32X86\2\cupsui.dll]
       Helpfile: [\\kde-bitshop\print$\W32X86\2\cups.hlp]
```

Remember, this command greps for the name you did choose for the driver in step Six. This command must succeed before you can proceed.

7.11.6.9. Ninth Step: Tell Samba which Printer should use these Driver Files ("setdriver")

```
kde4@kde-bitshop:~>rpcclient -Uroot%xxxx -c 'setdriver mysmbtstprn mydrivername' localhost

Succesfully set mysmbtstprn to driver mydrivername
```

Since you can bind any printername (==printqueue) to any driver, this is a very convenient way to setup many queues which use the same driver. You don't need to repeat all the previous steps for the setdriver command to succeed. The only pre-conditions are: "enumdrivers" must find the driver and "enumprinters" must find the printer.

7.11.6.10. Tenth Step (optional): Verify if Samba has this Association recognized

```
kde4@kde-bitshop:~>rpcclient -Uroot%xxxx -c 'getprinter mysmbtstprn 2' localhost | grep driver
      drivername:[mydrivername]

kde4@kde-bitshop:~>rpcclient -Uroot%xxxx -c 'getprinter mysmbtstprn 2' localhost | grep -C4 driv
      servername:[\\kde-bitshop]
      printername:[\\kde-bitshop\mysmbtstprn]
      sharename:[mysmbtstprn]
      portname:[Done]
      drivername:[mydrivername]
      comment:[mysmbtstprn]
      location:[]
      sepfile:[]
      printprocessor:[winprint]

kde-bitshop:~> rpcclient -U root%xxxx -c 'getdriver mysmbtstprn' localhost
[Windows NT x86]
Printer Driver Info 3:
      Version: [2]
      Driver Name: [mydrivername]
      Architecture: [Windows NT x86]
      Driver Path: [\\kde-bitshop\print$\W32X86\2\cupsdrvr.dll]
      Datafile: [\\kde-bitshop\print$\W32X86\2\mysmbtstprn.PPD]
      Configfile: [\\kde-bitshop\print$\W32X86\2\cupsui.dll]
      Helpfile: [\\kde-bitshop\print$\W32X86\2\cups.hlp]
      Monitorname: []
      Defaultdatatype: [RAW]
      Monitorname: []
      Defaultdatatype: [RAW]

kde4@kde-bitshop:~> rpcclient -Uroot%xxxx -c 'enumprinters' localhost | grep mysmbtstprn
      name:[\\kde-bitshop\mysmbtstprn]
```

```
        description:[\\kde-bitshop\mysmbtstprn,mydrivername,mysmbtstprn]
        comment:[mysmbtstprn]
```

Compare these results with the ones from steps 2 and 3. Note that every single of these commands show the driver is installed. (Even the "*enumprinters*" command now lists the driver on the "description" line.)

7.11.6.11. Eleventh Step (optional): Tickle the Driver into a correct Device Mode

You certainly know how to install the driver on the client. In case you are not som familiar with Windows, here is a short recipe: browse the ntework neighbourhood, go to the Samba server, look for the shares. You should see all shared Samba printers. Double-click on the one in question, and the driver should get installed, and the network connection set up. An alternative way is to open the "Printers (and Faxes)" folder, right-click on the printer in question and select "Connect" or "Install". As a result, a new printer should have appeared in your client's local "Printers (and Faxes)" folder, named s.th. like "printersharename on Sambahostnam".

It is important that you execute this step as a Samba printer admin (as defined in smb.conf). Here is another method to do this on Windows XP. It uses a commandline, which you may type into the "DOS box" (type root's smbpassword when prompted):

```
C:\> runas /netonly /user:root "rundll32 printui.dll,PrintUIEntry /in /n \\sambacupsserver\mysmbtstprn"
```

Change any printer setting once (like *"portrait" -->> "landscape"*), click "Apply"; change the setting back.

7.11.6.12. Twelveth Step: Install the Printer on a Client ("Point'n'Print")

```
C:\SambaPrintHOWTO> rundll32 printui.dll,PrintUIEntry /in /n "\\sambacupsserver\mysmbtstprn"
```

If it doesn't work it could be a permission problem with the [print$] share.

7.11.6.13. Thirteenth Step (optional): Print a Test Page

```
C:\SambaPrintHOWTO> rundll32 printui.dll,PrintUIEntry /p /n "\\sambacupsserver\mysmbtstprn"
```

Then hit [TAB] 5 times, [ENTER] twice, [TAB] once and [ENTER] again and march to the printer. (YMMV... ;-)

7.11.6.14. Fourteenth Step (recommended): Study the Test Page

Hmmm.... just kidding! By now you know everything about printer installations and you don't need to read a word. Just put it in a frame and bolt it to the wall with the heading "MY FIRST RPCCLIENT-INSTALLED PRINTER". Or throw it away, will ya?

7.11.6.15. Fifteenth Step (obligatory): Enjoy. Jump. Celebrate your Success

```
kde-bitshop:~# echo "Cheeeeerioooooo! Success..." >> /var/log/samba/log.smbd
```

7.11.7. Troubleshooting revisited

The setdriver command will fail, if in Samba's mind the queue is not already there. You had promising messages about the

```
Printer Driver ABC successfully installed.
```

after the "adddriver" parts of the procedure? But you are also seeing a disappointing message like this one beneath?

```
result was NT_STATUS_UNSUCCESSFUL
```

It is not good enough that ***you*** can see the queue ***in CUPS***, using the "*lpstat -p ir85wm*" command. A bug in most recent versions of Samba prevents the proper update of the queuelist. The recognition of newly installed CUPS printers fails unless you re-start Samba or send a HUP to all smbd processes. To verify if this is the reason why Samba doesn't execute the setdriver command successfully, check if Samba "sees" the printer:

```
kde-bitshop: # rpcclient transmeta -N -U'root%secret' -c 'enumprinters 0'| grep  ir85wm
       printername:[ir85wm]
```

An alternative command could be this:

```
kde-bitshop: # rpcclient transmeta -N -U'root%secret' -c 'getprinter ir85wm'
       cmd = getprinter ir85wm
       flags:[0x800000]
       name:[\\transmeta\ir85wm]
       description:[\\transmeta\ir85wm,ir85wm,DPD]
```

```
        comment:[CUPS PostScript−Treiber für WinNT/2K/XP]
```

BTW, you can use these commands, plus a few more, of course, to install drivers on remote Windows NT print servers too!

7.12. The printing *.tdb Files

Some mystery is associated with the series of files with a *.tdb−suffix appearing in every Samba installation. They are "*connections.tdb*", "*printing.tdb*", "*share_info.tdb* ", "*ntdrivers.tdb*", "*unexpected.tdb*", "*brlock.tdb* ", "*locking.tdb*", "*ntforms.tdb*", "*messages.tdb* ", "*ntprinters.tdb*", "*sessionid.tdb*" and " *secrets.tdb*". What is their purpose?

7.12.1. Trivial DataBase Files

A Windows NT (Print) Server keeps track of all info needed to serve its duty toward its clients by storing entries in the Windows "Registry". Client queries are answered by reading from the registry, Administrator or user configuration settings are saved by writing into the Registry. Samba and Unix obviously don't have such a kind of Registry. Samba instead keeps track of all client related info in a series of *.tdb* files. (TDB = Trivial Data Base). These are often located in */var/lib/samba/* or */var/lock/samba/* . The printing related files are *ntprinters.tdb*, *printing.tdb*, *ntforms.tdb* and *ntdrivers.tdb*.

7.12.2. Binary Format

*.tdb files are not human readable. They are written in a binary format. "Why not ASCII?", you may ask. "After all, ASCII configuration files are a good and proofed tradition on UNIX." −− The reason for this design decision by the Samba Team is mainly performance. Samba needs to be fast; it runs a separate *smbd* process for each client connection, in some environments many thousand of them. Some of these smbds might need to write−access the same *.tdb file ***at the same time***. The file format of Samba's *.tdb files allows for this provision. Many smbd processes may write to the same *.tdb file at the same time. This wouldn't be possible with pure ASCII files.

7.12.3. Loosing *.tdb Files

It is very important that all *.tdb files remain consistent over all write and read accesses. However, it may happen that these files *do* get corrupted. (A "`kill −9 `pidof smbd`" while a write access is in progress could do the damage as well as a power interruption, etc.). In cases of trouble, a deletion of the old printing−related *.tdb files may be the only option. You need to re−create all print related setup after that. Or you have made a backup of the *.tdbs in time...

7.12.4. Using *tdbbackup*

Samba ships with a little utility which helps the root user of your system to back up your *.tdb files. If you run it with no argument, it prints a little usage message:

```
kde−bitshop:/etc/samba # tdbbackup
Usage: tdbbackup [options] <fname...>

Version:3.0a
  −h            this help message
  −s suffix     set the backup suffix
  −v            veryify mode (restore if corrupt)
```

Here is how I backed up my printing.tdb file:

```
kde−bitshop:/var/lock/samba # ls
.           browse.dat       locking.tdb     ntdrivers.tdb   printing.tdb    share_info.tdb
..          connections.tdb  messages.tdb    ntforms.tdb     printing.tdbkp  unexpected.tdb
brlock.tdb  gmon.out         namelist.debug  ntprinters.tdb  sessionid.tdb

kde−bitshop:/var/lock/samba # tdbbackup −s .bak printing.tdb
printing.tdb : 135 records

kde−bitshop:/var/lock/samba # ls −l printing.tdb*
−rw−−−−−−−    1 root     root        40960 May  2 03:44 printing.tdb
−rw−−−−−−−    1 root     root        40960 May  2 03:44 printing.tdb.bak
```

7.13. CUPS Print Drivers from Linuxprinting.org

CUPS ships with good support for HP LaserJet type printers. You can install the generic driver as follows:

```
lpadmin -p laserjet4plus -v parallel:/dev/lp0 -E -m laserjet.ppd
```

(The "-m" switch will retrieve the "laserjet.ppd" from the standard repository for not-yet-installed-PPDs, which CUPS typically stores in /usr/share/cups/model. Alternatively, you may use "-P /path/to/your.ppd").

The generic laserjet.ppd however does not support every special option for every LaserJet-compatible model. It constitutes a sort of "least denominator" of all the models. If for some reason it is ruled out to you to pay for the commercially available ESP Print Pro drivers, your first move should be to consult the database on http://www.linuxprinting.org/printer_list.cgi. Linuxprinting.org has excellent recommendations about which driver is best used for each printer. Its database is kept current by the tireless work of Till Kamppeter from MandrakeSoft, who is also the principal author of the foomatic-rip utility.

NOTE, that the former "cupsomatic" concept is now be replaced by the new, much more powerful "foomatic-rip". foomatic-rip is the successor of cupsomatic. cupsomatic is not longer maintained. Here is the new URL to the Foomatic-3.0 database: http://www.linuxprinting.org/driver_list.cgi. If you upgrade to foomatic-rip, don't forget to also upgrade to the new-style PPDs for your foomatic-driven printers. foomatic-rip will not work with PPDs generated for the old cupsomatic. The new-style PPDs are 100% compliant to the Adobe PPD specification. They are intended to be used by Samba and the cupsaddsmb utility also, to provide the driver files for the Windows clients also!

7.13.1. foomatic-rip and Foomatic explained

Nowadays most Linux distros rely on the utilities of Linuxprinting.org to create their printing related software (which, BTW, works on all UNIXes and on Mac OS X or Darwin too). It is not known as well as it should be, that it also has a very end-user friendly interface which allows for an easy update of drivers and PPDs, for all supported models, all spoolers, all operatings systems and all package formats (because there is none). Its history goes back a few years.

They recently have achieved the astonishing milestone of 1000 listed printer models. Linuxprinting.org keeps all the important facts about printer drivers, supported models and which options are available for the various driver/printer combinations in its "Foomatic" database. Currently there are 245 drivers in the database -- many drivers support various models, and many models may be driven by different drivers; it's your choice!

7.13.1.1. 690 "perfect" Printers

At present there are 690 devices dubbed as working "perfectly", 181 "mostly", 96 "partially" and 46 are "Paperweights". Keeping in mind that most of these are non-PostScript models (PostScript printers are automatically supported supported by CUPS in perfection, by using the manufacturer-provided Windows-PPD...), and that a multifunctional device never qualifies for "perfectly" if it doesn't also scan and copy and fax under GNU/Linux -- then this is a truely astonishing achievement. 3 years ago the number was not more than 500 -- and Linux or UNIX "printing" at the time wasn't anywhere near the quality it is today!

7.13.1.2. How the "Printing HOWTO" started it all...

A few years ago Grant Taylor started it all. The roots of today's Linuxprinting.org are in the first "Linux Printing HOWTO" which he authored. As a side-project to this document, which served many Linux users and admins to guide their first steps in this complicated and delicate setup (printing to a scientist is "applying a structured deposition of distinct patterns of ink or toner particles on paper substrates" ***;-)***, he started to build in a little Postgres database with info about the hardware and driver zoo that made up Linux printing of the time. This database became the core component of today's Foomatic collection of tools and data. In the meantime it has moved to an XML representation of the data blobs.

7.13.1.3. Foomatic's strange Name

"Why the funny name?", you ask. When it really took off, around spring 2000, CUPS was far less popular than today, and most systems used LPD, LPRng or even PDQ to print. CUPS shipped with a few generic "drivers" (good for a few hundred different printer models). These didn't support many device-specific options. CUPS also shipped with its own built-in rasterization filter ("pstoraster", derived from Ghostscript). On the other hand, CUPS had this brilliant support for *controlling* all printer options through standardized and well-defined "PPD files" (PostScript Printers Description files). Plus, CUPS was designed to be easily extensible.

Grant had in his database already a respectable compilation of facts about a lot more printers, and the Ghostscript "drivers" they run with. His idea, to generate PPDs from the database info and use them to make standard Ghostscript filters work

within CUPS, proofed to work very well. It also "killed several birds with one stone":

- It made all current and future Ghostscript filter developments available for CUPS;
- It made available a lot of additional printer models to CUPS users (because often the "traditional" Ghostscript way of printing was the only one available);
- It gave all the advanced CUPS options (web interface, GUI driver configurations) to users wanting (or needing) to use Ghostscript filters.

7.13.1.4. cupsomatic, pdqomatic, lpdomatic, directomatic....

It worked through a quickly−hacked up filter script named "cupsomatic". cupsomatic was running the printfile through Ghostscript, constructing automatically the rather complicated command line needed. It just required to be copied into the CUPS system to make it work. To "configure" the way cupsomatic controls the Ghostscript rendering process, it needs a CUPS−PPD. This PPD is generated directly from the contents of the database. For CUPS and the respective printer/filter combo another Perl script named "CUPS−O−Matic" did the PPD generation. After that was working, Grant implemented within a few days a similar thing for two other spoolers. Names chosen for the config−generator scripts were PDQ−O−Matic (for PDQ) and LPD−O−Matic (for −− you guessed it −− LPD); the configuration here didn't use PPDs but other spooler−specific files.

From late summer of that year, Till Kamppeter started to put work into the database. Till had been newly employed by MandrakeSoft to convert their printing system over to CUPS, after they had seen his FLTK−based XPP (a GUI frontend to the CUPS lp−command). He added a huge amount of new info and new printers. He also developed the support for other spoolers, like PPR (via ppromatic), GNUlpr and LPRng (both via an extended lpdomatic) and "spoolerless" printing (directomatic)....

Soooo, to answer your question: "Foomatic" is the general name for all the overlapping code and data behind the "*omatic" scripts.... −− Foomatic up to versions 2.0.x required (ugly) Perl data structures attached the Linuxprinting.org PPDs for CUPS. It had a different "*omatic" script for every spooler, as well as different printer configuration files..

7.13.1.5.The *Grand Unification* achieved...

This all has changed in Foomatic versions 2.9 (Beta) and now released "stable" 3.0. There is now the convergence of all *omatic scripts achieved: it is called the "foomatic−rip". This single one script is the unification of the previously different spooler−specific *omatic scripts. foomatic−rip is used by all the different spoolers alike. Because foomatic−rip can read PPDs (both, original PostScript printer PPDs and Linuxprinting.org−generated ones), all of a sudden all supported spoolers will have the power of PPDs at their disposal; users only need to plug "foomatic−rip" into their system.... For users there is improved media type and source support −− paper sizes and trays are easier to configure.

Also, the New Generation of Linuxprinting.org PPDs doesn't contain Perl data structures any more. If you are a distro maintainer and have used the previous version of Foomatic, you may want to give the new one a spin −− but don't forget to generate a new−version set of PPDs, via the new "foomatic−db−engine"! (Individual users just need to generate a single new PPD specific to their model by following the steps outlined in the Foomatic tutorial or further below...) This new development is truly amazing.

foomatic−rip is a very clever wrapper around the need to run Ghostscript with a different syntax, different options, different device selections and/or different filters for each different printer or different spooler. At the same time it can read the PPD associated with a print queue and modify the print job according to the user selections. Together with this comes the 100% compliance of the new Foomatic PPDs with the Adobe spec. Some really innovative features of the Foomatic concept is out for surprise to users: it will support custom paper sizes for many printers −− and it will support the printing on media drawn from different paper trays within the same job (in both cases: even where there is no support for this from Windows−based vendor printer drivers).

7.13.1.6. Driver Development outside

Most driver development itself does not happen on Linuxprinting.org. Drivers are done by independent maintainers. Linuxprinting.org just pools all the info, and stores it in its database. In addition, it also provides the Foomatic glue to integrate the many drivers into any modern (or legacy) printing system known to the world.

Speaking of the different driver development groups: most of the work is currently done in three projects. These are:

- Omni −− a Free Software project by IBM which tries to convert their printer driver knowledge from good−ol' OS/2 times into a modern, modular, universal driver architecture for Linux/Unix (still Beta). Currently supports 437 models.
- HPIJS −− a Free Software project by HP to provide the support for their own range of models (very mature, printing in most cases is perfect and provides true photo quality). Currently supports 369 models.

- Gimp−Print −− a Free software effort, started by Michael Sweet (also lead developer for CUPS), now directed by Robert Krawitz, which has achieved an amazing level of photo print quality (many Epson users swear, that its quality is better than the vendor drivers' provided by Epson for the Microsoft platforms). Currently supports 522 models.

7.13.1.7. Forums, Downloads, Tutorials, Howtos −− also for Mac OS X and commercial Unix

Linuxprinting.org today is the one−stop "shop" to download printer drivers, look for printer info and tutorials or solve printing problems in its popular forums. But it's not just GNU/Linux: also users and admins of commercial UNIX systems are going there, and the relatively new Mac OS X forum has turned out to be one of the most frequented ones after only a few weeks.

Linuxprinting.org and the Foomatic driver wrappers around Ghostscript are now a standard toolchain for printing on all the important distros. Most of them also have CUPS underneath. While in recent years most printer data had been added by Till (who works at Mandrake), many additional contributions came from engineers with SuSE, RedHat, Connectiva, Debian and others. Vendor−neutrality is an important goal of the Foomaticproject.

Note: Till Kamppeter from MandrakeSoft is doing an excellent job in his spare time to maintain Linuxprinting.org and Foomatic. (So if you use it often, please send him a note showing your appreciation).

7.13.1.8. Foomatic Database generates PPDs

The Foomatic database is an amazing piece of ingenuity in itself. Not only does it keep the printer and driver info. It is organized in a way that it can generate "PPD" files "on the fly" from its internal XML−based datasets. While these PPDs are modelled to the Adobe specification of "PostScript Printer Descriptions" (PPDs) −− the Linuxprinting.org/Foomatic−PPDs don't normally drive PostScript printers: they are used to describe all the bells and whistles you could ring or blow on an Epson Stylus inkjet, or a HP Photosmart or what−have−you. The main "trick" is one little additional line, not envisaged by the PPD specification, starting with the "*cupsFilter" keyword: it tells the CUPS daemon how to proceed with the PostScript print file. (Old−styple Foomatic−PPDs named the *cupsomatic* filter script, new−styple PPDs now call the *foomatic−rip*.). This filter script calls Ghostscript on the host system (the recommended variant is ESP Ghostscript) to do the rendering work. foomatic−rip knows which filter or internal device setting it should ask from Ghostscript to convert the PostScript printjob into a raster format ready for the target device. This usage of PPDs to describe the options of non−PS printers was the invention of the CUPS developers. The rest is easy: GUI tools (like KDE's marvellous "kprinter", or the GNOME "gtklp", "xpp" and the CUPS web interface) read the PPD too and use this info to present the available settings to the user as an intuitive menu selection.

7.13.2. foomatic−rip and Foomatic−PPD Download and Installation

Here are the steps to install a foomatic−rip driven "LaserJet 4 Plus" compatible printer in CUPS. (Note, that recent distributions of SuSE, UnitedLinux and Mandrake may ship with a complete package of Foomatic−PPDs plus the foomatic−rip utility. going directly to Linuxprinting.org ensures you to get the latest driver/PPD files):

- Surf to http://www.linuxprinting.org/printer_list.cgi.
- Check the complete list of printers in the database: http://www.linuxprinting.org/printer_list.cgi?make=Anyone
- There select your model and click on the link.
- You'll arrive at a page listing all drivers working with this model. (For all printers, there will always be *one* recommended driver. Try this one first....)
- In our case ("HP LaserJet 4 Plus"), we'll arrive here: http://www.linuxprinting.org/show_printer.cgi?recnum=HP−LaserJet_4_Plus
- The recommended driver is "ljet4".
- There are several links provided here. You should visit them all, if you are not familiar with the Linuxprinting.org database.
- There is a link to the database page for the "ljet4": http://www.linuxprinting.org/show_driver.cgi?driver=ljet4. On the driver's page, you'll find important and detailed info about how to use that driver within the various available spoolers.
- Another link may lead you to the homepage of the driver author or the driver.
- Important links are the ones which provide hints with setup instructions for CUPS (http://www.linuxprinting.org/cups−doc.html), PDQ (http://www.linuxprinting.org/pdq−doc.html), LPD, LPRng and GNUlpr (http://www.linuxprinting.org/lpd−doc.html) as well as PPR (http://www.linuxprinting.org/ppr−doc.html) or "spooler−less" printing (http://www.linuxprinting.org/direct−doc.html).
- You can view the PPD in your browser through this link: http://www.linuxprinting.org/ppd−o−matic.cgi?driver=ljet4&printer=HP−LaserJet_4_Plus&show=1.

- You can also (most importantly) generate and download the PPD: http://www.linuxprinting.org/ppd−o−matic.cgi?driver=ljet4&printer=HP−LaserJet_4_Plus&show=0.
- The PPD contains all the info needed to use our model and the driver; this is, once installed, working transparently for the user −− later you'll only need to choose resolution, paper size etc. from the web−based menu, or from the print dialog GUI, or from the commandline...
- Should you have ended up on the driver's page (http://www.linuxprinting.org/show_driver.cgi?driver=ljet4), you can choose to use the "PPD−O−Matic" online PPD generator program.
- Select the exact model and check either "download" or "display PPD file" and click on "Generate PPD file".
- If you save the PPD file from the browser view, please don't use "cut'n'past" (since it could possibly damage line endings and tabs, which makes the PPD likely to fail its duty), but use "Save as..." in your browser's menu. (Best is to use the "download" option from the web page directly).
- Another very interesting part on each driver page is the "***Show execution details***" button. If you select your printer model and click that button, you will get displayed a complete Ghostscript command line, enumerating all options available for that driver/printermodel combo. This is a great way to "Learn Ghostscript By Doing". It is also an excellent "cheat sheet" for all experienced users who need to re−construct a good command line for that damn printing script, but can't remember the exact syntax.... ;−)
- Some time during your visit to Linuxprinting.org, save the PPD to a suitable place on your harddisk, say "/path/to/my−printer.ppd". (If you prefer to install your printers with the help of the CUPS web interface, save the PPD to the "/usr/share/cups/model/" path and re−start cupsd.)
- Then install the printer with a suitable commandline, f.e.:
```
lpadmin −p laserjet4plus −v parallel:/dev/lp0 −E −P path/to/my−printer.ppd
```
- Note again this: for all the new−style "Foomatic−PPDs" from Linuxprinting.org, you also need a special "CUPS filter" named "foomatic−rip".Get the latest version of "foomatic−rip" from: http://www.linuxprinting.org/foomatic2.9/download.cgi?filename=foomatic−rip&show=0.
- The foomatic−rip Perlscript itself also makes some interesting reading (http://www.linuxprinting.org/foomatic2.9/download.cgi?filename=foomatic−rip&show=1), because it is very well documented by Till's inline comments. (Even non−Perl hackers will learn quite a bit about printing by reading it... ;−)
- Save foomatic−rip either directly in "/usr/lib/cups/filter/foomatic−rip" or somewhere in your $PATH (and don't forget to make it world−executable). Again, don't save by "copy'n'paste" but use the appropriate link, or the "Save as..." menu item in your browser.
- If you save foomatic−rip in your $PATH, create a symlink: "*cd /usr/lib/cups/filter/ ; ln −s `which foomatic−rip`*" (For CUPS to discover this new available filter at startup, you need to re−start cupsd).

Once you print to a printqueue set up with the Foomatic−PPD, CUPS will insert the appropriate commands and comments into the resulting PostScript jobfile. foomatic−rip is able to read and act upon these. foomatic−rip uses some specially encoded Foomatic comments, embedded in the jobfile. These in turn are used to construct (transparently for you, the user) the complicated ghostscript command line telling for the printer driver how exactly the resulting raster data should look like and which printer commands to embed into the data stream. −−

Summary. You need:

- A "foomatic+*something*" PPD −− but it this not enough to print with CUPS (it is only *one* important component)
- The "foomatic−rip" filter script (Perl) in /usr/lib/cups/filters/
- Perl to make foomatic−rip run
- Ghostscript (because it is doing the main work, controlled by the PPD/foomatic−rip combo) to produce the raster data fit for your printermodel's consumption
- Ghostscript *must* (depending on the driver/model) contain support for a certain "device", representing the selected "driver" for your model (as shown by "gs −h")
- foomatic−rip needs a new version of PPDs (PPD versions produced for cupsomatic don't work with foomatic−rip).

7.14. Page Accounting with CUPS

Often there are questions regarding "print quotas": Samba users (=Windows clients) should not be able to print beyond a certain amount of pages or data volume per day, week or month. This feature is dependent on the real print subsystem you're using. Samba's part is always to receive the job files from the clients (filtered *or* unfiltered) and hand it over to this printing subsystem.

Of course one could "hack" things with one's own scripts....But there is CUPS (Common Unix Printing System). CUPS supports "quotas". Quotas can be based on sizes of jobs or on the number of pages or both, and are spanning any time period you want.

7.14.1. Setting up Quotas

This is an example command how root would set a print quota in CUPS, assuming an existing printer named "quotaprinter":

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

This would limit every single user to print 100 pages or 1024 KB of data (whichever comes first) within the last 604.800 seconds (= 1 week).

7.14.2. Correct and incorrect Accounting

For CUPS to count correctly, the printfile needs to pass the CUPS "pstops" filter, otherwise it uses a "dummy" count of "1". Some printfiles don't pass it (eg: image files) but then those are mostly 1 page jobs anyway. This also means, proprietary drivers for the target printer running on the client computers and CUPS/Samba then spooling these files as "raw" (i.e. leaving them untouched, not filtering them), will be counted as "1-pagers" too!

You need to send PostScript from the clients (i.e. run a PostScript driver there) for having the chance to get accounting done. If the printer is a non-PostScript model, you need to let CUPS do the job to convert the file to a print-ready format for the target printer. This will be working for currently ~1.000 different printer models, see http://www.linuxprinting.org/printer_list.cgi

7.14.3. Adobe and CUPS PostScript Drivers for Windows Clients

Before CUPS-1.1.16 your only option was to use the Adobe PostScript Driver on the Windows clients. The output of this driver was not always passed through the "pstops" filter on the CUPS/Samba side, and therefor was not counted correctly (the reason is that it often --- depending on the "PPD" being used --- did write a "PJL"-header in front of the real PostScript which made CUPS to skip the pstops and go directly to the "pstoraster" stage).

From CUPS-1.1.16 onward you can use the "CUPS PostScript Driver for Windows NT/2K/XP clients" (it is tagged in the download area of http://www.cups.org/ as the "cups-samba-1.1.16.tar.gz" package). It is *not* working for Win9x/ME clients. But it:

- it guarantees to not write an PJL-header
- it guarantees to still read and support all PJL-options named in the driver PPD with its own means
- it guarantees the file going through the "pstops" filter on the CUPS/Samba server
- it guarantees to page-count correctly the printfile

You can read more about the setup of this combination in the manpage for "cupsaddsmb" (only present with CUPS installed, only current with CUPS 1.1.16).

7.14.4. The page_log File Syntax

These are the items CUPS logs in the "page_log" for every single *page* of a job:

```
* Printer name
* User name
* Job ID
* Time of printing
* the page number
* the number of copies
* a billing info string (optional)
* the host which sent the job (included since version 1.1.19)
```

Here is an extract of my CUPS server's page_log file to illustrate the format and included items:

```
infotec_IS2027 kurt 401 [22/Apr/2003:10:28:43 +0100] 1 3 #marketing 10.160.50.13
infotec_IS2027 kurt 401 [22/Apr/2003:10:28:43 +0100] 2 3 #marketing 10.160.50.13
infotec_IS2027 kurt 401 [22/Apr/2003:10:28:43 +0100] 3 3 #marketing 10.160.50.13
infotec_IS2027 kurt 401 [22/Apr/2003:10:28:43 +0100] 4 3 #marketing 10.160.50.13
DigiMaster9110 boss 402 [22/Apr/2003:10:33:22 +0100] 1 440 finance-dep 10.160.51.33
```

This was job ID "401", printed on "infotec_IS2027" by user "kurt", a 64-page job printed in 3 copies and billed to "#marketing"..., sent from IP address 10.160.50.13. The next job had ID "402", was sent by user "boss" from IP address 10.160.51.33,printed from one page 440 copies and is set to be billed to "finance-dep"

7.14.5. Possible Shortcomings

What flaws or shortcomings are there?

- the ones named above (wrongly logged job in case of printer hardware failure, etc.)
- CUPS really counts the job pages being processed in software* (going through the "RIP") rather than the physical sheets successfully leaving the printing device −− if there is a jam while printing the 5th sheet out of 1000 and the job is aborted by the printer, the "page count" will still show the figure of 1000 for that job
- all quotas are the same for all users (no flexibility to give the boss a higher quota than the clerk) no support for groups
- no means to read out the current balance or "used−up" number of current quota
- a user having used up 99 sheets of 100 quota will still be able to send and print a 1.000 sheet job
- a user being denied a job because of a filled−up quota doesn't get a meaningful error message from CUPS other than "client−error−not−possible".

7.14.6. Future Developments

But this is the best system out there currently. And there are huge improvements under development for CUPS 1.2:

- page counting will go into the "backends" (these talk directly to the printer and will increase the count in sync with the actual printing process −− a jam at the 5th sheet will lead to a stop in the counting)
- quotas will be handled more flexibly
- probably there will be support for users to inquire their "accounts" in advance
- probably there will be support for some other tools around this topic

7.14.7. Other Accounting Tools

PrintAnalyzer, pyKota, printbill, LogReport... (FIXME: There need to go a few sentences in here... FIXME)

more collected material...

A printer queue with *no* PPD associated to it is a "raw" printer and all files will go directly there as received by the spooler. The exceptions are file types "application/octet−stream" which need "passgh feature" enabled. "Raw" queues don't do any filtering at all, they hand the file directly to the CUPS backend. This backend is responsible for the sending of the data to the device (as in the "device URI" notation as lpd://, socket://, smb://, ipp://, http://, parallel:/, serial:/, usb:/ etc.)

"cupsomatic"/Foomatic are *not* native CUPS drivers and they don't ship with CUPS. They are a Third Party add−on, developed at Linuxprinting.org. As such, they are a brilliant hack to make all models (driven by Ghostscript drivers/filters in traditional spoolers) also work via CUPS, with the same (good or bad!) quality as in these other spoolers. "cupsomatic" is only a vehicle to execute a ghostscript commandline at that stage in the CUPS filtering chain, where "normally" the native CUPS "pstoraster" filter would kick in. cupsomatic by−passes pstoraster, "kidnaps" the printfile from CUPS away and re−directs it to go through Ghostscript. CUPS accepts this, because the associated CUPS−O−Matic−/Foomatic−PPD specifies:

```
   *cupsFilter:  "application/vnd.cups-postscript 0 cupsomatic"
```

This line persuades CUPS to hand the file to cupsomatic, once it has successfully converted it to the MIME type "application/vnd.cups−postscript". This conversion will not happen for Jobs arriving from Windows which are auto−typed "application/octet−stream", with the according changes in "/etc/cups/mime.types" in place.

CUPS is widely configurable and flexible, even regarding its filtering mechanism. Another workaround in some situations would be to have in "*/etc/cups/mime.types*" entries as follows:

```
   application/postscript           application/vnd.cups-raw  0  -
   application/vnd.cups-postscript  application/vnd.cups-raw  0  -
```

This would prevent all Postscript files from being filtered (rather, they will through the virtual "*nullfilter*" denoted with "−"). This could only be useful for PS printers. If you want to print PS code on non−PS printers (provided they support ASCII text printing) an entry as follows could be useful:

```
   */*           application/vnd.cups-raw  0  -
```

and would effectively send *all* files to the backend without further processing.

Lastly, you could have the following entry:

```
application/vnd.cups-postscript  application/vnd.cups-raw  0  my_PJL_stripping_filter
```

You will need to write a "*my_PJL_stripping_filter*" (could be a shellscript) that parses the PostScript and removes the unwanted PJL. This would need to conform to CUPS filter design (mainly, receive and pass the parameters printername, job−id, username, jobtitle, copies, print options and possibly the filename). It would be installed as world executable into "/usr/lib/cups/filters/" and will be called by CUPS if it encounters a MIME type "application/vnd.cups−postscript".

CUPS can handle "*−o job−hold−until=indefinite* ". This keeps the job in the queue "on hold". It will only be printed upon manual release by the printer operator. This is a requirement in many "central reproduction departments", where a few operators manage the jobs of hundreds of users on some big machine, where no user is allowed to have direct access. (The operators often need to load the proper paper type before running the 10.000 page job requested by marketing for the mailing, etc.).

7.15. Auto−Deletion or Preservation of CUPS Spool Files

Samba print files pass through two "spool" directories. One is the incoming directory managed by Samba, (set in the "*path = /var/spool/samba*" directive in the *[printers]* section of "smb.conf"). The other is the spool directory of your UNIX print subsystem. For CUPS it is normally "*/var/spool/cups/*", as set by the cupsd.conf directive "*RequestRoot /var/spool/cups* ".

7.15.1. Configuration Settings Docu

I am not sure, which one of your directories keeps the files. From what you say, it is most likely the Samba part.

For the CUPS part, you may want to consult:

```
http://localhost:631/sam.html#PreserveJobFiles and
http://localhost:631/sam.html#PreserveJobHistory and
http://localhost:631/sam.html#MaxJobs
```

There are the settings described for your CUPS daemon, which could lead to completed job files not being deleted.

7.15.2. Configuration Settings explained

"PreserveJobHistory Yes"
: keeps some details of jobs in cupsd's mind (well it keeps the "c12345", "c12346" etc. files in the CUPS spool directory, which do a similar job as the old−fashioned BSD−LPD control files). This is set to "Yes" as a default.

"PreserveJobFiles Yes"
: keeps the job files themselves in cupsd's mind (well it keeps the "d12345", "d12346" etc. files in the CUPS spool directory...). This is set to "No" as the CUPS default.

"MaxJobs 500"
: this directive controls the maximum number of jobs that are kept in memory. Once the number of jobs reaches the limit, the oldest completed job is automatically purged from the system to make room for the new one. If all of the known jobs are still pending or active then the new job will be rejected. Setting the maximum to 0 disables this functionality. The default setting is 0.

(There are also additional settings for "MaxJobsPerUser" and "MaxJobsPerPrinter"...)

7.15.3. Pre−conditions

For everything to work as announced, you need to have three things:

- a Samba−smbd which is compiled against "libcups" (Check on Linux by running "ldd `which smbd`")
- a Samba−smb.conf setting of "printing = cups"
- another Samba−smb.conf setting of "printcap = cups"

Note, that in this case all other manually set printing−related commands (like "print command", "lpq command", "lprm command", "lppause command" or "lpresume command") are ignored and they should normally have no influence what−so−ever on your printing.

7.15.4. Manual Configuration

If you want to do things manually, replace the "printing = cups" by "printing = bsd". Then your manually set commands may work (haven't tested this), and a "print command = lp −d %P %s; rm %s" may do what you need.

7.16. When *not* to use Samba to print to CUPS

[TO BE DONE]

7.17. In Case of Trouble.....

If you have more problems, post the output of these commands to the CUPS or Samba mailing lists (choose the one which seems more relevant to your problem):

```
grep −v ^# /etc/cups/cupsd.conf | grep −v ^$
grep −v ^# /etc/samba/smb.conf | grep −v ^$ | grep −v "^;"
```

(adapt paths as needed). These commands leave out the empty lines and lines with comments, providing the "naked settings" in a compact way. Don't forget to name the CUPS and Samba versions you are using! This safes bandwidth and makes for easier readability for experts (and you are expecting experts to read them, right? ;−)

[FIXME]

7.17.1. Where to find Documentation

[FIXME]

7.17.2. How to ask for Help

[FIXME]

7.17.3. Where to find Help

[FIXME]

Chapter 8: APPENDIX

8.1. Trouble Shooting Guidelines to fix typical Samba printing Problems

This is a short description of how to debug printing problems with Samba. This describes how to debug problems with printing from a SMB client to a Samba server, not the other way around. For the reverse see the examples/printing directory.

8.2. Printing *from* CUPS to Windows attached Printers

From time to time the question arises, how you can print ***to*** a Windows attached printer ***from*** Samba. Normally the local connection "Windows host <−−> printer" would be done by USB or parallel cable. But this doesn't matter to Samba. From here only a SMB connection needs to be opened to the Windows host. Of course, this printer must be "shared" first. As you have learned by now, CUPS uses "*backends*" to talk to printers and other servers. To talk to Windows shared printers you need to use the "*smb*" (surprise, surprise!) backend. Check if this is in the CUPS backend directory. This resides usually in */usr/lib/cups/backend/*. You need to find a "smb" file there. It should be a symlink to *smbspool* . And this smbspool must exist and be executable:

```
kde−bitshop:~# ls −l /usr/lib/cups/backend/
total 253
drwxr−xr−x    3 root     root          720 Apr 30 19:04 .
drwxr−xr−x    6 root     root          125 Dec 19 17:13 ..
−rwxr−xr−x    1 root     root        10692 Feb 16 21:29 canon
−rwxr−xr−x    1 root     root        10692 Feb 16 21:29 epson
lrwxrwxrwx    1 root     root            3 Apr 17 22:50 http −> ipp
−rwxr−xr−x    1 root     root        17316 Apr 17 22:50 ipp
−rwxr−xr−x    1 root     root        15420 Apr 20 17:01 lpd
−rwxr−xr−x    1 root     root         8656 Apr 20 17:01 parallel
−rwxr−xr−x    1 root     root         2162 Mar 31 23:15 pdfdistiller
lrwxrwxrwx    1 root     root           25 Apr 30 19:04 ptal −> /usr/local/sbin/ptal−cups
−rwxr−xr−x    1 root     root         6284 Apr 20 17:01 scsi
```

```
lrwxrwxrwx    1 root     root           17 Apr  2 03:11 smb -> /usr/bin/smbspool
-rwxr-xr-x    1 root     root         7912 Apr 20 17:01 socket
-rwxr-xr-x    1 root     root         9012 Apr 20 17:01 usb

kde-bitshop:/etc/samba/tmp # ls -l `which smbspool`
-rwxr-xr-x    1 root     root       563245 Dec 28 14:49 /usr/bin/smbspool
```

If this symlink doesn't exist, create it:

```
kde-bitshop:~# ln -s `which smbspool` /usr/lib/cups/backend/smb
```

smbspool has been written by Mike Sweet from the CUPS folks. It is included and ships with Samba. (It may also be used with other print subsystems than CUPS, to spool jobs to Windows printer shares.). To set up printer "winprinter" on CUPS, you need to have a "driver" for it. Essentially this means to convert the print data on the CUPS/Samba host to such a format the printer can digest. (The Windows host is unable to convert any files you may send.) This also means you should be able to print to the printer if it where hooked directly at your Samba/CUPS host. (For troubleshooting purposes, this is what you should do, to determine if that part of the process chain is in order. Then proceed to fix the network connection/authentication to the Windows host, etc.).

To install a printer with the smb backend on CUPS, use this command:

```
kde-bitshop:~# lpadmin -p winprinter -v smb://WINDOWSNETBIOSNAME/printersharename -P /path/to/PPD
```

The "*PPD*" must be fit to direct CUPS to generate the print data for the target model. (For PostScript printers just use the PPD that would be used with the Windows NT PostScript driver). But what to do, if the printer is only accessible with a password? Or if the printer's host is part of another workgroup? This is provided for: then you can include the required parameters as part of the "*smb://*" device-URI. Like this:

```
smb://WORKGROUP/WINDOWSNETBIOSNAME/printersharename
smb://username:password@WORKGROUP/WINDOWSNETBIOSNAME/printersharename
smb://username:password@WINDOWSNETBIOSNAME/printersharename
```

Note that the device-URI will be visible in the process list of the Samba server (f.e. when someone uses the "*ps -aux*" command on Linux), even if the username and passwords are sanitized before they get written into the log files. So this is an inherently insecure option. It is the only one. Don't use it if you want to protect your passwords. Better share the printer in a way that doesn't require a password! Printing will only work if you have a working netbios name resolution up and running. (You don't necessarily need to have smbd running -- but who wants that? :-)

8.3. More CUPS filtering Chains

The following diagrams reveal how CUPS handles print jobs.

```
##############################################################################
#
# CUPS in and of itself has this (general) filter chain (CAPITAL
# letters are FILE-FORMATS or MIME types, other are filters (this is
# true for pre-1.1.15 of pre-4.3 versions of CUPS and ESP PrintPro):
#
# SOMETHNG-FILEFORMAT
#      |
#      V
#     something tops
#      |
#      V
# APPLICATION/POSTSCRIPT
#      |
#      V
#     pstops
#      |
#      V
# APPLICATION/VND.CUPS-POSTSCRIPT
#      |
#      V
#     pstoraster   # as shipped with CUPS, independent from any Ghostscipt
#      |           # installation on the system
#      |  (= "postscipt interpreter")
#      V
# APPLICATION/VND.CUPS-RASTER
#      |
```

```
#      V
#     rastertosomething  (f.e. Gimp-Print filters may be plugged in here)
#      |   (= "raster driver")
#      V
# SOMETHING-DEVICE-SPECIFIC
#      |
#      V
#     backend
#
#
# ESP PrintPro has some enhanced "rastertosomething" filters as compared to
# CUPS, and also a somewhat improved "pstoraster" filter.
#
# NOTE: Gimp-Print and some other 3rd-Party-Filters (like TurboPrint) to
#       CUPS and ESP PrintPro plug-in where rastertosomething is noted.
#
##############################################################################

##############################################################################
#
# This is how "cupsomatic" comes into play:
# =========================================
#
# SOMETHNG-FILEFORMAT
#      |
#      V
#    somethingtops
#      |
#      V
# APPLICATION/POSTSCRIPT
#      |
#      V
#    pstops
#      |
#      V
# APPLICATION/VND.CUPS-POSTSCRIPT -----------------+
#      |                                           V
#      V                                         cupsomatic
#    pstoraster                                  (constructs complicated
#      |  (= "postscipt interpreter")            Ghostscript commandline
#      |                                         to let the file be
#      V                                         processed by a
# APPLICATION/VND.CUPS-RASTER                    "-sDEVICE=s.th."
#      |                                         call...)
#      V                                          |
#    rastertosomething                           V
#      |    (= "raster driver")     +-------------------------+
#      |                            | Ghostscript at work.... |
#      V                            |                         |
# SOMETHING-DEVICE-SPECIFIC         *-------------------------+
#      |                                          |
#      V                                          |
#    backend <-----------------------------------+
#      |
#      V
#    THE PRINTER
#
#
# Note, that cupsomatic "kidnaps" the printfile after the
# "APPLICATION/VND.CUPS-POSTSCRPT" stage and deviates it gh
# the CUPS-external, systemwide Ghostscript installation, bypassing the
# "pstoraster" filter (therefor also bypassing the CUPS-raster-drivers
# "rastertosomething", and hands the rasterized file directly to the CUPS
# backend...
#
# cupsomatic is not made by the CUPS developers. It is an independent
# contribution to printing development, made by people from
# Linuxprinting.org. (see also http://www.cups.org/cups-help.html)
#
# NOTE: Gimp-Print and some other 3rd-Party-Filters (like TurboPrint) to
#       CUPS and ESP PrintPro plug-in where rastertosomething is noted.
#
##############################################################################

##############################################################################
#
# And this is how it works for ESP PrintPro from 4.3:
# ===================================================
#
```

```
# SOMETHNG-FILEFORMAT
#      |
#      V
#     somethingtops
#      |
#      V
# APPLICATION/POSTSCRIPT
#      |
#      V
#     pstops
#      |
#      V
# APPLICATION/VND.CUPS-POSTSCRIPT
#      |
#      V
#     gsrip
#      |  (= "postscipt interpreter")
#      V
# APPLICATION/VND.CUPS-RASTER
#      |
#      V
#     rastertosomething  (f.e. Gimp-Print filters may be plugged in here)
#      |   (= "raster driver")
#      V
# SOMETHING-DEVICE-SPECIFIC
#      |
#      V
#     backend
#
# NOTE: Gimp-Print and some other 3rd-Party-Filters (like TurboPrint) to
#       CUPS and ESP PrintPro plug-in where rastertosomething is noted.
#
##############################################################################

##############################################################################
#
# This is how "cupsomatic" would come into play with ESP PrintPro:
# ================================================================
#
#
# SOMETHNG-FILEFORMAT
#      |
#      V
#    somethingtops
#      |
#      V
# APPLICATION/POSTSCRIPT
#      |
#      V
#    pstops
#      |
#      V
# APPLICATION/VND.CUPS-POSTSCRIPT -----------------+
#      |                                           V
#      V                                          cupsomatic
#    gsrip                                        (constructs complicated
#      |  (= "postscipt interpreter")             Ghostscript commandline
#      |                                          to let the file be
#      V                                          processed by a
# APPLICATION/VND.CUPS-RASTER                     "-sDEVICE=s.th."
#      |                                          call...)
#      V                                           |
#    rastertosomething                            V
#      |   (= "raster driver")      +--------------------------+
#      |                            | Ghostscript at work.... |
#      V                            |                         |
# SOMETHING-DEVICE-SPECIFIC         *--------------------------+
#      |                                          |
#      V                                          |
#    backend <-----------------------------------+
#      |
#      V
#    THE PRINTER
#
# NOTE: Gimp-Print and some other 3rd-Party-Filters (like TurboPrint) to
#       CUPS and ESP PrintPro plug-in where rastertosomething is noted.
#
##############################################################################
```

```
##############################################################################
#
# And this is how it works for CUPS from 1.1.15:
# ==============================================
#
# SOMETHNG-FILEFORMAT
#      |
#      V
#     somethingtops
#      |
#      V
# APPLICATION/POSTSCRIPT
#      |
#      V
#     pstops
#      |
#      V
# APPLICATION/VND.CUPS-POSTSCRIPT-----+
#                  +------------------v-------------------------------+
#                  | Ghostscript                                      |
#                  | at work...                                       |
#                  | (with                                            |
#                  | "-sDEVICE=cups")                                 |
#                  |                                                  |
#                  |         (= "postscipt interpreter")              |
#                  |                                                  |
#                  +------------------v-------------------------------+
#                                     |
# APPLICATION/VND.CUPS-RASTER >-------+
#      |
#      V
#     rastertosomething
#      |   (= "raster driver")
#      V
# SOMETHING-DEVICE-SPECIFIC
#      |
#      V
#     backend
#
#
# NOTE: since version 1.1.15 CUPS "outsourced" the pstoraster process to
#       Ghostscript. GNU Ghostscript needs to be patched to handle the
#       CUPS requirement; ESP Ghostscript has this builtin. In any case,
#       "gs -h" needs to show up a "cups" device. pstoraster is now a
#       calling an appropriate "gs -sDEVICE=cups..." commandline to do
#       the job. It will output "application/vnd.cup-raster", which will
#       be finally processed by a CUPS raster driver "rastertosomething"
#       Note the difference to "cupsomatic", which will *not* output
#       CUPS-raster, but a final version of the printfile, ready to be
#       sent to the printer. cupsomatic also doesn't use the "cups"
#       devicemode in Ghostscript, but one of the classical devicemodes....
#
# NOTE: Gimp-Print and some other 3rd-Party-Filters (like TurboPrint) to
#       CUPS and ESP PrintPro plug-in where rastertosomething is noted.
#
##############################################################################

##############################################################################
#
# And this is how it works for CUPS from 1.1.15, with cupsomatic included:
# ========================================================================
#
# SOMETHNG-FILEFORMAT
#      |
#      V
#     somethingtops
#      |
#      V
# APPLICATION/POSTSCRIPT
#      |
#      V
#     pstops
#      |
#      V
# APPLICATION/VND.CUPS-POSTSCRIPT-----+
#                  +------------------v-------------------------------+
#                  | Ghostscript        . Ghostscript at work....     |
#                  | at work...         . (with "-sDEVICE=            |
#                  | (with              .            s.th."          |
```

```
#                  | "-sDEVICE=cups")   .                            |
#                  |                    .                            |
#                  | (CUPS standard)    .      (cupsomatic)          |
#                  |                    .                            |
#                  |          (= "postscript interpreter")           |
#                  |                    .                            |
#                  +------------------v-------------v---------------+
#                                     |             |
# APPLICATION/VND.CUPS-RASTER >-------+             |
#      |                                            |
#      V                                            |
#     rastertosomething                            |
#      |   (= "raster driver")                      |
#      V                                            |
# SOMETHING-DEVICE-SPECIFIC >-------------------------+
#      |
#      V
#     backend
#
#
# NOTE: Gimp-Print and some other 3rd-Party-Filters (like TurboPrint) to
#       CUPS and ESP PrintPro plug-in where rastertosomething is noted.
#
##############################################################################
```

8.4. my TO-BO-DONE list...

Replace the ASCII art flowcharts by properly drawn PNG ones (any volunteers?)

Examples for customized filters/backends...

Where to find help...

Hacking CUPS (a "PDF Distilling Service" for all network clients -- customized backends and filters -- customized web interface -- an "add printer command" --...)

Caveats for the PostScript Driver Settings on the Win clients

Document the "3" subdir in [print$]/W32X86/

A "FAQ"?

Screenshots were applicable

Other printing protocols between Windows and UNIX

##

8.5. Troubleshooting Tips

Win9x client can't install driver

For Win9x clients keep the printer names at 8 chars (or "8 plus 3 chars suffix") max -- otherwise the driver files won't get transferred when you want to download them from Samba....

testparm

Run "*testparm*": It will tell you if smb.conf parameters are in the wrong section. Many people have had the "printer admin" parameter in the *[printers]* section and experienced problems. "testparm" will tell you if it sees this...

"cupsaddsmb" keeps asking for a root password in a neverending loop

Have you "*security = user*"? Have you used "smbpasswd" to give root a Samba account? You can do 2 things: open another terminal and execute "*smbpasswd -a root*" to create the account, and continue with entering the password into the first terminal. Or break out of the loop by hitting ENTER twice (without trying to type a password).

"cupsaddsmb" gives "No PPD file for printer..." message (but I swear there is one!)

-->** Have you enabled printer sharing on CUPS? (This means: do you have a "`<Location /printers>....</Location>`" section in CUPS server's *cupsd.conf* which doesn't deny access to the host you run "cupsaddsmb" from? It ***could be an issue if you use cupsaddsmb remotely, or if you use it with a "`-h`" parameter: "`cupsaddsmb -H sambaserver -h cupsserver -v printername`").

-->** Is your "`TempDir`" directive in ***cupsd.conf set to a valid value and is it writeable?

I can't connect client to Samba printer.

Use *smbstatus* to check which user you are from Samba's point of view. Do you have the privileges to write into the [print$] share?

I can't reconnect to Samba under a new account from Win2K/XP

Once you are connected as the "wrong" user (for example as "nobody", which often occurs if you have "*map to guest = bad user*"), Windows Explorer will not accept an attempt to connect again as a different user. There won't be any byte transfered on the wire to Samba, but still you'll see a stupid error message which makes you think that Samba has denied access.

Use *smbstatus* to check for active connections. Kill the PIDs. You still can't re−connect and get the dreaded "You can't connect with a second account from the same machine" message, as soon as you are trying? And you don't see any single byte arriving at Samba (see logs; use "ethereal") indicating a renewed connection attempt? Shut all Explorer Windows. This makes Windows forget what it has cached in its memory as established connections. Then re−connect as the right user. Best method is to use a DOS terminal window and *first* do "*net use z: \\SAMBAHOST\print$ /user:root*".

Check with "smbstatus" that you are connected under a different account. Now open the "Printers" folder (on the Samba server in the " *Network Neighbourhood*"), right−click the printer in question and select "*Connect...*"

Prevent to be connected to the Samba server as the "wrong" user

You see per "smbstatus" that you are connected as user "nobody" −− while you wanted to be "root" or "printeradmin"? This is probably due to "map to guest = bad user", which silently connects you under the guest account, when you gave (maybe by accident) an incorrect username. Remove "map to guest", if you want to prevent this.

Upgrading to CUPS drivers from Adobe drivers on NT/2K/XP clients gives problems

First delete all "old" Adobe−using printers. Then delete all "old" Adobe drivers. (On Win2K/XP, right−click in background of "Printers" folder, select "Server Properties...", select tab "Drivers" and delete here).

I can't use "cupsaddsmb"on a Samba server which is a PDC

Do you use the "naked" root user name? Try to do it this way: " *cupsaddsmb −U DOMAINNAME\\root −v printername*" (Note the two backslashes −− the first one is required to "escape" the second one).

I deleted a printer on Win2K −− but I still see its driver

Deleting a printer on the client won't delete the driver too. (To verify, right−click on the white background of the "Printers" folder, select "Server Properties" and click on the "Drivers" tab....) These same old driver will be re−used when you try to install a printer with the same name.... If you want to update to a new driver, delete the old ones first. Deletion is only possible if no other printer uses the same driver.

Win2K/XP "Local Security Policies

"*Local Security Policies*" may not allow the installation of unsigned drivers. "Local Security Policies" may not allow the installation of printer drivers at all.

WinXP clients −− Administrator can not install printers for all local users

Windows XP handles SMB printers on a "per−user" basis. This means every user needs to install the printer himself. To have a printer available for everybody, you might want to use the built−in IPP client capabilities of WinXP. Add a printer with the print path of " *http://cupsserver:631/printers/printernam*e". Still looking into this one.... (probably a "logon script" could automatically install printers for all users?).

"Print Change Notify" functions on NT−clients

For "print change notify" functions on NT++ clients, these need to run the "Server" service first (re−named to "*File & Print Sharing for MS Networks*" in XP).

WinXP−SP1

WinXP−SP1 introduced this "*Point and Print Restriction Policy*" (this restriction doesn't apply to "Administrator" or "Power User" groups of users). In Group Policy Object Editor: go to "*User Configuration −−> Administrative Templates −−> Control Panel −−> Printers*". The policy is automatically set to "*Enabled*" and the "*Users can only Point and Print to machines in their Forest* ". You probably need to change it to "*Disabled*" or "*Users can only Point and Print to these servers*" in order to make driver downloads from Samba possible...

I can't set and save default print options for all users on Win2K/XP

How are you doing it? I bet the wrong way.... (It is not very easy to find out, though). There are 3 different ways to bring you to a dialog that *seems* to set everything. All three dialogs *look* the same. Only one of them *does* what you intend. (You need to be Administrator or Print Administrator to do it for all users)

Here is how I do in on XP (my interface is German −− my re−translation into English might not be accurate):

A. the first "wrong" way:

- ◊ 1. Open the "*Printers*" folder.
- ◊ 2. Right−click on the printer ("*remoteprinter on cupshost*") and select in context menu "*Print Settings..."*
- ◊ 3. Look at this dialog closely and remember how it looks like.

B. the second, another "wrong" way:

- ◊ 1. Open the "*Printers*" folder.
- ◊ 2 Right−click on the printer ("*remoteprinter on cupshost*") and select in context menu "*Properties*"
- ◊ 3. Click on the "*General*" tab
- ◊ 4. Click on the button "*Print Settings...*"
- ◊ 5. A new dialog opens. Keep this dialog open and go back to the parent dialog.

C. the third, the "correct" way:

(Should you do it from the beginning, just do steps 1. + 2. from above "B.")

- ◊ 3. Click on the "*Advanced*" tab.

◊ 4. Click on the "*Standard Settings...*" (maybe "*Printing Defaults...*") button. ***(*)***
◊ 5. On any of the two new tabs, click on the "*Advanced...*" (maybe "More..."?) button. ***(*)***
◊ 6. A new dialog opens. Compare this one to the other, identical looking one from "B.5" or A.3".

Do you see any difference? I don't either... However, only the last one, which you arrived at with steps "C.1.−6." will safe any settings to become permanent and be the defaults for new users. If you want all clients to get the same defaults, you need to conduct these steps *as* *Administrator* (*printer admin* in smb.conf) *before* a client downloads the driver. (The clients can later set their own *per−user defaults* by following the procedures ***A.*** or ***B.*** above...)

[(*) *Could someone with an English version of Win2K/XP please verify my translations and post corrections? Thanks. −kp−]*

What are the most common blunders in driver settings on Windows clients?

Don't use "*Optimize for Speed*" −− use "*Optimize for Portability*" instead (Adobe PS Driver)
Don't use "*Page Independence: No*" −− always settle with " *Page Independence: Yes*" (Microsoft PS Driver and CUPS PS Driver for WinNT/2K/XP)
If there are problems with fonts: use "*Download as Softfont into printer*" (Adobe PS Driver). For "*TrueType Download Options*" choose "*Outline*".
Use PostScript Level 2, if you are having trouble with a non−PS printer, and if there is a choice.

I can't make "cupsaddsmb" work with newly installed printer

Symptom: the last command of "*cupsaddsmb*" doesn't complete successfully:

```
cmd = setdriver printername printername
result was NT_STATUS_UNSUCCESSFUL
```

then possibly the printer was not yet "recognized" by Samba... Did it show up in "*Network Neighbourhood*"? Did it show up in " *rpcclient hostname −c 'enumprinters'*"? Restart smbd (or send a "kill −HUP" to all processes listed by "*smbstatus*" and try again.

My permissions on /var/spool/samba/ get reset after each reboot

Have you by accident set the CUPS spool directory to the same location?? ("*RequestRoot /var/spool/samba/*" −− or the other way round: "*/var/spool/cups/*" is set as "path" in the `[printers]` section). These *must* be different. Set *RequestRoot* in *cupsd.conf* to */var/spool/cups/* and "*path = /var/spool/samba*" in the [printers] section of smb.conf. Otherwise cupsd will sanitize permissions to its spool directory with each restart, and printing will not work reliably.

My printers work fine −− just the printer "lp" does intermittently swallows jobs and spits out completely different ones

It is a very bad idea to name any printer "lp". This is the traditional Unix name for the default printer. CUPS may be set up to do an automatic creation of "Implicit Classes". This means, to group all printers with the same name to a pool of devices, and loadbalancing the jobs across them in a round−robin fashion. Chances are high that someone else has an "lp" named printer too. You may receive his jobs and send your own to his device unwittingly. To have tight control over the printer names, set "BrowseShortNames No". It will present any printer as "printername@cupshost" then, giving you a better control over what may happen in a large networked environment.

How do I "watch" my Samba server?

You can use "*tail −f /var/log/samba/log.smbd*" (you may need a different path) to see a live scrolling of all log messages. " *smbcontrol smbd debuglevel*" tells you which verbosity goes into the logs. "*smbcontrol smbd debug 3*" sets the verbosity to a quite high level (you can choose from 0 to 10 or 100). This works "on the fly", without the need to restart the smbd daemon. Don't use more than 3 initially −− or you'll drown in an ocean of messages.

I can't use Samba from my WinXP Home box, while access from WinXP Prof works flawlessly

You have our condolences! WinXP home has been completely neutered by Microsoft as compared to WinXP Prof: you can not log into a WinNT domain. It cannot join a Win NT domain as a member server. While it is possible to access domain resources, users don't have "single sign−on". They need to supply username and password each time they connect to a resource. Logon scripts and roaming profiles are not supported. It can server file and print shares −− but only in "share−mode security" level. It can not use "user−mode security" (what Windows 95/98/ME still can do).

Where do I find the Adobe PostScript driver files I need for "cupsaddsmb"?

Use "*smbclient*" to connect to any Windows box with a shared PostScript printer: "*smbclient //windowsbox/print\$ −U guest*". You can navigate to the *W32X86/2* subdir to "*mget ADOBE**" and other files or to "*WIN40/0*" to do the same. −− Another option is to download the *.exe packaged files from the Adobe website.