CUPS Driver Development Kit Manual

CUPS−DDK−1.0

Easy Software Products
Copyright 1997−2004, All Rights Reserved

Table of Contents

Preface **1**
- Software Overview 1
- Document Overview 2
- Notation Conventions 2
- Abbreviations 3
- Other References 4
- Providing Feedback 4

1 − Building and Installing the CUPS DDK **5**
- Before You Begin 5
- Building the DDK 5
  - Configuring the Software 5
  - Compiling the Software 6
- Installing the DDK 6
  - Installing Using the Makefile 6
  - Installing Using EPM 7

2 − Getting Started with the CUPS DDK **9**
- The Basics 9
- Using the PPD Compiler 10
- Driver Information Files 10
  - A Simple Example 10
  - Grouping and Inheritance 13
  - Defining Constants 13
  - Defining Color Support 15
  - Defining Custom Options and Option Groups 17
  - Defining Constraints 18
- Importing Existing PPD Files 20

3 − Developing Raster Printer Drivers **21**
- The DDK Drivers 21
- The HP−PCL Driver 21
  - ModelNumber Constants 22
  - Writing a Basic HP LaserJet Driver 23
  - PJL Attributes 27
  - Adding PJL Options to the Basic LaserJet Driver 28
- The ESC/P Driver 32
  - ModelNumber Constants 32
  - Writing a Basic Epson Stylus Photo R300 Driver 33
  - Epson Remote Mode Attributes 37
  - Adding Remote Mode Commands to the R300 Driver 38

4 − Developing PostScript Printer Drivers **41**
- Overview of PostScript Driver Development 41
  - Required Attributes 41
  - Query Commands 42
  - Adding Filters 43
- Importing Existing PostScript Drivers 43

Table of Contents

5 − Localizing Printer Drivers 45
Overview of the Localization Process 45
The Message Catalog File Format 45
The ppdpo Localization Utility 46
Using a Message Catalog with the PPD Compiler 47

6 − Doing Raster Color Management 49
Overview of Raster Color Management 49
RIP−Based Color Profiles 50
Driver−Based Color Profiles 51
RGB Color Profiles 51
CMYK Color Profiles 51
Dithering Tables 53

A − Software License Agreement 55
CUPS Driver Development Kit License Agreement 55
Introduction 55
Trademarks 56
Binary Distribution Rights 56
Support 56
GNU GENERAL PUBLIC LICENSE 57
Preamble 57

B − PPD Compiler Driver Information File Reference 63
Keyword Index 63
#define 64
#font 65
#include 66
#media 67
Attribute 68
Choice 69
ColorDevice 70
ColorModel 71
ColorProfile 72
Copyright 73
CustomMedia 74
Cutter 75
Darkness 76
DriverType 77
Duplex 78
Filter 79
Finishing 80
Font 81
Group 82
HWMargins 83
InputSlot 84
Installable 85
ManualCopies 85

Table of Contents

B − PPD Compiler Driver Information File Reference
Manufacturer....86
MaxSize....86
MediaSize....87
MediaType....88
MinSize....89
ModelName....89
ModelNumber....90
Option....91
PCFileName....92
Resolution....93
SimpleColorProfile....94
Throughput....95
UIConstraints....96
VariablePaperSize....97
Version....97
Standard Include Files....98
Printer Driver ModelNumber Constants....98
The CUPS Dymo Sample Driver (dymo)....98
The CUPS ESC/P Sample Driver (epson)....99
The DDK ESC/P Driver (escp)....100
The CUPS HP−PCL Sample Driver (hp)....101
The DDK HP−PCL Driver (pcl)....102
The CUPS HP−RTL Sample Driver....103
Color Keywords....104
Colorspace Keywords....104
Color Order Keywords....105

C − CUPS Driver API Reference....107
Header File....107
Library....107
Contents....107
Functions....108
cupsCMYKDelete()....109
cupsCMYKDoBlack()....110
cupsCMYKDoCMYK()....111
cupsCMYKDoGray()....112
cupsCMYKDoRGB()....113
cupsCMYKLoad()....114
cupsCMYKNew()....115
cupsCMYKSetBlack()....116
cupsCMYKSetCurve()....117
cupsCMYKSetGamma()....118
cupsCMYKSetInkLimit()....119
cupsCMYKSetLtDk()....120
cupsCheckBytes()....121
cupsCheckValue()....122
cupsDitherDelete()....123

Table of Contents

C − CUPS Driver API Reference
cupsDitherLine() 124
cupsDitherNew() 125
cupsFindAttr() 126
cupsLutDelete() 127
cupsLutLoad() 128
cupsLutNew() 129
cupsPackHorizontal() 130
cupsPackHorizontal2() 131
cupsPackHorizontalBit() 132
cupsPackVertical() 133
cupsRGBDelete() 134
cupsRGBDoGray() 135
cupsRGBDoRGB() 136
cupsRGBLoad() 137
cupsRGBNew() 138
Structures 139
cups_cmyk_s 140
cups_dither_s 141
cups_lut_s 142
cups_rgb_s 143
cups_sample_s 144
Types 145
cups_cmyk_t 146
cups_dither_t 147
cups_lut_t 148
cups_rgb_t 149
cups_sample_t 150
Variables 151
cups_scmy_lut[256] 152
cups_srgb_lut[256] 153

D − Man Pages 155
Man Page Index 155
cupsprofile(1) 156
ppdc(1) 157
ppdcfile(5) 158
ppdi(1) 159
ppdpo(1) 160

E − Release Notes 161
Changes in v1.0rc1 161
Changes in v0.20 161
Changes in v0.15 162
Changes in v0.14 162
Changes in v0.13 162
Changes in v0.12 162
Changes in v0.11 162

Table of Contents

E − Release Notes

Changes in v0.10.....163

Preface

This software users manual provides a step−by−step guide for using the CUPS™ Driver Development Kit ('DDK'), version 1.0.

Software Overview

The CUPS Driver Development Kit ('DDK') provides a suite of standard drivers, a PPD file compiler, and other utilities that can be used to develop printer drivers for CUPS and other printing environments. CUPS provides a portable printing layer for UNIX®−based operating systems. The CUPS DDK provides the means for mass−producing PPD files and drivers/filters for CUPS−based printer drivers.

The CUPS DDK is licensed under the GNU General Public License. Please contact Easy Software Products at the following address for commercial support and 'binary distribution' rights:

Attn: CUPS Licensing Information
Easy Software Products
44141 Airport View Drive, Suite 204
Hollywood, Maryland 20636−3142 USA

Voice: +1.301.373.9600
EMail: cups−info@cups.org

Free support is available by posting messages to the `cups.ddk` newsgroup at:

`http://www.cups.org/newsgroups.php`

The `cups.ddk` newsgroup is monitored by volunteers, so your message may go unanswered for days or weeks. Commercial support for the DDK is available for a fee; for more information, please visit the Easy Software Products web site at:

`http://www.easysw.com/cups/`

Document Overview

This software users manual is organized into the following sections:

- 1 − Building and Installing the CUPS DDK
- 2 − Getting Started with the CUPS DDK
- 3 − Developing Raster Printer Drivers
- 4 − Developing PostScript Printer Drivers
- 5 − Localizing Printer Drivers
- 6 − Doing Raster Color Management

- A − Software License Agreement
- B − PPD Compiler Driver Information File Reference
- C − CUPS Driver API Reference
- D − Man Pages
- E − Release Notes

Notation Conventions

Various font and syntax conventions are used in this guide. Examples and their meanings and uses are explained below:

Example	Description
`lpstat` `lpstat(1)`	The names of commands; the first mention of a command or function in a chapter is followed by a manual page section number.
/var */usr/share/cups/data/testprint.ps*	File and directory names.
`Request ID is Printer−123`	Screen output.
`lp −d printer filename ENTER`	Literal user input; special keys like **`ENTER`** are in ALL CAPS.
12.3	Numbers in the text are written using the period (.) to indicate the decimal point.

Abbreviations

The following abbreviations are used throughout this manual:

CMYK
: Cyan Magenta Yellow Black

DDK
: Device Driver Kit

ESC/P
: Epson Standard Code for Printers

Gb
: Gigabytes, or 1073741824 bytes

HP−PCL
: Hewlett Packard Page Control Language

K
: Black

kb
: Kilobytes, or 1024 bytes

Mb
: Megabytes, or 1048576 bytes

PJL
: Printer Job Language

PPD
: PostScript Printer Description

RGB
: Red Green Blue

Other References

Adobe PostScript Language Reference Manual, Third Edition
: The official reference manual for the PostScript language.

Adobe Technote #5003: PostScript Printer Description File Format Specification
: The official reference manual for the PPD file format.

CUPS Interface Design Description
: A programmer guide which describes the file formats and interfaces for the CUPS software.

CUPS Software Programmers Manual
: A programmer guide for interfacing with and/or extending the CUPS software.

Providing Feedback

Feedback and bug reports for this manual and the DDK as a whole are welcome. Please use the `cups.ddk` newsgroup to discuss potential problems and report all confirmed problems and documentation errors using the form at the following URL:

`http://www.cups.org/str.php`

1 – Building and Installing the CUPS DDK

This chapter describes how to build and install the CUPS DDK.

Before You Begin

You'll need ANSI−compliant C and C++ compilers, plus a make program and Bourne (POSIX) shell. The GNU compiler tools work well − we've tested the current DDK code against GCC 3.x with excellent results.

The makefiles used by the DDK should work with most versions of `make(1)`. We use them successfully with GNU make as well as the make programs shipped by SGI and Sun. *BSD, HP−UX, and Tru64 users should use GNU make (`gmake(1)`).

Building the DDK

The DDK is built by first configuring the software and then compiling it.

Configuring the Software

The DDK uses GNU autoconf, so you will find the usual `configure` script in the main DDK source directory. Type the following command to configure the DDK for your system using the default options:

```
./configure ENTER
```

The default options will put the DDK software under the */usr* directory on your system. Use the `−−prefix`

option to install the DDK software in another location:

```
./configure --prefix=/some/directory ENTER
```

If you will be distributing the DDK drivers to systems with different versions of CUPS installed on them, use the `--enable-static` option to link the DDK drivers against the static CUPS libraries:

```
./configure --enable-static ENTER
```

You can see all of the options supported by the `configure` script by using the `--help` option:

```
./configure --help ENTER
```

Compiling the Software

Once you have configured things, just type:

```
make ENTER
```

or:

```
gmake ENTER
```

to build the software. If you encounter any difficulties, please report them to the `cups.ddk` newsgroup on the CUPS web site at:

http://www.cups.org/newsgroups.php

Installing the DDK

Once you have built the software you need to install it before using it. The DDK supports installing via the makefile or using the EPM software.

Installing Using the Makefile

The `install` target provides a quick way to install the software on your local system:

```
make install ENTER
```

or:

```
gmake install ENTER
```

Similarly, the `uninstall` target removes the DDK from the local system:

```
make uninstall ENTER
```

or:

```
gmake uninstall ENTER
```

Installing Using EPM

The DDK also includes targets for building installable packages using the ESP Package Manager ("EPM") software, available separately at:

http://www.easysw.com/epm/

The EPM software creates binary packages that can be installed on other machines using the EPM list file "cupsddk.list". The top−level makefile included with the DDK provides targets for creating RPM (`rpm`), Debian (`deb`), and portable packages (`epm`) for installation:

```
make deb ENTER
make epm ENTER
make rpm ENTER
```

or

```
gmake deb ENTER
gmake epm ENTER
gmake rpm ENTER
```

After typing any of those commands, the corresponding software package file will be placed in the *dist* sub−directory. Use the corresponding commands to install the packages:

Debian:

```
dpkg −i cupsddk−1.0−linux−2.4−intel.deb ENTER
```

Portable:

```
./cupsddk.install ENTER
```

RPM:

```
rpm −i cupsddk−1.0−linux−2.4−intel.rpm ENTER
```

2 − Getting Started with the CUPS DDK

This chapter describes how to use the CUPS DDK and write PPD compiler source files.

The Basics

The DDK provides the basis for creating printer drivers that work within the architecture defined by the Common UNIX Printing System. The DDK includes a PostScript Printer Description ("PPD") file compiler, two general−purpose raster printer drivers for printers that understand the Hewlett−Packard Page Control Language ("HP−PCL") or Epson Standard Code for Printers ("ESC/P") languages, and a raster printer driver library that provides general−purpose dithering and color management/separation functions.

All of the tools in the DDK are currently command−line only, however future releases of the DDK will include an integrated development environment which provides an intuitive GUI for creating and testing printer drivers.

Aside from this manual, the DDK also includes several man pages that can be used as a quick reference when developing your printer drivers. The available man pages are:

- cupsprofile
- ppdc
- ppdcfile
- ppdi
- ppdpo
- rastertoescpx
- rastertopclx

Use the `man(1)` command to view these pages. For example, the following command:

```
man ppdc ENTER
```

displays the man page for the `ppdc(1)` command, which is the PPD compiler.

Using the PPD Compiler

The PPD compiler, `ppdc`, is a simple command−line tool that takes a single *driver information file*, which by convention uses the extension *.drv*, and produces one or more PPD files that may be distributed with your printer drivers for use with CUPS. For example, you would run the following command to create the English language PPD files defined by the driver information file *mydrivers.drv*:

```
ppdc mydrivers.drv ENTER
```

The PPD files are placed in a subdirectory called *ppd*. The `−d` option is used to put the PPD files in a different location, for example:

```
ppdc −d myppds mydrivers.drv ENTER
```

places the PPD files in a subdirectory named *myppds*. Finally, use the `−l` option to specify the language localization for the PPD files that are created, for example:

```
ppdc −d myppds/de −l de mydrivers.drv ENTER
ppdc −d myppds/en −l en mydrivers.drv ENTER
ppdc −d myppds/es −l es mydrivers.drv ENTER
ppdc −d myppds/fr −l fr mydrivers.drv ENTER
ppdc −d myppds/it −l it mydrivers.drv ENTER
```

creates PPD files in German (de), English (en), Spanish (es), French (fr), and Italian (it) in the corresponding subdirectories. You'll learn more about localization in "Chapter 5, Localizing Printer Drivers".

Driver Information Files

The driver information files accepted by the PPD compiler are plain text files that define the various attributes and options that are included in the PPD files that are generated. A driver information file can define the information for one or more printers and their corresponding PPD files.

A Simple Example

The example in Listing 2−1 shows a driver information file which defines the minimum required attributes to provide a valid PPD file.

The first part of the file includes standard definition files for fonts and media sizes:

```
#include <font.defs>
#include <media.defs>
```

```
// Include standard font and media definitions
#include <font.defs>
#include <media.defs>

// List the fonts that are supported, in this case all standard
// fonts...
Font *

// Manufacturer, model name, and version
Manufacturer "Foo"
ModelName "FooJet 2000"
Version 1.0

// Each filter provided by the driver...
Filter application/vnd.cups-raster 100 rastertofoo

// Supported page sizes
*MediaSize Letter
MediaSize A4

// Supported resolutions
*Resolution k 8 0 0 0 "600dpi/600 DPI"

// Specify the name of the PPD file we want to generate...
PCFileName "foojet2k.ppd"
```

Listing 2−1, "examples/minimum.drv":

The `#include` directive works just like the C/C++ include directive; files included using the angle brackets (`<filename>`) are found in any of the standard include directories and files included using quotes (`"filename"`) are found in the same directory as the source or include file. The `<font.defs>` include file defines the standard fonts which are included with ESP Ghostscript and the Apple PDF RIP. The `<media.defs>` include file defines the standard media sizes listed in Appendix B of the Adobe PostScript Printer Description File Format Specification.

Other standard include files include:

- `<escp.h>` − Defines all of the ESC/P driver constants.
- `<pcl.h>` − Defines all of the HP−PCL driver constants.
- `<raster.defs>` − Defines all of the CUPS raster format constants.

Next we list all of the fonts that are available in the driver; for CUPS raster drivers, the following line is all that is usually supplied:

```
Font *
```

The `Font` directive specifies the name of a single font or the asterisk to specify all fonts. For example, you would use the following line to define an additional bar code font that you are supplying with your printer driver:

```
//   name         encoding  version  charset  status
Font Barcode-Foo  Special   "(1.0)"  Special  ROM
```

The name of the font is `Barcode−Foo`. Since it is not a standard text font, the encoding and charset name `Special` is used. The version number is `1.0` and the status (where the font is located) is `ROM` to indicate that the

font does not need to be embedded in documents that use the font for this printer.

Third comes the manufacturer, model name, and version number information strings:

```
Manufacturer "Foo"
ModelName "FooJet 2000"
Version 1.0
```

These strings are used when the user (or auto−configuration program) selects the printer driver for a newly connected device.

The list of filters comes after the information strings; for the example in Listing 2−1, we have a single filter that takes CUPS raster data:

```
Filter application/vnd.cups−raster 100 rastertofoo
```

Each filter specified in the driver information file is the equivalent of a printer driver for that format; if a user submits a print job in a different format, CUPS figures out the sequence of commands that will produce a supported format for the least relative cost.

Once we have defined the driver information we specify the supported options. For the example driver we support a single resolution of 600 dots per inch and two media sizes, A4 and Letter:

```
*MediaSize Letter
MediaSize A4

*Resolution k 8 0 0 0 "600dpi/600 DPI"
```

The asterisk in front of the `MediaSize` and `Resolution` directives specify that those option choices are the default. The `MediaSize` directive is followed by a media size name which is normally defined in the `<media.defs>` file and corresponds to a standard Adobe media size name. If the default media size is `Letter`, the PPD compiler will override it to be `A4` for non−English localizations for you automatically.

The `Resolution` directive accepts several values after it as follows:

1. Colorspace for this resolution, if any. In the example file, the colorspace `k` is used which corresponds to black. For printer drivers that support color printing, this field is usually specified as "−" for "no change".
2. Bits per color. In the example file, we define 8 bits per color, for a continuous−tone grayscale output. Supported values include 1, 2, 4, and 8. Future versions of CUPS may also support 16 bits per color.
3. Rows per band. In the example file, we define 0 rows per band to indicate that our printer driver does not process the page in bands.
4. Row feed. In the example, we define the feed value to be 0 to indicate that our printer driver does not interleave the output.
5. Row step. In the example, we define the step value to be 0 to indicate that our printer driver does not interleave the output. This value normally indicates the spacing between the nozzles of an inkjet printer − when combined with the previous two values, it informs the driver how to stagger the output on the page to produce interleaved lines on the page for higher−resolution output.
6. Choice name and text. In the example, we define the choice name and text to be `"600dpi/600 DPI"`. The name and text are separated by slash (`/`) character; if no text is specified, then the name is used as the text. The PPD compiler parses the name to determine the actual resolution; the name can be of the form *`RESOLUTION`*`dpi` for resolutions that are equal horizontally and vertically or *`HRES`*`x`*`VRES`*`dpi` for isometric resolutions. Only integer resolution values are supported, so a resolution name of `300dpi` is

valid while `300.1dpi` is not.

Finally, the `PCFileName` directive specifies that the named PPD file should be written for the current driver definitions:

```
PCFileName "foojet2k.ppd"
```

The filename follows the directive and *must* conform to the Adobe filename requirements in the Adobe Postscript Printer Description File Format Specification. Specifically, the filename may not exceed 8 characters followed by the extension *.ppd*.

Grouping and Inheritance

The previous example created a single PPD file. Driver information files can also define multiple printers by using the PPD compiler grouping functionality. Directives are grouped using the curly braces (`{` and `}`) and every group that uses the `PCFileName` directive produces a PPD file with that name. Listing 2−2 shows a variation of the original example that uses two groups to define two printers that share the same printer driver filter but provide two different resolution options.

The second example is essentially the same as the first, except that each printer model is defined inside of a pair of curly braces. For example, the first printer is defined using:

```
{
  // Supported resolutions
  *Resolution k 8 0 0 0 "600dpi/600 DPI"

  // Specify the model name and filename...
  ModelName "FooJet 2000"
  PCFileName "foojet2k.ppd"
}
```

The printer *inherits* all of the definitions from the parent group (the top part of the file) and adds the additional definitions inside the curly braces for that printer driver. When we define the second group, it also inherits the same definitions from the parent group but *none* of the definitions from the first driver. Groups can be nested to any number of levels to support variations of similar models without duplication of information.

Defining Constants

Sometimes you will want to define constants for your drivers so that you can share values in different groups within the same driver information file, or to share values between different driver information files using the `#include` directive. The `#define` directive is used to define constants for use in your printer definitions:

```
#define NAME value
```

The `NAME` is any sequence of letters, numbers, and the underscore. The `value` is a number or string; if the value contains spaces you must put double quotes around it, for example:

```
#define FOO "My String Value"
```

```
// Include standard font and media definitions
#include <font.defs>
#include <media.defs>

// List the fonts that are supported, in this case all standard
// fonts...
Font *

// Manufacturer and version
Manufacturer "Foo"
Version 1.0

// Each filter provided by the driver...
Filter application/vnd.cups-raster 100 rastertofoo

// Supported page sizes
*MediaSize Letter
MediaSize A4

{
  // Supported resolutions
  *Resolution k 8 0 0 0 "600dpi/600 DPI"

  // Specify the model name and filename...
  ModelName "FooJet 2000"
  PCFileName "foojet2k.ppd"
}

{
  // Supported resolutions
  *Resolution k 8 0 0 0 "1200dpi/1200 DPI"

  // Specify the model name and filename...
  ModelName "FooJet 2001"
  PCFileName "foojt2k1.ppd"
}
```

Listing 2−2, "examples/grouping.drv":

Once defined, you use the notation $NAME to substitute the value of the constant in the file, for example:

```
#define MANUFACTURER "Foo"
#define FOO_600      0
#define FOO_1200     1

{
  Manufacturer "$MANUFACTURER"
  ModelNumber $FOO_600
  ModelName "FooJet 2000"
  ...
}

{
  Manufacturer "$MANUFACTURER"
  ModelNumber $FOO_1200
  ModelName "FooJet 2001"
  ...
}
```

Numeric constants can be bitwise OR'd together by placing the constants inside parenthesis, for example:

```
// ModelNumber capability bits
#define DUPLEX 1
#define COLOR  2

...

{
  // Define a model number specifying the capabilities of
  // the printer...
  ModelNumber ($DUPLEX $COLOR)
  ...
}
```

Defining Color Support

For printer drivers that support color printing, the `ColorDevice` and `ColorModel` directives must be used to tell the printing system that color output is desired and in what formats. Listing 2−3 shows a variation of the previous example which includes a color printer that supports printing at 300 and 600 DPI.

The key changes are the addition of the `ColorDevice` directive:

```
ColorDevice true
```

which tells the printing system that the printer supports color printing, and the `ColorModel` directives:

```
ColorModel Gray/Grayscale w chunky 0
*ColorModel RGB/Color rgb chunky 0
```

which tell the printing system which colorspaces are supported by the printer driver for color printing. Each of the `ColorModel` directives is followed by the option name and text (`Gray/Grayscale` and `RGB/Color`), the colorspace name (`w` and `rgb`), the color organization (`chunky`), and the compression mode number (`0`) to be passed to the driver. The option name can be any of the standard Adobe `ColorModel` names:

- `Gray` − Grayscale output.
- `RGB` − Color output, typically using the RGB colorspace, but without a separate black channel.
- `CMYK` − Color output with a separate black channel.

Custom names can be used, however it is recommended that you use your vendor prefix for any custom names, for example "fooName".

The colorspace name can be any of the following universally supported colorspaces:

- `w` − Luminance
- `rgb` − Red, green, blue
- `k` − Black
- `cmy` − Cyan, magenta, yellow
- `cmyk` − Cyan, magenta, yellow, black
- `ciexyz` − CIE XYZ
- `cielab` − CIE Lab

```
// Include standard font and media definitions
#include <font.defs>
#include <media.defs>

// List the fonts that are supported, in this case all standard
// fonts...
Font *

// Manufacturer and version
Manufacturer "Foo"
Version 1.0

// Each filter provided by the driver...
Filter application/vnd.cups-raster 100 rastertofoo

// Supported page sizes
*MediaSize Letter
MediaSize A4

{
  // Supported resolutions
  *Resolution k 8 0 0 0 "600dpi/600 DPI"

  // Specify the model name and filename...
  ModelName "FooJet 2000"
  PCFileName "foojet2k.ppd"
}

{
  // Supports color printing
  ColorDevice true

  // Supported colorspaces
  ColorModel Gray/Grayscale w chunky 0
  *ColorModel RGB/Color rgb chunky 0

  // Supported resolutions
  *Resolution - 8 0 0 0 "300dpi/300 DPI"
  Resolution - 8 0 0 0 "600dpi/600 DPI"

  // Specify the model name and filename...
  ModelName "FooJet Color"
  PCFileName "foojetco.ppd"
}
```

Listing 2-3, "examples/color.drv":

Additional colorspaces are supported by the standard CUPS image RIP filter and by ESP Ghostscript. The full list can be found in Appendix B, PPD Compiler Source File Reference.

The color organization can be any of the following values:

- `chunky` - Color values are passed together on a line as RGB RGB RGB RGB
- `banded` - Color values are passed separately on a line as RRRR GGGG BBBB; not supported by the Apple RIP filters
- `planar` - Color values are passed separately on a page as RRRR RRRR RRRR ... GGGG GGGG GGGG ... BBBB BBBB BBBB; not supported by the Apple RIP filters

The compression mode value is passed to the driver in the `cupsCompression` attribute. It is traditionally used to select an appropriate compression mode for the color model but can be used for any purpose, such as specifying a photo mode vs. standard mode.

Defining Custom Options and Option Groups

The `Group`, `Option`, and `Choice` directives are used to define or select a group, option, or choice. Listing 2−4 shows a variation of the first example that provides two custom options in a group named "Footasm".

The custom group is introduced by the `Group` directive which is followed by the name and optionally text for the user:

```
Group "Footasm"
```

The group name must conform to the PPD specification and cannot exceed 40 characters in length. If you specify user text, it cannot exceed 80 characters in length. The groups `General`, `Extra`, and `InstallableOptions` are predefined by CUPS; the general and extra groups are filled by the UI options defined by the PPD specification. The `InstallableOptions` group is reserved for options that define whether accessories for the printer (duplexer unit, finisher, stapler, etc.) are installed.

Once the group is specified, the `Option` directive is used to introduce a new option:

```
Option "fooEnhance/Resolution Enhancement" Boolean AnySetup 10
```

The directive is followed by the name of the option and any optional user text, the option type, the PostScript document group, and the sort order number. The option name must conform to the PPD specification and cannot exceed 40 characters in length. If you specify user text, it cannot exceed 80 characters in length.

The option type can be `Boolean` for true/false selections, `PickOne` for picking one of many choices, or `PickMany` for picking zero or more choices. Boolean options can have at most two choices with the names `False` and `True`. Pick options can have any number of choices, although for Windows compatibility reasons the number of choices should not exceed 255.

The PostScript document group is typically `AnySetup`, meaning that the option can be introduced at any point in the PostScript document. Other values include `PageSetup` to include the option before each page and `DocumentSetup` to include the option once at the beginning of the document.

The sort order number is used to sort the printer commands associated with each option choice within the PostScript document. This allows you to setup certain options before others as required by the printer. For most CUPS raster printer drivers, the value `10` can be used for all options.

Once the option is specified, each option choice can be listed using the `Choice` directive:

```
*Choice True/Yes "<</cupsCompression 1>>setpagedevice"
Choice False/No "<</cupsCompression 0>>setpagedevice"
```

The directive is followed by the choice name and optionally user text, and the PostScript commands that should be inserted when printing a file to this printer. The option name must conform to the PPD specification and cannot exceed 40 characters in length. If you specify user text, it cannot exceed 80 characters in length.

```
// Include standard font and media definitions
#include <font.defs>
#include <media.defs>

// List the fonts that are supported, in this case all standard
// fonts...
Font *

// Manufacturer, model name, and version
Manufacturer "Foo"
ModelName "FooJet 2000"
Version 1.0

// Each filter provided by the driver...
Filter application/vnd.cups-raster 100 rastertofoo

// Supported page sizes
*MediaSize Letter
MediaSize A4

// Supported resolutions
*Resolution k 8 0 0 0 "600dpi/600 DPI"

// Option Group
Group "Footasm"

  // Boolean option
  Option "fooEnhance/Resolution Enhancement" Boolean AnySetup 10
    *Choice True/Yes "<</cupsCompression 1>>setpagedevice"
    Choice False/No "<</cupsCompression 0>>setpagedevice"

  // Multiple choice option
  Option "fooOutputType/Output Quality" PickOne AnySetup 10
    *Choice "Auto/Automatic Selection" "<</OutputType(Auto)>>setpagedevice""
    Choice "Text/Optimize for Text" "<</OutputType(Text)>>setpagedevice""
    Choice "Graph/Optimize for Graphics" "<</OutputType(Graph)>>setpagedevice""
    Choice "Photo/Optimize for Photos" "<</OutputType(Photo)>>setpagedevice""

// Specify the name of the PPD file we want to generate...
PCFileName "foojet2k.ppd"
```

Listing 2−4, "examples/custom.drv":

The PostScript commands are also interpreted by any RIP filters, so these commands typically must be present for all option choices. Most commands take the form:

```
<</name value>>setpagedevice
```

where `name` is the name of the PostScript page device attribute and `value` is the numeric or string value for that attribute.

Defining Constraints

Constraints are strings that are used to specify that one or more option choices are incompatible, for example printing double−sided on transparency media. Constraints are also used to prevent the use of uninstalled features such as the duplexer unit, additional media trays, and so forth.

```
// Include standard font and media definitions
#include <font.defs>
#include <media.defs>

// List the fonts that are supported, in this case all standard
// fonts...
Font *

// Manufacturer, model name, and version
Manufacturer "Foo"
ModelName "FooJet 2000"
Version 1.0

// Each filter provided by the driver...
Filter application/vnd.cups-raster 100 rastertofoo

// Supported page sizes
*MediaSize Letter
MediaSize A4

// Supported resolutions
*Resolution k 8 0 0 0 "600dpi/600 DPI"

// Installable Option Group
Group "InstallableOptions/Options Installed"

  // Duplexing unit option
  Option "Option1/Duplexing Unit" Boolean AnySetup 10
    Choice True/Installed ""
    *Choice "False/Not Installed" ""

// General Option Group
Group General

  // Duplexing option
  Option "Duplex/Two-Sided Printing" PickOne AnySetup 10
    *Choice "None/No" "<</Duplex false>>setpagedevice""
    Choice "DuplexNoTumble/Long Edge Binding"
           "<</Duplex true/Tumble false>>setpagedevice""
    Choice "DuplexTumble/Short Edge Binding"
           "<</Duplex true/Tumble true>>setpagedevice""

// Only allow duplexing if the duplexer is installed
UIConstraint "*Duplex *Option1 False"

// Specify the name of the PPD file we want to generate...
PCFileName "foojet2k.ppd"
```

Listing 2−5, "examples/constraint.drv":

The `UIConstraint` directive is used to specify a constraint that is placed in the PPD file. The directive is followed by a string using one of the following formats:

```
UIConstraint "*Option1 *Option2"
UIConstraint "*Option1 Choice1 *Option2"
UIConstraint "*Option1 *Option2 Choice2"
UIConstraint "*Option1 Choice1 *Option2 Choice2"
```

Each option name is preceded by the asterisk (*). If no choice is given for an option, then all choices *except* `False` and `None` will conflict with the other option and choice(s). Since the PPD compiler automatically adds reciprocal constraints (option A conflicts with option B, so therefore option B conflicts with option A), you need only specify the constraint once.

Listing 2−5 shows a variation of the first example with an added `Duplex` option and installable option for the duplexer, `Option1`. A constraint is added at the end to specify that any choice of the `Duplex` option that is not `None` is incompatible with the "Duplexer Installed" option set to "Not Installed" (`False`):

```
UIConstraint "*Duplex *Option1 False"
```

Importing Existing PPD Files

The DDK includes a utility called `ppdi(1)` which allows you to import existing PPD files into the driver information file format. Once imported, you can modify, localize, and regenerate the PPD files easily. The PPD files can be for CUPS raster printer drivers or for PostScript printers − the DDK makes no distinction when managing driver information or PPD files.

Type the following command to import the PPD file *mydevice.ppd* into the driver information file *mydevice.drv*:

```
ppdi −o mydevice.drv mydevice.ppd ENTER
```

If you have a whole directory of PPD files that you would like to import, you can list multiple filenames or use shell wildcards to import more than one PPD file on the command−line:

```
ppdi −o mydevice.drv mydevice1.ppd mydevice2.ppd ENTER
ppdi −o mydevice.drv *.ppd ENTER
```

If the driver information file already exists, the new PPD file entries are appended to the end of the file. Each PPD file is placed in its own group of curly braces within the driver information file.

3 – Developing Raster Printer Drivers

This chapter describes how to develop PPD files for the included DDK raster printer drivers.

The DDK Drivers

The DDK includes two general−purpose raster printer drivers that support the HP−PCL and ESC/P languages. Driver information files based upon these drivers can use a driver−specific include file and the `DriverType` directive.

Both drivers offer color management and dithering capabilities which are described in detail in Chapter 6, Doing Raster Color Management.

The HP−PCL Driver

The HP−PCL driver includes a printer command filter called `commandtopclx` and a raster printer driver filter called `rastertopclx`. The command filter supports head cleaning, printing a self−test page, and ink cartridge alignment.

The raster printer driver filter accepts grayscale, RGB, and CMYK raster data for printing to laser and inkjet devices. It supports PJL commands for device−specific features and uses PCL 3/4/5 raster graphics commands for all laser and some older inkjet printers or the various PCL3GUI and HP−RTL variants that are used by most of the inkjet printers sold by HP. The driver does *not* support native text rendering due to the limitations of font support in most PCL implementations.

constant	description
PCL_PAPER_SIZE	Use paper size command (ESC & l # A)
PCL_INKJET	Use inkjet commands
PCL_RASTER_END_COLOR	Use new end−raster command (ESC * r C)
PCL_RASTER_CID	Use configure−image−data command (ESC * v # W)
PCL_RASTER_CRD	Use configure−raster−data command (ESC * g # W)
PCL_RASTER_SIMPLE	Use simple−raster−color command (ESC * r # U)
PCL_RASTER_RGB24	Use 24−bit RGB mode
PCL_PJL	Use PJL commands
PCL_PJL_PAPERWIDTH	Use PJL PAPERWIDTH/LENGTH commands
PCL_PJL_HPGL2	Use PJL ENTER HPGL2 command
PCL_PJL_PCL3GUI	Use PJL ENTER PCL3GUI command
PCL_PJL_RESOLUTION	Use PJL SET RESOLUTION command

Table 3−1, HP−PCL ModelNumber constants

The HP−PCL driver also provides an include file, <pcl.h>. Driver information files that use the HP−PCL driver begin with the following:

```
#include <font.defs>
#include <media.defs>
#include <raster.defs>
#include <pcl.h>

DriverType pcl
```

ModelNumber Constants

Table 3−1 shows the constants that are defined in the <pcl.h> include file. These constants are used with the ModelNumber directive to control the behavior of the driver. For example, a typical PCL laser printer would use the following ModelNumber specification:

```
ModelNumber ($PCL_PAPER_SIZE $PCL_PJL $PCL_PJL_RESOLUTION)
```

The parenthesis around the PCL constants tell the PPD compiler to compute the bitwise OR of each of the values. The HP−PCL driver will then use this information to tailor the output of the driver for the printer, in this case to use Printer Job Language ("PJL") commands to setup the job, including the PJL resolution command, and to use the PCL 3 paper size command. Table 3−2 shows the constants to use for several common types of HP printers.

Printer Model	ModelNumber Value
HP Color LaserJet Series	($PCL_PAPER_SIZE $PCL_RASTER_END_COLOR $PCL_RASTER_CID $PCL_RASTER_SIMPLE $PCL_RASTER_RGB24 $PCL_PJL $PCL_PJL_RESOLUTION)
HP DesignJet, Desktop	($PCL_PAPER_SIZE $PCL_RASTER_END_COLOR $PCL_RASTER_CRD $PCL_PJL $PCL_PJL_PCL3GUI)
HP DesignJet, High−End	($PCL_PAPER_SIZE $PCL_RASTER_END_COLOR $PCL_RASTER_CID $PCL_RASTER_SIMPLE $PCL_RASTER_RGB24 $PCL_PJL $PCL_PJL_PAPERWIDTH $PCL_PJL_HPGL2 $PCL_PJL_RESOLUTION)
HP DesignJet, Mid−Range	($PCL_PAPER_SIZE $PCL_RASTER_END_COLOR $PCL_RASTER_CRD $PCL_RASTER_SIMPLE $PCL_RASTER_RGB24 $PCL_PJL $PCL_PJL_PAPERWIDTH $PCL_PJL_PCL3GUI $PCL_PJL_RESOLUTION)
HP DeskJet Series	($PCL_PAPER_SIZE $PCL_RASTER_END_COLOR $PCL_RASTER_CRD $PCL_PJL $PCL_PJL_PCL3GUI)
HP LaserJet Series	($PCL_PAPER_SIZE $PCL_PJL $PCL_PJL_RESOLUTION)
HP OfficeJet Series	($PCL_PAPER_SIZE $PCL_RASTER_END_COLOR $PCL_RASTER_CRD $PCL_PJL $PCL_PJL_PCL3GUI)
HP Photosmart Series	($PCL_PAPER_SIZE $PCL_RASTER_END_COLOR $PCL_RASTER_CRD $PCL_PJL $PCL_PJL_PCL3GUI)

Table 3−2, ModelNumber values for common HP printers

Writing a Basic HP LaserJet Driver

Now that we have covered the HP−PCL driver definitions, we will create a driver for the HP LaserJet 2100, 2200, and 2300 series printers which supports all of the features that are available via PCL 5. While these printers also support PostScript, the interpreters have several known problems with TrueType fonts which can only be bypassed by using the PCL 5 printing path.

All three models support printing at 300 and 600 DPI through PCL 5 graphics; the 1200 DPI resolution is only available through PostScript and PCL 6. Each printer has an optional high−capacity paper tray and the 2200 and 2300 series printers also offer an optional duplexing unit. Listing 3−1 shows the driver information file for a basic HP LaserJet driver which supports the three models.

The file starts with the usual `#include` directives and then sets the driver type and model number so that we use the HP−PCL driver with the output tailored to a HP LaserJet printer:

```
// Specify that this driver uses the HP-PCL driver...
DriverType pcl

// Specify the driver options via the model number...
ModelNumber ($PCL_PAPER_SIZE $PCL_PJL $PCL_PJL_RESOLUTION)
```

```
// Include standard font and media definitions
#include <font.defs>
#include <media.defs>

// Include HP-PCL driver definitions
#include <pcl.h>

// Specify that this driver uses the HP-PCL driver...
DriverType pcl

// Specify the driver options via the model number...
ModelNumber ($PCL_PAPER_SIZE $PCL_PJL $PCL_PJL_RESOLUTION)

// List the fonts that are supported, in this case all standard
// fonts...
Font *

// Manufacturer and driver version
Manufacturer "HP"
Version 1.0

// Supported page sizes and their margins
HWMargins 18 12 18 12
*MediaSize Letter
MediaSize Legal
MediaSize Executive
MediaSize Monarch
MediaSize Statement
MediaSize FanFoldGermanLegal

HWMargins 18 12.72 18 12.72
MediaSize Env10

HWMargins 9.72 12 9.72 12
MediaSize A4
MediaSize A5
MediaSize B5
MediaSize EnvC5
MediaSize EnvDL
MediaSize EnvISOB5
MediaSize Postcard
MediaSize DoublePostcard

// Only black-and-white output with mode 3 compression...
ColorModel Gray k chunky 3

// Supported resolutions
Resolution - 1 0 0 0 "300dpi/300 DPI"
*Resolution - 8 0 0 0 "600dpi/600 DPI"

// Supported input slots
*InputSlot 7 "Auto/Automatic Selection"
InputSlot 2 "Manual/Tray 1 - Manual Feed"
InputSlot 4 "Upper/Tray 1"
InputSlot 1 "Lower/Tray 2"
InputSlot 5 "LargeCapacity/Tray 3"
```

Listing 3-1, "examples/laserjet-basic.drv"

```
// Tray 3 is an option...
Installable "OptionLargeCapacity/Tray 3 Installed"
UIConstraints "*OptionLargeCapacity False *InputSlot LargeCapacity"

{
  // HP LaserJet 2100 Series
  Throughput 10
  ModelName "LaserJet 2100 Series"
  PCFileName "hpljt211.ppd"
}

{
  // LaserJet 2200 and 2300 series have duplexer option...
  Duplex normal
  Installable "OptionDuplex/Duplexer Installed"
  UIConstraints "*OptionDuplex False *Duplex"

  {
    // HP LaserJet 2200 Series
    Throughput 19
    ModelName "LaserJet 2200 Series"
    PCFileName "hpljt221.ppd"
  }

  {
    // HP LaserJet 2300 Series
    Throughput 25
    ModelName "LaserJet 2300 Series"
    PCFileName "hpljt231.ppd"
  }
}
```

Listing 3−1, "examples/laserjet−basic.drv", continued...

Then we list all of the media sizes that are supported by the printers along with the margins that should be used:

```
HWMargins 18 12 18 12
*MediaSize Letter
MediaSize Legal
MediaSize Executive
MediaSize Monarch
MediaSize Statement
MediaSize FanFoldGermanLegal

HWMargins 18 12.72 18 12.72
MediaSize Env10

HWMargins 9.72 12 9.72 12
MediaSize A4
MediaSize A5
MediaSize B5
MediaSize EnvC5
MediaSize EnvDL
MediaSize EnvISOB5
MediaSize Postcard
MediaSize DoublePostcard
```

Next we use the `ColorModel` directive to specify that our driver only prints grayscale output using the black colorspace and PCL mode 3 raster compression:

```
ColorModel Gray k chunky 3
```

These printers support printing at 300 and 600 DPI through HP−PCL 5, so we list those resolutions using the `Resolution` directive. We use 1 bit per color for 300 DPI to provide fast printing and 8 bits per color for 600 DPI to provide the highest quality:

```
Resolution − 1 0 0 0 "300dpi/300 DPI"
*Resolution − 8 0 0 0 "600dpi/600 DPI"
```

All of the models provide two standard paper trays and one optional tray. The first tray is used as both the multi−purpose and manual feed tray and gets listed twice. We also provide an "auto" tray which tells the printer to grab media from the first available location:

```
*InputSlot 7 "Auto/Automatic Selection"
InputSlot 2 "Manual/Tray 1 − Manual Feed"
InputSlot 4 "Upper/Tray 1"
InputSlot 1 "Lower/Tray 2"
InputSlot 5 "LargeCapacity/Tray 3"
```

The numbers we used for the input slots are the PCL values associated with each tray; consult the HP−PCL Language Reference Manual for a complete list of possible values on all printers.

Since the third tray is an optional accessory, we list an installable option along with a constraint so that users may only select the third tray if it has been installed:

```
Installable "OptionLargeCapacity/Tray 3 Installed"
UIConstraints "*OptionLargeCapacity False *InputSlot LargeCapacity"
```

By convention, installable options usually begin with the prefix `Option`. Some vendors number the options, e.g. `Option1`, `Option2`, etc., however we have opted to use a more readable name, `OptionLargeCapacity`. Using a textual name also allows you to add additional installable options without renumbering existing options, and makes it easier to validate constraints.

The HP LaserJet 2200 and 2300 series printers also have an optional duplexer, which is listed using the `Duplex` directive along with another installable option and constraint:

```
Duplex normal
Installable "OptionDuplex/Duplexer Installed"
UIConstraints "*OptionDuplex False *Duplex"
```

Again, we are using the more readable installable option name `OptionDuplex` instead of `Option2`.

We finish things up by using grouping to isolate the three printer models and provide unique values for the `Throughput`, `ModelName`, and `PCFileName` directives. Notice how we are able to group the duplex option and then share the definition with the 2200 and 2300 drivers:

```
{
  // HP LaserJet 2100 Series
  Throughput 10
  ModelName "LaserJet 2100 Series"
  PCFileName "hpljt211.ppd"
}

{
  // LaserJet 2200 and 2300 series have duplexer option...
  Duplex normal
  Installable "OptionDuplex/Duplexer Installed"
  UIConstraints "*OptionDuplex False *Duplex"

  {
    // HP LaserJet 2200 Series
    Throughput 19
    ModelName "LaserJet 2200 Series"
    PCFileName "hpljt221.ppd"
  }

  {
    // HP LaserJet 2300 Series
    Throughput 25
    ModelName "LaserJet 2300 Series"
    PCFileName "hpljt231.ppd"
  }
}
```

To test the new drivers, start by running the `ppdc` program to create the PPD files:

```
ppdc laserjet-basic.drv ENTER
```

Then use the `lpadmin(8)` command to add the printer with the correct device URI. The following example adds a HP LaserJet 2100 which is connected via a JetDirect interface:

```
lpadmin -p lj2100 -E -v socket://lj2100 -i ppd/hpljt211.ppd ENTER
```

Finally, print a test page to see it work:

```
lp -d lj2100 /usr/share/cups/data/testprint.ps ENTER
```

PJL Attributes

The HP-PCL driver also supports extensive Printer Job Language (PJL) commands through a combination of PPD attributes and options. Table 3-3 lists the PJL attributes that are supported along with the PPD options they map to. PJL attributes are specified using the `Attribute` directive using the `cupsPJL` keyword. For example, the following attribute provides the PJL commands to enable or disable the resolution enhancement features of the printer:

```
Attribute cupsPJL cupsRET "@PJL SET SMOOTHING=%?False:OFF;%?True:ON;%n"
```

The directive, `Attribute`, is followed by the attibute keyword, `cupsPJL`, the attribute name, `cupsRET`, and the attribute value, in this case a PJL command string. The command string consists of PJL command text and special substitution fields starting with the percent (%) character.

In the example above, the `%?` substitution conditionally inserts some text if the string matches the option value, typically the name of the choice. In this case, we have two conditional substitutions. The first inserts the text `OFF` if the `cupsRET` option is `False`, and the second inserts the text `ON` if the option is `True`. The syntax is as follows:

```
%?look for:insert;
```

Multiple conditional substitutions can be listed up to about 230 characters − the PPD file format imposed a 255 character line length limit, and attribute values cannot span multiple lines.

Aside from conditional substitutions, the HP−PCL driver supports the following additional substitutions. Unknown substitutions are inserted verbatim:

- `%%` − inserts the percent character (%).
- `%b` − inserts the job−billing value for this job.
- `%h` − inserts the job−originating−hostname value for this job.
- `%j` − inserts the job−id value for this job.
- `%n` − inserts the carriage return (ASCII CR or `0D` hex) and linefeed (ASCII LF or `0A` hex) characters.
- `%q` − inserts the double quote character (").
- `%s` − inserts the current option value or choice.
- `%t` − inserts the name/title of this job.
- `%u` − inserts the job−originating−username value for this job.

Adding PJL Options to the Basic LaserJet Driver

All three HP LaserJet models support additional options via PJL commands. Listing 3−2 shows a modified version of the driver which adds support the resolution enhancement and toner saving features of the printers.

The new resolution enhancement option consists of a PPD attribute containing the `cupsRET` command followed by the `cupsRET` option. Since this option applies to the entire job, the option is placed in the `DocumentSetup` section:

```
Attribute cupsPJL cupsRET "@PJL SET SMOOTHING=%?False:OFF;%?True:ON;%n"

Option "cupsRET/Smoothing" Boolean DocumentSetup 10
  Choice "False/Off" ""
  *Choice "True/On" ""
```

The toner saving option is added the same way using the `cupsTonerSave` attribute and option:

```
Attribute cupsPJL cupsTonerSave "@PJL SET ECONOMODE=%?False:OFF;%?True:ON;%n"

Option "cupsTonerSave/Save Toner" Boolean DocumentSetup 10
  *Choice "False/No" ""
  Choice "True/Yes" ""
```

Since we didn't specify a group for these options, they will be put in the `General` option group.

Attribute	Option	Description
COLORSPACE.ColorModel	Varies	Specifies the colorspace to set at the beginning of the job.
cupsBooklet	cupsBooklet	Specifies the PJL commands to send for setting the booklet printing mode.
cupsPunch	cupsPunch	Specifies the PJL commands to send for setting the punch mode.
cupsRET	cupsRET	Specifies the PJL commands to set the resolution enhancement mode.
cupsStaple	cupsStaple	Specifies the PJL commands to send for setting the stapler mode.
cupsTonerSave	cupsTonerSave	Specifies the PJL commands to set the toner saving mode.
Duplex	Duplex	Specifies the PJL commands to send for setting the duplex mode.
EndJob	N/A	Specifies the PJL commands to send at the end of a job.
Jog	Varies	Specifies the PJL commands to send for setting the output jogging.
MediaClass	Varies	Specifies the PJL commands to send for setting the media class.
MediaColor	Varies	Specifies the PJL commands to send for setting the media color.
MediaType	MediaType	Specifies the PJL commands to send for setting the media type.
OutputType	Varies	Specifies the PJL commands to send for setting the output type.
RENDERINTENT.ColorModel	Varies	Specifies the rendering intent to set at the beginning of the job.
RENDERMODE.ColorModel	Varies	Specifies the render mode to set at the beginning of the job.
StartJob	N/A	Specifies the PJL commands to send at the beginning of a job.
Tumble	Duplex	Specifies the PJL commands to send for setting the duplex tumble mode.

Table 3−3, PJL attributes and options

To test the new drivers, start by running the `ppdc` program to create the PPD files:

```
ppdc laserjet-pjl.drv ENTER
```

Then use the `lpadmin(8)` command to add the printer with the correct device URI. The following example adds a HP LaserJet 2100 which is connected via a JetDirect interface:

```
lpadmin -p lj2100 -E -v socket://lj2100 -i ppd/hpljt212.ppd ENTER
```

Finally, print a test page to see it work:

```
lp -d lj2100 -o cupsTonerSave=True /usr/share/cups/data/testprint.ps ENTER
```

```
// Include standard font and media definitions
#include <font.defs>
#include <media.defs>

// Include HP-PCL driver definitions
#include <pcl.h>

// Specify that this driver uses the HP-PCL driver...
DriverType pcl

// Specify the driver options via the model number...
ModelNumber ($PCL_PAPER_SIZE $PCL_PJL $PCL_PJL_RESOLUTION)

// List the fonts that are supported, in this case all standard
// fonts...
Font *

// Manufacturer and driver version
Manufacturer "HP"
Version 2.0

// Supported page sizes and their margins
HWMargins 18 12 18 12
*MediaSize Letter
MediaSize Legal
MediaSize Executive
MediaSize Monarch
MediaSize Statement
MediaSize FanFoldGermanLegal

HWMargins 18 12.72 18 12.72
MediaSize Env10

HWMargins 9.72 12 9.72 12
MediaSize A4
MediaSize A5
MediaSize B5
MediaSize EnvC5
MediaSize EnvDL
MediaSize EnvISOB5
MediaSize Postcard
MediaSize DoublePostcard
```

Listing 3-2, "examples/laserjet-pjl.drv"

```
// Only black-and-white output with mode 3 compression...
ColorModel Gray k chunky 3

// Supported resolutions
Resolution - 1 0 0 0 "300dpi/300 DPI"
*Resolution - 8 0 0 0 "600dpi/600 DPI"

// Supported input slots
*InputSlot 7 "Auto/Automatic Selection"
InputSlot 2 "Manual/Tray 1 - Manual Feed"
InputSlot 4 "Upper/Tray 1"
InputSlot 1 "Lower/Tray 2"
InputSlot 5 "LargeCapacity/Tray 3"

// Tray 3 is an option...
Installable "OptionLargeCapacity/Tray 3 Installed"
UIConstraints "*OptionLargeCapacity False *InputSlot LargeCapacity"

// PJL options
Attribute cupsPJL cupsRET "@PJL SET SMOOTHING=%?False:OFF;%?True:ON;%n"

Option "cupsRET/Smoothing" Boolean DocumentSetup 10
  Choice "False/Off" ""
  *Choice "True/On" ""

Attribute cupsPJL cupsTonerSave "@PJL SET ECONOMODE=%?False:OFF;%?True:ON;%n"

Option "cupsTonerSave/Save Toner" Boolean DocumentSetup 10
  *Choice "False/No" ""
  Choice "True/Yes" ""

{
  // HP LaserJet 2100 Series
  Throughput 10
  ModelName "LaserJet 2100 Series PJL"
  PCFileName "hpljt212.ppd"
}

{
  // LaserJet 2200 and 2300 series have duplexer option...
  Duplex normal
  Installable "OptionDuplex/Duplexer Installed"
  UIConstraints "*OptionDuplex False *Duplex"

  {
    // HP LaserJet 2200 Series
    Throughput 19
    ModelName "LaserJet 2200 Series PJL"
    PCFileName "hpljt222.ppd"
  }

  {
    // HP LaserJet 2300 Series
    Throughput 25
    ModelName "LaserJet 2300 Series PJL"
    PCFileName "hpljt232.ppd"
  }
}
```

Listing 3-2, "examples/laserjet-pjl.drv", continued...

The ESC/P Driver

The ESC/P driver includes a printer command filter called `commandtoescpx` and a raster printer driver filter called `rastertoescpx`. The command filter supports head cleaning, printing a self−test page, and ink cartridge alignment.

The raster printer driver filter accepts grayscale, RGB, and CMYK raster data for printing to inkjet devices. It supports EJL commands for device−specific features and uses the appropriate ESC/P2 raster graphics commands for all inkjet printers sold by Epson. The driver does *not* support native text rendering due to the limitations of text support in most ESC/P implementations.

The ESC/P driver also provides an include file, `<escp.h>`. Driver information files that use the ESC/P driver begin with the following:

```
#include <font.defs>
#include <media.defs>
#include <raster.defs>
#include <escp.h>

DriverType escp
```

Constant	Description
ESCP_MICROWEAVE	Use microweave command?
ESCP_STAGGER	Are color jets staggered?
ESCP_ESCK	Use print mode command?
ESCP_EXT_UNITS	Use extended unit commands?
ESCP_EXT_MARGINS	Use extended margin command?
ESCP_USB	Send USB packet mode escape
ESCP_PAGE_SIZE	Use page size command
ESCP_RASTER_ESCI	Use `ESC i` graphics command
ESCP_REMOTE	Use remote mode commands

Table 3−4, ESC/P ModelNumber constants

ModelNumber Constants

Table 3−4 shows the constants that are defined in the `<escp.h>` include file. These constants are used with the `ModelNumber` directive to control the behavior of the driver. For example, a typical Epson Stylus Color printer would use the following `ModelNumber` specification:

```
ModelNumber ($ESCP_MICROWEAVE $ESCP_USB $ESCP_REMOTE)
```

The parenthesis around the ESCP constants tell the PPD compiler to compute the bitwise OR of each of the values. The ESC/P driver will then use this information to tailor the output of the driver for the printer, in this case to use the printer's built−in microweaving (a way of printing using multiple passes), send the USB packet mode

escape sequence, and to use remote mode commands. Table 3−5 shows the constants to use for several common types of Epson printers.

Printer Model	ModelNumber Value
Epson Stylus Color	($ESCP_MICROWEAVE $ESCP_USB $ESCP_REMOTE)
Epson Stylus C−Series	($ESCP_STAGGER $ESCP_USB $ESCP_REMOTE)
Epson Stylus Photo	($ESCP_MICROWEAVE $ESCP_ESCK $ESCP_USB $ESCP_REMOTE)
Epson Stylus Pro	($ESCP_MICROWEAVE $ESCP_EXT_UNITS $ESCP_EXT_MARGINS $ESCP_PAGE_SIZE $ESCP_REMOTE)

Table 3−5, ModelNumber values for common Epson printers

Writing a Basic Epson Stylus Photo R300 Driver

Now that we have covered the ESC/P driver definitions, we will create a driver for the popular Epson Stylus Photo R300 series printers. These printers offer 7−color, full−bleed printing at up to 5760x1440 DPI and can print on recordable CDs and DVDs as well as standard printer media.

Our first cut of the driver will support printing at up to 1440 DPI. We'll add the full−bleed printing support in the next section. Listing 3−3 shows the driver information file for a basic Epson Stylus Photo R300 driver.

The file starts with the usual `#include` directives and then sets the driver type and model number so that we use the ESC/P driver with the output tailored to the R300:

```
// Specify that this driver uses the ESC/P driver...
DriverType escp

// Specify the driver options via the model number...
ModelNumber ($ESCP_ESCK $ESCP_EXT_UNITS $ESCP_EXT_MARGINS $ESCP_USB
             $ESCP_PAGE_SIZE $ESCP_RASTER_ESCI)
```

Then we list all of the media sizes that are supported by the printers along with the margins that should be used:

```
HWMargins 8.4 0 8.4 0
*MediaSize Letter
MediaSize Legal
MediaSize Executive
MediaSize Statement
MediaSize A4
MediaSize A5
MediaSize A6
MediaSize B5
MediaSize Env10
MediaSize EnvC5
MediaSize EnvDL
MediaSize EnvISOB5
MediaSize Postcard
MediaSize DoublePostcard
```

```
// Include standard font and media definitions
#include <font.defs>
#include <media.defs>

// Include ESC/P driver definitions
#include <escp.h>

// Specify that this driver uses the ESC/P driver...
DriverType escp

// Specify the driver options via the model number...
ModelNumber ($ESCP_ESCK $ESCP_EXT_UNITS $ESCP_EXT_MARGINS $ESCP_USB
             $ESCP_PAGE_SIZE $ESCP_RASTER_ESCI)

// List the fonts that are supported, in this case all standard
// fonts...
Font *

// Manufacturer and driver version
Manufacturer "Epson"
Version 1.0

// Supported page sizes and their margins
HWMargins 8.4 0 8.4 0
*MediaSize Letter
MediaSize Legal
MediaSize Executive
MediaSize Statement
MediaSize A4
MediaSize A5
MediaSize A6
MediaSize B5
MediaSize Env10
MediaSize EnvC5
MediaSize EnvDL
MediaSize EnvISOB5
MediaSize Postcard
MediaSize DoublePostcard

VariablePaperSize Yes
MinSize 1in 4in
MaxSize 8.5in 44in

// Four color modes are supported...
ColorModel Gray/Grayscale w chunky 1
ColorModel Black k chunky 1
*ColorModel RGB/Color rgb chunky 1
ColorModel CMYK cmyk chunky 1

// Supported resolutions
Resolution - 8 90 0 103 "360dpi/360 DPI"
*Resolution - 8 90 0 206 "720dpi/720 DPI"
Resolution - 8 90 0 412 "1440dpi/1440 DPI"

// Very basic dithering settings
Attribute cupsInkChannels "" 7
Attribute cupsInkLimit "" 2.0
```

Listing 3-3, "examples/r300-basic.drv"

```
Attribute cupsCyanLtDk "" "0.5 1.0"
Attribute cupsMagentaLtDk "" "0.5 1.0"
Attribute cupsBlackLtDk "" "0.5 1.0"

Attribute cupsAllDither 360dpi "0.5 0.75 1.0"
Attribute cupsAllDither 720dpi "0.6 0.9 1.2"
Attribute cupsAllDither 1440dpi "0.9 1.35"

Attribute cupsESCPDotSize 360dpi 16
Attribute cupsESCPDotSize 720dpi 17
Attribute cupsESCPDotSize 1440dpi 18

{
  // EPSON Stylus Photo R300 Series
  Throughput 1
  ModelName "Epson Stylus Photo R300"
  PCFileName "epspr301.ppd"
}
```

Listing 3−3, "examples/r300−basic.drv" continued...

The R300 supports custom page sizes up to 44 inches (1.1m) in length. We use the `VariablePaperSize` directive to instruct the PPD compiler to include the custom page size attributes, and the `MinSize` and `MaxSize` directives to specify the range of sizes that are supported:

```
VariablePaperSize Yes
MinSize 1in 4in
MaxSize 8.5in 44in
```

The R300 driver also supports four color modes: Grayscale (colorspace = w), Black (colorspace = k), RGB (colorspace = rgb), and CMYK (colorspace = cmyk). The Grayscale and RGB modes provide color/gamma−corrected output while the Black and CMYK modes offer uncorrected color printing. The `ColorModel` directive tells the PPD compiler to include each of these modes:

```
ColorModel Gray/Grayscale w chunky 1
ColorModel Black k chunky 1
*ColorModel RGB/Color rgb chunky 1
ColorModel CMYK cmyk chunky 1
```

The driver provides printing at 360, 720, and 1440 DPI using the `Resolution` directives. Each resolution makes use of the bits per color, row count, and row step values:

```
Resolution − 8 90 0 103 "360dpi/360 DPI"
*Resolution − 8 90 0 206 "720dpi/720 DPI"
Resolution − 8 90 0 412 "1440dpi/1440 DPI"
```

The R300 has 90 nozzles per color spaced at 120 DPI vertically; the head controller can print 360 DPI horizontally. The row count value is the same as the nozzle count, 90. The row step values require a small amount of calculation; each value is computed using the following formula:

```
row−step = 100 * x−dpi / 360 + y−dpi / 120
```

For 360x360 DPI, the row step is therefore:

```
row-step = 100 * x-dpi / 360 + y-dpi / 120
         = 100 * 360 / 360 + 360 / 120
         = 100 * 1 + 3
         = 100 + 3
         = 103
```

The 720 and 1440 DPI resolutions are computed similarly, and it is possible to support any multiple of the base resolution, 360x120 DPI, up to the physical limit of the printer controller, 5760x1440 DPI.

The R300 driver also needs some additional attributes defined to control the amount and kind of ink that is printed on the page. The first attribute we need to define is `cupsInkChannels` which tells the driver how many colors are used by the printer. In this case, the R300 is a 7-color printer:

```
Attribute cupsInkChannels "" 7
```

Next we want to limit the amount of ink that is put on the page to 200%. The `cupsInkLimit` attribute specifies this value:

```
Attribute cupsInkLimit "" 2.0
```

Since the R300 uses light versions of cyan, magenta, and black, we need to tell the driver when to use the light ink and when to use the dark ink. The simplest attributes for this specify the transition range as two numbers from 0 to 1. We'll transition from 0.5 to 1.0 for all three colors:

```
Attribute cupsCyanLtDk "" "0.5 1.0"
Attribute cupsMagentaLtDk "" "0.5 1.0"
Attribute cupsBlackLtDk "" "0.5 1.0"
```

Next, we want to specify the ink density for the dots that the R300 can produce for each resolution using the `cupsAllDither` attribute. Each number represents a percentage of ink, so a value of 1.0 means 100% coverage for each dot and a value of 2.0 means 200% coverage for each dot. The driver uses this information to reduce the amount of ink that is put on the page for each individual color. Since the R300 supports three different dot sizes, we include three numbers for the 360 and 720 DPI modes, however at 1440 DPI the largest dot size is not needed:

```
Attribute cupsAllDither 360dpi "0.5 0.75 1.0"
Attribute cupsAllDither 720dpi "0.6 0.9 1.2"
Attribute cupsAllDither 1440dpi "0.9 1.35"
```

Finally, we tell the driver which dot sizes to use for each resolution using the `cupsESCPDotSize` attribute. These values are defined in the corresponding developer reference manual from Epson. For the R300, size 16 represents the largest variable size dots and size 18 the smallest:

```
Attribute cupsESCPDotSize 360dpi 16
Attribute cupsESCPDotSize 720dpi 17
Attribute cupsESCPDotSize 1440dpi 18
```

To test the new drivers, start by running the `ppdc` program to create the PPD files:

```
ppdc r300-basic.drv ENTER
```

Then use the `lpadmin(8)` command to add the printer with the correct device URI. The following example adds a R300 which is connected via a USB port:

```
lpadmin −p r300 −E −v 'usb://EPSON/Stylus%20Photo%20R300' −i ppd/epspr301.ppd ENTER
```

Finally, print a test page to see it work:

```
lp −d r300 /usr/share/cups/data/testprint.ps ENTER
```

Epson Remote Mode Attributes

Most Epson inkjet printers support a special "remote" mode which allows you to control things such as the paper cutter, media type, drying time, and so forth. Remote mode also allows you to query the current ink status from the printer, clean the print heads, etc., and that functionality is supported via the `commandtoescpx` filter.

Remote mode support is specified using the `ESCP_REMOTE` model number constant. If remote mode is enabled, the `rastertoescpx` driver will look for several attributes to determine which remote mode commands to send and with what values. Table 3−6 lists the remote mode attributes and their values.

Attribute	Description
`cupsESCPAC`	Enables/disables the cutter based upon the `CutMedia` value
`cupsESCPCO`	Enables/disables the cutter based upon the `CutMedia` value
`cupsESCPEX`	Sets the media position based upon the `MediaPosition` value
`cupsESCPFP`	Sets the full−bleed horizontal position offset
`cupsESCPMS`	Sets the media size based upon the `MediaPosition` and `PageSize` values
`cupsESCPMT`	Sets the media type based upon the `cupsMediaType` value
`cupsESCPPC`	Sets paper checking based upon the `MediaPosition` value
`cupsESCPPH`	Sets the paper thickness based upon the `cupsMediaType` value
`cupsESCPPP`	Sets the paper path based upon the `MediaPosition` value
`cupsESCPSN0`	Sets the feed sequence based upon the `cupsMediaType` value
`cupsESCPSN1`	Sets the platten gap based upon the `cupsMediaType` value
`cupsESCPSN2`	Sets the paper feed/eject sequence based upon the `cupsMediaType` value
`cupsESCPSN6`	Sets the eject delay based upon the `cupsMediaType` value
`cupsESCPSN80`	Sets the cutting method based upon the `CutMedia` value
`cupsESCPSN81`	Sets the cutting pressure based upon the `CutMedia` value

Table 3−6, Epson remote mode attributes

Typically, each attribute maps from a single, integer value from the page device dictionary to a single integer value which is used for the corresponding remote mode command. The only exception to this rule is the

`cupsESCPPP` attribute which maps the `MediaPosition` value to two integer values for the "PP" (paper path) remote mode command, for example:

```
Attribute cupsESCPPP 0 "1 255"
```

The integer values used for all remote mode commands are documented in the corresponding Epson programming guide for your printer.

Adding Remote Mode Commands to the R300 Driver

The R300 supports several remote mode commands that we can use. We will add support for full−bleed printing and specification of the media source. Listing 3−4 shows the updated driver information file. We'll start by adding the `ESCP_REMOTE` constant to the model number definition:

```
ModelNumber ($ESCP_ESCK $ESCP_EXT_UNITS $ESCP_EXT_MARGINS $ESCP_USB
             $ESCP_PAGE_SIZE $ESCP_RASTER_ESCI $ESCP_REMOTE)
```

Next we add the horizontal offset attribute, `cupsESCPFP`, that is used to offset the page for full−bleed printing:

```
Attribute cupsESCPFP "" −80
```

Finally, we add `InputSlot` definitions for automatic and manual feed printing, and `cupsESCPPP` attributes for each `MediaPosition` value:

```
*InputSlot 0 "Auto/Auto Select"
InputSlot 1 "Manual/Manual Feed"

Attribute cupsESCPPP 0 "1 255"
Attribute cupsESCPPP 1 "2 1"
```

To test the new drivers, start by running the `ppdc` program to create the PPD files:

```
ppdc r300−remote.drv ENTER
```

Then use the `lpadmin(8)` command to add the printer with the correct device URI. The following example adds a R300 which is connected via a USB port:

```
lpadmin −p r300 −E −v 'usb://EPSON/Stylus%20Photo%20R300' −i ppd/epspr302.ppd ENTER
```

Finally, print a test page to see it work:

```
lp −d r300 /usr/share/cups/data/testprint.ps ENTER
```

```
// Include standard font and media definitions
#include <font.defs>
#include <media.defs>

// Include ESC/P driver definitions
#include <escp.h>

// Specify that this driver uses the ESC/P driver...
DriverType escp

// Specify the driver options via the model number...
ModelNumber ($ESCP_ESCK $ESCP_EXT_UNITS $ESCP_EXT_MARGINS $ESCP_USB
             $ESCP_PAGE_SIZE $ESCP_RASTER_ESCI $ESCP_REMOTE)

// List the fonts that are supported, in this case all standard
// fonts...
Font *

// Manufacturer and driver version
Manufacturer "Epson"
Version 2.0

// Supported page sizes and their margins
HWMargins 0 0 0 0
*MediaSize Letter
MediaSize Legal
MediaSize Executive
MediaSize Statement
MediaSize A4
MediaSize A5
MediaSize A6
MediaSize B5
MediaSize Env10
MediaSize EnvC5
MediaSize EnvDL
MediaSize EnvISOB5
MediaSize Postcard
MediaSize DoublePostcard

VariablePaperSize Yes
MinSize 1in 4in
MaxSize 8.5in 44in

// Borderless printing offset...
Attribute cupsESCPFP "" -80

// Four color modes are supported...
ColorModel Gray/Grayscale w chunky 1
ColorModel Black k chunky 1
*ColorModel RGB/Color rgb chunky 1
ColorModel CMYK cmyk chunky 1

// Supported resolutions
Resolution - 8 90 0 103 "360dpi/360 DPI"
*Resolution - 8 90 0 206 "720dpi/720 DPI"
Resolution - 8 90 0 412 "1440dpi/1440 DPI"
```

Listing 3-4, "examples/r300-remote.drv"

```
// Paper trays...
*InputSlot 0 "Auto/Auto Select"
InputSlot 1 "Manual/Manual Feed"

Attribute cupsESCPPP 0 "1 255"
Attribute cupsESCPPP 1 "2 1"

// Very basic dithering settings
Attribute cupsInkChannels "" 7
Attribute cupsInkLimit "" 2.0

Attribute cupsCyanLtDk "" "0.5 1.0"
Attribute cupsMagentaLtDk "" "0.5 1.0"
Attribute cupsBlackLtDk "" "0.5 1.0"

Attribute cupsAllDither 360dpi "0.5 0.75 1.0"
Attribute cupsAllDither 720dpi "0.6 0.9 1.2"
Attribute cupsAllDither 1440dpi "0.9 1.35"

Attribute cupsESCPDotSize 360dpi 16
Attribute cupsESCPDotSize 720dpi 17
Attribute cupsESCPDotSize 1440dpi 18

{
  // EPSON Stylus Photo R300 Series
  Throughput 1
  ModelName "Epson Stylus Photo R300"
  PCFileName "epspr302.ppd"
}
```

Listing 3−4, "examples/r300−remote.drv" continued...

4 – Developing PostScript Printer Drivers

This chapter describes how to develop PPD files for PostScript printer drivers.

Overview of PostScript Driver Development

The PPD compiler is capable of producing PPD files for PostScript printers just as easily as for non−PostScript printers. PostScript printer drivers use the `ps` driver type:

```
DriverType ps
```

Required Attributes

PostScript drivers require the attributes listed in Table 4−1. If not specified, the defaults for CUPS drivers are used. A typical PostScript driver information file would include the following attributes:

```
Attribute DefaultColorSpace "" Gray
Attribute LandscapeOrientation "" Minus90
Attribute LanguageLevel "" "3"
Attribute Product "" "(Foo LaserProofer 2000)"
Attribute PSVersion "" "(3010) 0"
Attribute TTRasterizer "" Type42
```

Attribute	Description>
DefaultColorSpace	The default colorspace: Gray, RGB, CMY, or CMYK. If not specified, then RGB is assumed.
LandscapeOrientation	The preferred landscape orientation: Plus90, Minus90, or Any. If not specified, Plus90 is assumed.
LanguageLevel	The PostScript language level supported by the device: 1, 2, or 3. If not specified, 2 is assumed.
Product	The string returned by the PostScript product operator, which *must* include parenthesis to conform with PostScript syntax rules for strings. Multiple Product attributes may be specified to support multiple products with the same PPD file. If not specified, "(ESP Ghostscript)" and "(GNU Ghostscript)" are assumed.
PSVersion	The PostScript interpreter version numbers as returned by the version and revision operators. The required format is "(version) revision". Multiple PSVersion attributes may be specified to support multiple interpreter version numbers. If not specified, "(3010) 705" and "(3010) 707" are assumed.
TTRasterizer	The type of TrueType font rasterizer supported by the device, if any. The supported values are None, Accept68k, Type42, and TrueImage. If not specified, None is assumed.

Table 4−1, Required PostScript printer driver attributes

Query Commands

Most PostScript printer PPD files include query commands (?PageSize, etc.) that allow applications to query the printer for its current settings and configuration. Query commands are included in driver information files as attributes. For example, the query command for the PageSize option might look like the following:

```
Attribute "?PageSize" "" "
      save
      currentpagedevice /PageSize get aload pop
      2 copy gt {exch} if (Unknown)
      23 dict
              dup [612 792] (Letter) put
              dup [612 1008] (Legal) put
              dup [595 842] (A4) put
              {exch aload pop 4 index sub abs 5 le exch
               5 index sub abs 5 le and
              {exch pop exit} {pop} ifelse
      } bind forall = flush pop pop
      restore"
```

Query commands can span multiple lines, however no single line may contain more than 255 characters.

Adding Filters

Normally a PostScript printer driver will not utilize any additional print filters. For drivers that provide additional filters such as a CUPS command file filter for doing printer maintenance, you must also list the following `Filter` directive to handle printing PostScript files:

```
Filter application/vnd.cups-postscript 0 -
```

Importing Existing PostScript Drivers

The `ppdi(1)` utility included with the CUPS DDK imports existing PPD files into driver information files. This allows you to make modifications and localize PPD files for other languages with great ease. Use the following command to import a single PPD file called *filename.ppd* into a driver information file called *filename.drv*:

```
ppdi filename.drv filename.ppd ENTER
```

The driver information file is created if it does not exist. Otherwise the PPD file information is appended to the end of the file. You can use shell wildcards to import whole directories of PPD files:

```
ppdi filename.drv *.ppd ENTER
```

Once imported, you can edit the driver information file and use the `ppdc` program to regenerate the PPD files:

```
ppdc filename.drv ENTER
```

5 – Localizing Printer Drivers

This chapter describes how to localize printer drivers for different languages and defaults.

Overview of the Localization Process

The PPD compiler provides localization of PPD files in different languages through *message catalog* files in the GNU gettext format. Each user text string and several key PPD attribute values such as `LanguageVersion` and `LanguageEncoding` are looked up in the corresponding message catalog and the translated text is substituted in the generated PPD files. One message catalog file can be used by multiple driver information files, and each file contains a single language translation.

The DDK provides a utility program to aid in the localization of drivers called `ppdpo(1)`. In addition, the standard DDK installation includes basic localizations of all standard media sizes and options in English, French, German, Italian, and Spanish.

Localizations are created using a few options to the PPD compiler which are covered later in this chapter.

The Message Catalog File Format

Each message catalog file can be edited with your favorite text editor and consists of one or more messages translated into a single language. Comment lines can be included using the # character, for example:

```
# This is a comment
```

Each message is specified using a pair of directives: `msgid` and `msgstr`. The `msgid` string specifies the original (probably English) version of a string which is looked up when localizing a PPD file. The `msgstr` string contains the translated message. For example, the following message catalog translates the word "Yes" to French:

```
msgid "Yes"
msgstr "Oui"
```

Each message catalog will also contain two special entries for the `LanguageVersion` and `LanguageEncoding` attribute values. The standard `msgid` values are "English" and "ISOLatin1", respectively. They should be replaced with the Adobe−defined keywords for the language being localized and the character encoding, respectively. Table 5−1 lists the standard `LanguageVersion`, `LanguageEncoding`, and corresponding POSIX language abbreviation values that are supported.

LanguageVersion	LanguageEncoding	POSIX
English	ISOLatin1	en
Chinese	None	zh
Danish	ISOLatin1	da
Dutch	ISOLatin1	nl
Finnish	ISOLatin1	fi
French	ISOLatin1	fr
German	ISOLatin1	de
Italian	ISOLatin1	it
Japanese	JIS83−RKSJ	ja
Norwegian	ISOLatin1	no
Portuguese	ISOLatin1	pt
Russian	None	ru
Spanish	ISOLatin1	es
Swedish	ISOLatin1	sv
Turkish	None	tr

Table 5−1, Supported LanguageVersion, LanguageEncoding, and POSIX language abbreviation values.

The ppdpo Localization Utility

The `ppdpo` program creates or updates a message catalog file based upon one or more driver information files. New messages are added with the word "TRANSLATE" added to the front of the translation string to make locating new strings for translation easier. The program accepts the message catalog filename and one or more driver information files.

For example, run the following command to create a new German message catalog called *de.po* for all of the driver information files in the current directory:

```
ppdpo -o de.po *.drv
```

If the file *de.po* already exists, `ppdpo` will update the contents of the file with any new messages that need to be translated.

Using a Message Catalog with the PPD Compiler

Once you have created a message catalog, use the `-c`, `-l`, and `-d` options with `ppdc` to create PPD files in alternate languages. The `-c` option specifies the message catalog to use, the `-l` option specifies the POSIX language abbreviation, and the `-d` option specifies the output directory. For example, use the following command to generate the German PPD files for the drivers listed in *mydrivers.drv* into the directory *ppd/de*:

```
ppdc -c de.po -l de -d ppd/de mydrivers.drv
```

6 − Doing Raster Color Management

This chapter describes how to use the built−in raster color management functions to produce enhanced color and grayscale output on raster printers.

Overview of Raster Color Management

CUPS provides several ways for CUPS−based raster printer drivers to manage the colors produced by a non−PostScript printer. Figure 6−1 shows the raster processing workflow within CUPS. Starting with the print file, applications can embed ICC and other types of color profiles to alter the printed output.

Once converted to a RIP file format, the standard RIP filters can perform various linear transformations during rasterization, such as gamma correction, density adjustment, and color transforms using a matrix.

Upon arrival at the printer driver filter, the driver can perform any number of linear and non−linear transformations as needed. The DDK drivers and custom driver API offer non−linear 3D lookup tables for transforming grayscale and RGB colors to a K or CMYK representation and non−linear 2D lookup tables for transforming K or CMYK colors to device color values. Once converted to a device color, the color values are then dithered, as necessary, using per−color dithering parameters specific to the target device, rendering mode, resolution, and media.

Color management for PostScript and Ghostscript−based printer drivers is not currently supported by CUPS. Please consult the corresponding vendor documentation for any color management capabilities that are offered for those printers and drivers.

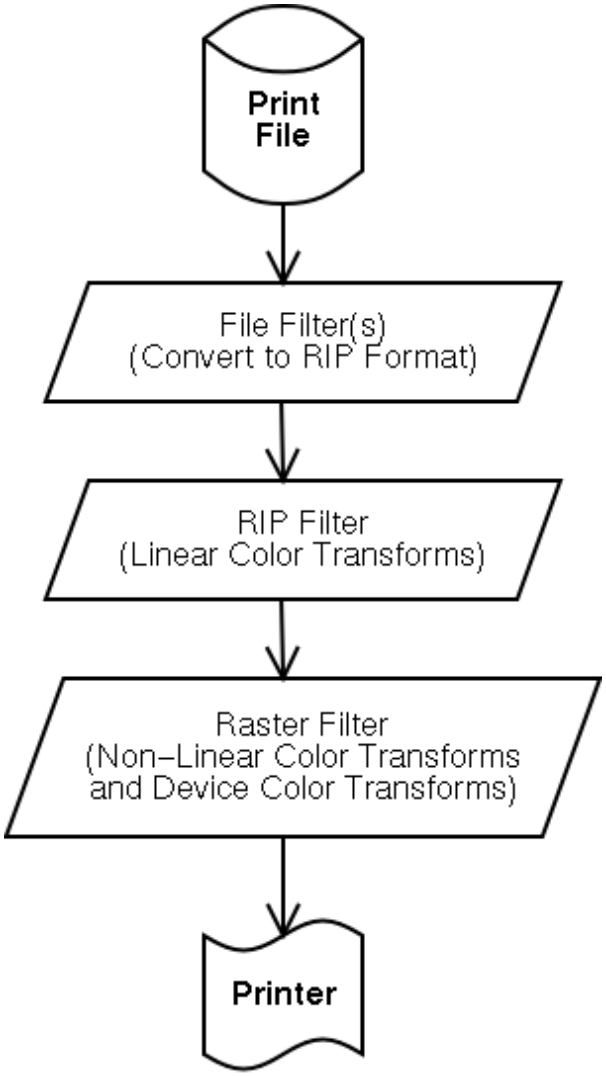

Figure 6−1, The CUPS raster processing workflow

RIP−Based Color Profiles

The first type of color profile supported by CUPS is provided by the RIP filters and is defined using the `cupsColorProfile` attribute in the PPD file or the `profile` attribute in the job options. Each profile consists of 11 numbers: the density, gamma, and CMY 3x3 color transform matrix. The `cupsColorProfile` PPD attribute value contains 11 real numbers separated by spaces, while the `profile` job attribute contains 11 integers separated by commas (`,`); multiple the PPD attribute numbers by 1000 to compute the job attribute numbers.

The `cupsColorProfile` attribute is specified using the `ColorProfile` or `SimpleColorProfile` directives in a driver information file. Each color profile can apply to a specific combination of resolution and media type or to all resolutions or media types. The RIP filter will use the first matching profile in the PPD file, so it is important to specify the specific profiles first followed by the general ones.

The `cupsprofile(1)` utility can be used to generate a RIP−based color profile using a series of four test pages. The results are suitable for calibrating basic business−type graphics to a particular combination of resolution and media.

One limitation of RIP−based color profiles is that they do not allow for non−linear color transformations typically required for accurate color reproduction.

Driver−Based Color Profiles

The DDK drivers and driver API provide two types of color profiles: non−linear 3D RGB or grayscale color lookup table−based profiles, linear and non−linear 2D device color lookup table−based profiles, and 2D color dithering tables. Driver profiles are available only as attributes in the printer's PPD file.

RGB Color Profiles

RGB profiles utilize 3D color lookup tables that map from sRGB color values to CMYK color values. Two PPD attributes, `cupsRGBProfile` and `cupsRGBSample`, are used to specify an RGB color profile. The `cupsRGBProfile` attribute specifies the number of samples on each side of the lookup cube, the number of color channels (1, 3, or 4 for K, CMY, and CMYK devices), and the number of color samples in the profile:

```
*cupsRGBProfile Glossy.720dpi: "3 4 27"
```

Currently, the number of color samples must equal the cube of the cube size, that is an RGB color profile with a cube size of 4 must have 4 * 4 * 4 = 64 color samples which map the sRGB colors to CMYK. Also, the RGB samples must be spread evenly over the color cube.

Each color sample is represented using `cupsRGBSample` attributes. The value of each attribute consists of the sRGB and CMYK color values separated by spaces. Each color value is a real number from 0 to 1. For example, a sample for black which maps to a CMYK color of 0,0,0,1 might look like the following:

```
*cupsRGBSample Glossy.720dpi: "0 0 0 0 0 0 1"
```

Similarly, blue mapping to a CMYK value of 1,0.5,0,0 would look like:

```
*cupsRGBSample Glossy.720dpi: "0 0 1 1 0.5 0 0"
```

CMYK Color Profiles

CMYK color profiles map K, CMY, or CMYK colors to device colors. They are normally used to provide linear CMYK output and to transition between light and dark inks. When combined with application or RGB profiles, CMYK profiles allow for easier profiling using existing 4−color profiling software.

The CMYK color profiles are implemented as 2D lookup tables; the tables can be generated automatically using gamma and density values, or manually using measurements of specific values. For printers that support more than 4 inks, the light ink lookup tables are driven by the main color, e.g. the light cyan output value is based upon the cyan input value.

CMYK color profiles are only available via PPD attributes. Table 6−1 lists the supported attributes. The `cupsInkChannels` attribute specifies the number of output color channels, typically 1, 3, 4, 6, or 7.

The `cupsInkLimit` attribute specifies the maximum amount of ink that can be placed on the page. When the total amount of ink exceeds the ink limit, all color channels are scaled linearly to the specified limit.

Attribute	Description
cupsAllGamma	Set default curve using gamma + density
cupsAllXY	Set default curve using XY points
cupsBlackGamma	Set black curve using gamma + density
cupsBlackGeneration	Set black generation
cupsBlackLightDark	Set black light/dark transition
cupsBlackXY	Set black curve using XY points
cupsCyanGamma	Set cyan curve using gamma + density
cupsCyanLightDark	Set cyan light/dark transition
cupsCyanXY	Set cyan curve using XY points
cupsInkChannels	Set number of color channels
cupsInkLimit	Set total ink limit
cupsLightBlackGamma	Set light black curve using gamma + density
cupsLightBlackXY	Set light black curve using XY points
cupsLightCyanGamma	Set light cyan curve using gamma + density
cupsLightCyanXY	Set light cyan curve using XY points
cupsLightMagentaGamma	Set light magenta curve using gamma + density
cupsLightMagentaXY	Set light magenta curve using XY points
cupsMagentaGamma	Set magenta curve using gamma + density
cupsMagentaLightDark	Set magenta light/dark transition
cupsMagentaXY	Set magenta curve using XY points
cupsYellowGamma	Set yellow curve using gamma + density
cupsYellowXY	Set yellow curve using XY points

Table 6−1, CMYK color profile attributes

The `cupsBlackGeneration` attribute specifies a transition range for representing shades of gray as a combination of CMY or as K. The first number specifies the start of the transition range where K = 0 and the second the end of the transition range where CMY = 0.

The gamma attributes define the gamma and density of the specified color channel. For example, the following attribute sets the default gamma and density at 1440dpi to 2.0 and 0.9, respectively:

```
Attribute cupsAllGamma 1440dpi "2.0 0.9"
```

The XY attributes specify a point in a piecewise linear curve from an input color value to a measured output value. For example, the XY values for four measurements at 1440dpi would look like the following:

```
Attriute cupsAllXY 1440dpi "0.25 0.337"
Attriute cupsAllXY 1440dpi "0.5 0.633"
Attriute cupsAllXY 1440dpi "0.75 0.91"
Attriute cupsAllXY 1440dpi "1.0 1.24"
```

The `LtDk` ("light dark") attributes specify a transition area for light to dark inks using the dark ink lookup table. The first value specifies the input color value which maps to 100% of the light color output value and the second value specifies the input color value which maps to 0% of the light color output value. The dark color is mapped to 0 up to the first value and is ramped up to the second value as the light color decreases.

Dithering Tables

Once converted to device colors, the dither functions use PPD attributes to define the dot densities for each supported size. Table 6−2 lists the supported attributes. For example, the following attributes might be used to define the default dither values for three resolutions:

```
Attribute cupsAllDither 360dpi "0.5 0.75 1.0"
Attribute cupsAllDither 720dpi "0.6 0.9 1.2"
Attribute cupsAllDither 1440dpi "0.9 1.35"
```

Attribute	Description
`cupsAllDither`	Sets the default dither values
`cupsBlackDither`	Sets the black dither values
`cupsCyanDither`	Sets the cyan dither values
`cupsLightBlackDither`	Sets the light black dither values
`cupsLightCyanDither`	Sets the light cyan dither values
`cupsLightMagentaDither`	Sets the light magenta dither values
`cupsMagentaDither`	Sets the maenta dither values
`cupsYellowDither`	Sets the yellow dither values

Table 6−2, Dither attributes

A − Software License Agreement

This appendix provides the software license agreement for the CUPS DDK and included drivers.

CUPS Driver Development Kit License Agreement

Copyright 1997−2004 by Easy Software Products
44141 AIRPORT VIEW DR STE 204
HOLLYWOOD, MARYLAND 20636−3142 USA

Voice: +1.301.373.9600
Email: cups−info@cups.org
WWW: http://www.cups.org/

Introduction

The CUPS™ Driver Development Kit ("DDK") is provided under the GNU General Public License ("GPL").

For those not familiar with the GNU GPL, the license basically allows you to:

- Use the CUPS DDK software at no charge.
- Distribute verbatim copies of the software in source or binary form.
- Sell verbatim copies of the software for a media fee, or sell support for the software.
- Distribute or sell printer drivers and filters that use the CUPS DDK so long as source code is made available under the GPL.

What this license **does not** allow you to do is make changes or add features to the CUPS DDK and then sell a binary distribution without source code. You must provide source for any new drivers, changes, or additions to the software, and all code must be provided under the GPL.

Trademarks

Easy Software Products has trademarked the Common UNIX Printing System, CUPS, and CUPS logo. These names and logos may be used freely in any direct port or binary distribution of the CUPS DDK. Please contract Easy Software Products for written permission to use them in derivative products. Our intention is to protect the value of these trademarks and ensure that any derivative product meets the same high−quality standards as the original.

Binary Distribution Rights

Easy Software Products also sells rights to the CUPS DDK source code under a binary distribution license for vendors that are unable to release source code for their drivers, additions, and modifications to CUPS under the GNU GPL. For information please contact us at the address shown above.

Support

Easy Software Products sells a commercial version of the CUPS DDK with software support. You can find out more at our web site:

```
http://www.easysw.com/
```

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111−1307 USA

Everyone is permitted to copy and distribute verbatim
copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software−−to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.
1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.
2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
   c. if the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

   In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.
3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a. Accompany it with the complete corresponding machine−readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine−readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty−free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the

free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

B – PPD Compiler Driver Information File Reference

This appendix describes all of the directives that may be used in driver information files.

Keyword Index

#define	DriverType	MediaType
#font	Duplex	MinSize
#include	Filter	ModelName
#media	Finishing	ModelNumber
Attribute	Font	Option
Choice	Group	PCFileName
ColorDevice	HWMargins	Resolution
ColorModel	InputSlot	SimpleColorProfile
ColorProfile	Installable	Throughput
Copyright	ManualCopies	UIConstraints
CustomMedia	Manufacturer	VariablePaperSize
Cutter	MaxSize	Version
Darkness	MediaSize	

#define

Syntax

```
#define name value
```

Examples

```
#define FOO 100
#define BAR "Bar, Inc."
```

Description

The `#define` directive assigns a value to a name which can be later referenced using `$name`. The name is case−insensitive and can be any sequence of letters, numbers, and the underscore. The value can be any number or string.

See Also

`#include`

#font

Syntax

```
#font name encoding "version" charset status
```

Examples

```
#font Courier Standard "(1.05)" Standard ROM
#font Symbol Special "(001.005)" Special ROM
#font Barcode-Foo Special "(1.0)" Special Disk
#font Unicode-Foo Expert "(2.0)" Adobe-Identity ROM
```

Description

The `#font` directive defines a "base font" for all printer drivers. The name is the PostScript font name.

The encoding is the default encoding of the font, usually `Standard`, `Expert`, or `Special`, as defined in the Adobe PPD file specification.

The version is the PostScript string definition that corresponds to the font version number.

The charset defines the available characters in the font, usually `Standard` or `Special`, as defined in the Adobe PPD file specification.

The status is the installation status of the font and must be either the word `ROM` or `Disk`.

Base fonts differ from fonts defined using the `Font` directive in that they are not automatically associated with all drivers − you must use the special `Font *` directive to include them in a driver.

Currently the `#font` directive is used mainly for defining the standard raster fonts in the `<font.defs>` include file.

See Also

`#include`, `Font`

#include

Syntax

```
#include <filename>
#include "filename"
```

Examples

```
#include <font.defs>
#include "myfile.h"
```

Description

The `#include` directive reads the named driver information file. If the filename is included inside angle brackets (`<filename>`), then the PPD compiler will look for the file in all of the include directories it knows about. Otherwise, the file is opened in the current directory relative to the current driver information file, and if that fails then it looks in the include directories for the file.

The `#include` directive can be nested to as many files as are allowed by the host operating system, typically at least 100 files.

See Also

`#define`, `#font`, `#media`

#media

Syntax

```
#media name width length
#media "name/text" width length
```

Examples

```
#media "Letter/Letter – 8.5x11in" 8.5in 11in
#media "A4/A4 – 210x297mm" 210mm 297mm
#media "w936h1368/Super B/A3 – 13x19in" 936 1368
#media Photo 4in 6in
```

Description

The `#media` directive defines a named media size for inclusion in a driver. The name with optional user text defines the name for the media size and is used with the `MediaSize` directive to associate the media size with the driver. The name may only contain letters, numbers, and the underscore and may not exceed 40 characters in length. The user text, if supplied, may not exceed 80 characters in length.

The width and length define the dimensions of the media. Each number is optionally followed by one of the following unit suffixes:

- `cm` − centimeters
- `ft` − feet
- `in` − inches
- `m` − meters
- `mm` − millimeters
- `pt` − points (72 points = 1 inch)

Points are assumed if no units are specified.

See Also

`#include`, `CustomMedia`, `MediaSize`

Attribute

Syntax

```
Attribute name "" value
Attribute name keyword value
Attribute name "keyword/text" value
```

Examples

```
Attribute cupsInkChannels "" 1
Attribute cupsAllDither 600dpi "1.0"
Attribute fooProfile "Photo/Photographic Profile" "photopro.icc"
```

Description

The `Attribute` directive creates a PPD attribute. The name is any combination of letters, numbers, and the underscore and can be up to 40 characters in length.

The selector can be the empty string (`""`), a keyword consisting of up to 40 letters, numbers, and the underscore, or a string composed of a keyword and user text of up to 80 characters.

The value is any string or number; the string may contain multiple lines, however no one line may exceed 255 characters.

Choice

Syntax

```
Choice name "code"
Choice "name/text" "code"
```

Examples

```
Choice None "<</MediaType (None)>>setpagedevice"
Choice "False/No" "<</cupsCompression 0>>setpagedevice"
```

Description

The `Choice` directive adds a single choice to the current option. The name is any combination of letters, numbers, and the underscore and can be up to 40 characters in length.

If provided, the text can be any string up to 80 characters in length. If no text is provided, the name is used.

The code is any string and may contain multiple lines, however no one line may exceed 255 characters.

See Also

`ColorModel`, `Cutter`, `Darkness`, `Duplex`, `Finishing`, `Group`, `InputSlot`, `Installable`, `MediaType`, `Option`, `Resolution`, `UIConstraints`

ColorDevice

Syntax

```
ColorDevice boolean-value
```

Examples

```
ColorDevice no
ColorDevice yes
```

Description

The `ColorDevice` directive tells the application if the printer supports color. It is typically used in conjunction with the `ColorModel` directive to provide color printing support.

See Also

`ColorModel`

ColorModel

Syntax

```
ColorModel name colorspace colororder compression
ColorModel "name/text" colorspace colororder compression
```

Examples

```
ColorModel Gray/Grayscale w chunky 0
ColorModel RGB/Color rgb chunky 0
ColorModel CMYK cmyk chunky 0
```

Description

The `ColorModel` directive is a convenience directive which creates a ColorModel option and choice for the current printer driver. The name is any combination of letters, numbers, and the underscore and can be up to 40 characters in length.

If provided, the text can be any string up to 80 characters in length. If no text is provided, the name is used.

The colorspace argument is one of the standard colorspace keywords defined later in this appendix in the section titled, "Colorspace Keywords".

The colororder argument is one of the standard color order keywords defined later in this appendix in the section titled, "Color Order Keywords".

The compression argument is any number and is assigned to the `cupsCompression` attribute in the PostScript page device dictionary.

See Also

`Choice`, `ColorDevice`, `Cutter`, `Darkness`, `Duplex`, `Finishing`, `Group`, `InputSlot`, `Installable`, `MediaType`, `Option`, `Resolution`, `UIConstraints`

ColorProfile

Syntax

```
ColorProfile resolution/mediatype gamma density matrix
```

Examples

```
ColorProfile −/− 1.7 1.0
     1.0    0.0    0.0
     0.0    1.0    0.0
     0.0    0.0    1.0

ColorProfile 360dpi/− 1.6 1.0
     1.0   −0.05  −0.3
    −0.35   1.0   −0.15
    −0.095 −0.238  0.95

ColorProfile 720dpi/Special 1.5 1.0
     1.0    0.0   −0.38
    −0.4    1.0    0.0
     0.0   −0.38   0.9
```

Description

The `ColorProfile` directive defines a CMY transform−based color profile. The resolution and mediatype arguments specify the `Resolution` and `MediaType` choices which use the profile; the hyphen (−) is used to specify that any resolution or mediatype can be used with the profile.

The gamma argument specifies the gamma correction to apply to the color values ($P = p^g$) and is a real number greater than 0. Values larger than 1 cause a general lightening of the print while values smaller than 1 cause a general darkening of the print. A value of 1 disables gamma correction.

The density argument specifies the linear density correction to apply to the color values ($P = d * p^g$) and is a real number greater than 0 and less than or equal to 1. A value 1 of disables density correction while lower values produce proportionately lighter output.

The matrix argument specifies a 3x3 linear transformation matrix in row−major order. The matrix is applied only to the CMY component of a RGB to CMYK transformation and is not used when printing in grayscale or CMYK mode unless the printer only supports printing with 3 colors.

See Also

`SimpleColorProfile`

Copyright

Syntax

```
Copyright "text"
```

Examples

```
Copyright "Copyright 2004 by Foo Enterprises"

Copyright
"This software is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License as
published by the Free Software Foundation; either version 2 of
the License, or (at your option) any later version.

This software is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public
License along with this software; if not, write to the Free
Software Foundation, Inc., 59 Temple Place, Suite 330, Boston,
MA 02111 USA"
```

Description

The `Copyright` directive adds text comments to the top of a PPD file, typically for use in copyright notices. The text argument can contain multiple lines of text, but no line may exceed 255 characters.

CustomMedia

Syntax

```
CustomMedia name width length left bottom right top
    "size-code" "region-code"

CustomMedia "name/text" width length left bottom right top
    "size-code" "region-code"
```

Examples

```
CustomMedia Letter 8.5in 11in 0.25in 0.46in 0.25in 0.04in
    "<</PageSize[612 792]/ImagingBBox null/ManualFeed false>>setpagedevice"
    "<</PageSize[612 792]/ImagingBBox null/ManualFeed true>>setpagedevice"

CustomMedia "A4/A4 - 210x297mm" 210mm 297mm 12 12 12 12
    "<</PageSize[595 842]/ImagingBBox null>>setpagedevice"
    "<</PageSize[595 842]/ImagingBBox null>>setpagedevice"
```

Description

The `CustomMedia` directive adds a custom media size to the driver. The name is any combination of letters, numbers, and the underscore and can be up to 40 characters in length.

If provided, the text can be any string up to 80 characters in length. If no text is provided, the name is used.

The width and length arguments specify the dimensions of the media as defined for the `#media` directive.

The left, bottom, right, and top arguments specify the printable margins of the media.

The size-code and region-code arguments specify the PostScript commands to run for the `PageSize` and `PageRegion` options, respectively. The commands can contain multiple lines, however no line may be more than 255 characters in length.

See Also

`#media`, `MediaSize`

Cutter

Syntax

```
Cutter boolean-value
```

Examples

```
Cutter yes
Cutter no
```

Description

The `Cutter` directive specifies whether the printer has a built-in media cutter. When a cutter is present, the printer's PPD file will contain a `CutMedia` option that allows the user to control whether the media is cut at the end of the job.

See Also

`Choice`, `ColorModel`, `Darkness`, `Duplex`, `Finishing`, `Group`, `InputSlot`, `Installable`, `MediaType`, `Option`, `Resolution`, `UIConstraints`

Darkness

Syntax

```
Darkness temperature name
Darkness temperature "name/text"
```

Examples

```
Darkness 0 Light
Darkness 2 "Normal/Standard"
```

Description

The `Darkness` directive defines a choice for the `cupsDarkness` option which sets the `cupsCompression` attribute in the PostScript page device dictionary. It is used with the CUPS *rastertodymo* sample driver to control the print head temperature and therefore the darkness of the print.

The temperature argument specifies a temperature value for the Dymo driver from 0 (lowest) to 3 (highest), with 2 representing the normal setting.

The name is any combination of letters, numbers, and the underscore and can be up to 40 characters in length.

If provided, the text can be any string up to 80 characters in length. If no text is provided, the name is used.

See Also

`Choice`, `ColorModel`, `Cutter`, `Duplex`, `Finishing`, `Group`, `InputSlot`, `Installable`, `MediaType`, `Option`, `Resolution`, `UIConstraints`

DriverType

Syntax

```
DriverType type
```

Examples

```
DriverType custom
DriverType escp
DriverType pcl
DriverType ps
```

Description

The `DriverType` directive tells the PPD compiler which DDK filters to include in the PPD file. The following types are supported:

- `custom` − Use only those filters that are defined in the driver information file
- `dymo` − Use the CUPS sample Dymo driver filter *rastertodymo*
- `epson` − Use the CUPS sample Epson driver filter *rastertoepson*
- `escp` − Use the ESC/P DDK driver filters *commandtoescpx* and *rastertoescpx*
- `hp` − Use the CUPS sample HP driver filter *rastertohp*
- `pcl` − Use the HP−PCL DDK driver filters *commandtopclx* and *rastertopclx*
- `ps` − Use no filters; this driver is for a standard PostScript device
- `rtl` − Use the CUPS sample RTL driver filter *rastertortl*

See Also

`Filter`, `ModelNumber`

Duplex

Syntax

```
Duplex type
```

Examples

```
Duplex none
Duplex normal
Duplex flip
```

Description

The `Duplex` directive determines whether double−sided printing is supported in the current driver. The type argument specifies the type of duplexing that is supported:

- `none` − double−sided printing is not supported
- `normal` − double−sided printing is supported
- `flip` − double−sided printing is supported, but the back side image needs to be flipped vertically (used primarily with inkjet printers)

See Also

`Choice`, `ColorModel`, `Cutter`, `Darkness`, `Finishing`, `Group`, `InputSlot`, `Installable`, `MediaType`, `Option`, `Resolution`, `UIConstraints`

Filter

Syntax

```
Filter mime-type cost program
```

Examples

```
Filter application/vnd.cups-raster 50 rastertofoo
Filter application/vnd.hp-HPGL 25 /usr/foo/filter/hpgltofoo
```

Description

The `Filter` directive adds a filter for the current driver. The mime-type argument is a valid MIME media type name as defined in a CUPS *mime.types* file.

The cost argument specifies the relative cost of the filter. In general, use a number representing the average percentage of CPU time that is used when printing the specified MIME media type.

The program argument specifies the program to run; if the program is not an absolute filename, then CUPS will look for the program in the CUPS filter directory.

See Also

`DriverType`

Finishing

Syntax

```
Finishing name
Finishing "name/text"
```

Examples

```
Finishing None
Finishing "Glossy/Photo Overcoat"
```

Description

The `Finishing` directive adds a choice to the `cupsFinishing` option. The name is any combination of letters, numbers, and the underscore and can be up to 40 characters in length. The name is stored in the `OutputType` attribute in the PostScript page device dictionary.

If provided, the text can be any string up to 80 characters in length. If no text is provided, the name is used.

See Also

`Choice`, `ColorModel`, `Cutter`, `Darkness`, `Duplex`, `Group`, `InputSlot`, `Installable`, `MediaType`, `Option`, `Resolution`, `UIConstraints`

Font

Syntax

```
Font name encoding "version" charset status
Font *
```

Examples

```
Font *
Font Courier Standard "(1.05)" Standard ROM
Font Symbol Special "(001.005)" Special ROM
Font Barcode-Foo Special "(1.0)" Special Disk
Font Unicode-Foo Expert "(2.0)" Adobe-Identity ROM
```

Description

The `Font` directive defines a "device font" for the current printer driver. The name is the PostScript font name.

The encoding is the default encoding of the font, usually `Standard`, `Expert`, or `Special`, as defined in the Adobe PPD file specification.

The version is the PostScript string definition that corresponds to the font version number.

The charset defines the available characters in the font, usually `Standard` or `Special`, as defined in the Adobe PPD file specification.

The status is the installation status of the font and must be either the word `ROM` or `Disk`.

Device fonts differ from fonts defined using the `#font` directive in that they are automatically associated with the current driver. Fonts defined using `#font` may be imported into the current driver using the `Font *` form of this directive.

See Also

`#font`

Group

Syntax

```
Group name
Group "name/text"
```

Examples

```
Group General
Group "InstallableOptions/Options Installed"
Group "Special/Vendor Options"
```

Description

The `Group` directive specifies the group for new `Option` directives. The name is any combination of letters, numbers, and the underscore and can be up to 40 characters in length. The names `General` and `InstallableOptions` are predefined for the standard Adobe UI keywords and for installable options, respectively.

If provided, the text can be any string up to 40 characters in length. If no text is provided, the name is used.

> **Note:**
>
> Because of certain API binary compatibility issues, CUPS limits the length of PPD group translation strings (text) to 40 characters, while the PPD specification allows for up to 80 characters.

See Also

`Choice`, `ColorModel`, `Cutter`, `Darkness`, `Duplex`, `Finishing`, `InputSlot`, `Installable`, `MediaType`, `Option`, `Resolution`, `UIConstraints`

HWMargins

Syntax

```
HWMargins left bottom right top
```

Examples

```
HWMargins 18 36 18 36
HWMargins 0.25in 0.5in 0.25in 0.5in
HWMargins 0.6cm 1.2cm 0.6cm 1.2cm
```

Description

The `HWMargins` directive specifies the current margins for `MediaSize` that follow. The left, bottom, right, and top margin values specify the printable margins.

See Also

`MediaSize`

InputSlot

Syntax

```
InputSlot position name
InputSlot position "name/text"
```

Examples

```
InputSlot 0 Auto
InputSlot 1 "Upper/Tray 1"
```

Description

The `InputSlot` directive adds a new choice to the `InputSlot` option. The position argument is a number from 0 to $2^{32}-1$ specifying the value that is placed in the `MediaPosition` attribute in the PostScript page device dictionary.

The name is any combination of letters, numbers, and the underscore and can be up to 40 characters in length.

If provided, the text can be any string up to 80 characters in length. If no text is provided, the name is used.

See Also

`Choice`, `ColorModel`, `Cutter`, `Darkness`, `Duplex`, `Finishing`, `Group`, `Installable`, `MediaType`, `Option`, `Resolution`, `UIConstraints`

Installable

Syntax

```
Installable name
Installable "name/text"
```

Examples

```
Installable EnvTray
Installable "Option1/Duplexer Installed"
```

Description

The `Installable` directive adds a new boolean option to the `InstallableOptions` group with a default value of `False`. The name is any combination of letters, numbers, and the underscore and can be up to 40 characters in length.

If provided, the text can be any string up to 80 characters in length. If no text is provided, the name is used.

ManualCopies

Syntax

```
ManualCopies boolean-value
```

Examples

```
ManualCopies no
ManualCopies yes
```

Description

The `ManualCopies` directive specifies whether copies need to be produced by the RIP filters. The default is `no`.

See Also

`Choice`, `ColorModel`, `Cutter`, `Darkness`, `Duplex`, `Finishing`, `Group`, `InputSlot`, `MediaType`, `Option`, `Resolution`, `UIConstraints`

Manufacturer

Syntax

```
Manufacturer "name"
```

Examples

```
Manufacturer "Foo"
Manufacturer "HP"
```

Description

The `Manufacturer` directive specifies the manufacturer name for the current driver. The name argument must conform to the manufacturer name requirements in the Adobe PPD file specification.

See Also

`ModelName`, `PCFileName`, `Version`

MaxSize

Syntax

```
MaxSize width length
```

Examples

```
MaxSize 36in 100ft
MaxSize 300cm 30m
```

Description

The `MaxSize` directive specifies the maximum width and length that is supported for custom page sizes.

See Also

`MinSize`, `VariablePaperSize`

MediaSize

Syntax

```
MediaSize name
```

Examples

```
MediaSize Letter
MediaSize A4
```

Description

The `MediaSize` directive adds the named size to the current printer driver using the current margins defined with the `HWMargins` directive. The name argument must match a media size defined using the `#media` directive.

See Also

`#media`, `HWMargins`

MediaType

Syntax

```
MediaType type name
MediaType type "name/text"
```

Examples

```
MediaType 0 Auto
MediaType 1 "Plain/Plain Paper"
```

Description

The `MediaType` directive adds a new choice to the `MediaType` option. The type argument is a number from 0 to $2^{32}-1$ specifying the value that is placed in the `cupsMediaType` attribute in the PostScript page device dictionary.

The name is any combination of letters, numbers, and the underscore and can be up to 40 characters in length. The name is placed in the `MediaType` attribute in the PostScript page device dictionary.

If provided, the text can be any string up to 80 characters in length. If no text is provided, the name is used.

See Also

`Choice`, `ColorModel`, `Cutter`, `Darkness`, `Duplex`, `Finishing`, `Group`, `InputSlot`, `Installable`, `Option`, `Resolution`, `UIConstraints`

MinSize

Syntax

```
MinSize width length
```

Examples

```
MinSize 4in 8in
MinSize 10cm 20cm
```

Description

The `MinSize` directive specifies the minimum width and length that is supported for custom page sizes.

See Also

`MaxSize`, `VariablePaperSize`

ModelName

Syntax

```
ModelName "name"
```

Examples

```
ModelName "Foo Laser Printer 2000"
ModelName "Colorific 123"
```

Description

The `ModelName` directive sets the printer name for the `ModelName`, `NickName`, and `ShortNickName` attributes for the printer driver. The name is any string of letters, numbers, spaces, and the characters ".", "/", "−", and "+" and should not begin with the manufacturer name since the PPD compiler will add this automatically for you. The maximum length of the name string is 31 characters to conform to the Adobe limits on the length of `ShortNickName`.

See Also

`Manufacturer`, `PCFileName`, `Version`

ModelNumber

Syntax

```
ModelNumber number
```

Examples

```
ModelNumber 123
ModelNumber ($PCL_PAPER_SIZE $PCL_PJL)
```

Description

The `ModelNumber` directive sets the `cupsModelNumber` attribute for the printer driver, which is often used by the printer driver filter to tailor its output for the current device. The number is any integer or bitwise OR of integers and constants that is appropriate for the printer driver filters.

A complete list of printer driver model number constants is available later in this appendix in the section titled, "Printer Driver ModelNumber Constants".

See Also

`DriverType`, `Filter`

Option

Syntax

```
Option name type section order
Option "name/text" type section order
```

Examples

```
Option Punch Boolean AnySetup 10
Option "fooFinish/Finishing Option" PickOne DocumentSetup 10
```

Description

The `Option` directive creates a new option in the current group, by default the `General` group. The name is any combination of letters, numbers, and the underscore and can be up to 40 characters in length.

If provided, the text can be any string up to 80 characters in length. If no text is provided, the name is used.

The type argument is one of the following keywords:

- `Boolean` − a true/false option
- `PickOne` − allows the user to pick one choice from a list
- `PickMany` − allows the user to pick zero or more choices from a list

The section argument is one of the following keywords:

- `AnySetup` − The option can be placed in either the DocumentSetup or PageSetup sections of the PostScript document
- `DocumentSetup` − The option must be placed in the DocumentSetup section of the PostScript document; this does not allow the option to be overridden on individual pages
- `ExitServer` − The option must be placed in a separate initialization job prior to the document (not used for raster printer drivers)
- `JCLSetup` − The option contains job control language commands and must be sent prior to the document using the `JCLBegin` and `JCLToPSInterpreter` attributes (not used for raster printer drivers)
- `PageSetup` − The option must be placed at the beginning of each page in the PostScript document
- `Prolog` − The option must be placed in the prolog section of the PostScript document; this is typically used to add special comments for high−end typesetters, but can also be used to add CUPS PostScript job ticket comments.

The order argument is a real number greater than or equal to 0.0 and is used to sort the printer commands from many options before sending them to the printer or RIP filter.

See Also

`Choice`, `ColorModel`, `Cutter`, `Darkness`, `Duplex`, `Finishing`, `Group`, `InputSlot`, `Installable`, `MediaType`, `Resolution`, `UIConstraints`

PCFileName

Syntax

```
PCFileName "filename.ppd"
```

Examples

```
PCFileName "foljt2k1.ppd"
PCFileName "deskjet.ppd"
```

Description

The `PCFileName` attribute specifies the name of the PPD file for the current driver. The filename argument must conform to the Adobe PPD file specification and can be no more than 8 filename characters plus the extension ".ppd".

See Also

`Manufacturer`, `ModelName`, `Version`

Resolution

Syntax

```
Resolution colorspace bits-per-color row-count row-feed row-step name
Resolution colorspace bits-per-color row-count row-feed row-step "name/text"
```

Examples

```
Resolution - 8 0 0 0 300dpi
Resolution k 8 0 0 0 "600x300dpi/600 DPI Grayscale"
```

Description

The Resolution directive creates a new Resolution option choice which sets the HWResolution, cupsBitsPerColor, cupsRowCount, cupsRowFeed, cupsRowStep, and optionally the cupsColorSpace page device dictionary attributes. The colorspace argument specifies a colorspace to use for the specified resolution and can be the hyphen (-) character to make no change to the selected color model or any keyword listed in the section titled, "Colorspace Keywords", to force the named colorspace.

The bits-per-color argument specifies the number of bits per color to generate when RIP'ing a job. The values 1, 2, 4, and 8 are currently supported by CUPS.

The row-count, row-feed, and row-step argument specify the driver-dependent values for the cupsRowCount, cupsRowFeed, and cupsRowStep attributes, respectively. Most drivers leave these attributes set to 0, but any number from 0 to $2^{32}-1$ is allowed.

The name argument must conform to the resolution naming conventions in the Adobe PPD file specification, either HHHdpi for symmetric resolutions or HHHxVVVdpi for asymmetric resolutions. The HHH and VVV in the examples represent the horizontal and vertical resolutions which must be positive integer values.

If provided, the text can be any string up to 80 characters in length. If no text is provided, the name is used.

See Also

Choice, ColorModel, Cutter, Darkness, Duplex, Finishing, Group, InputSlot, Installable, MediaType, Option, UIConstraints

SimpleColorProfile

Syntax

```
SimpleColorProfile resolution/mediatype density
    yellow-density red-density gamma
    red-adjust green-adjust blue-adjust
```

Examples

```
SimpleColorProfile -/- 100 100 200 1.0 0 0 0

SimpleColorProfile 360dpi/- 100 95 150 1.2 5 10 15

SimpleColorProfile 720dpi/Glossy 100 90 120 1.5 -5 5 10
```

Description

The `SimpleColorProfile` directive creates a matrix-based `ColorProfile` using values chosen with the `cupsprofile(1)` utility. The resolution and mediatype arguments specify the `Resolution` and `MediaType` choices which use the profile; the hyphen (`-`) is used to specify that any resolution or mediatype can be used with the profile.

The density argument specifies the linear density correction to apply to the color values ($P = d * 0.01 * p^g$) and is an integer greater than 0 and less than or equal to 100. A value 100 of disables density correction while lower values produce proportionately lighter output. The density value adjusts all color channels equally in all color modes.

The yellow-density argument specifies the density of the yellow channel when printing in grayscale or RGB mode and is an integer greater than 0 and less then or equal to 100. A value of 100 disables yellow density correction while lower values produce proportionately lighter output.

The red-density argument specifies the two-color density limit (e.g. C + M, C + Y, M + Y) when printing in grayscale or RGB mode and is an integer greater than 0 and less then or equal to 200. A value of 200 disables two-color density correction while lower values produce proportionately lighter output.

The gamma argument specifies the gamma correction to apply to the color values ($P = p^g$) and is a real number greater than 0. Values larger than 1 cause a general lightening of the print while values smaller than 1 cause a general darkening of the print. A value of 1 disables gamma correction.

The red-adjust, green-adjust, blue-adjust arguments specify the percentage of color to add or remove. Positive red-adjust values add magenta and negative values add yellow. Positive green-adjust values add cyan and negative values add yellow. Positive blue-adjust values add cyan and negative values add magenta. Values of 0 disable color adjustments.

See Also

`ColorProfile`

Throughput

Syntax

```
Throughput pages-per-minute
```

Examples

```
Throughput 1
Throughput 10
```

Description

The `Througput` directive sets the `Troughput` attribute for the current printer driver. The pages−per−minute argument is a positive integer representing the peak number of pages per minute that the printer is capable of producing. Use a value of 1 for printers that produce less than 1 page per minute.

UIConstraints

Syntax

```
UIConstraints "*Option1 *Option2"
UIConstraints "*Option1 Choice1 *Option2"
UIConstraints "*Option1 *Option2 Choice2"
UIConstraints "*Option1 Choice1 *Option2 Choice2"
```

Examples

```
UIConstraints "*Finishing *MediaType"
UIConstraints "*Option1 False *Duplex"
UIConstraints "*Duplex *MediaType Transparency"
UIConstraints "*Resolution 600dpi *ColorModel RGB"
```

Description

The `UIConstraints` directive adds a constraint between two options. Constraints inform the application when a user has chosen incompatible options. Each option name is preceded by the asterisk (`*`). If no choice is given for an option, then all choices *except* `False` and `None` will conflict with the other option and choice(s). Since the PPD compiler automatically adds reciprocal constraints (option A conflicts with option B, so therefore option B conflicts with option A), you need only specify the constraint once.

See Also

`Choice`, `ColorModel`, `Cutter`, `Darkness`, `Duplex`, `Finishing`, `Group`, `InputSlot`, `Installable`, `MediaType`, `Option`, `Resolution`

VariablePaperSize

Syntax

```
VariablePaperSize boolean-value
```

Examples

```
VariablePaperSize yes
VariablePaperSize no
```

Description

The `VariablePaperSize` directive specifies whether the current printer supports variable (custom) page sizes. When `yes` is specified, the PPD compiler will include the standard PPD attributes required to support custom page sizes.

See Also

`MaxSize`, `MinSize`

Version

Syntax

```
Version number
```

Examples

```
Version 1.0
Version 3.7
```

Description

The `Version` directive sets the `FileVersion` attribute in the PPD file and is also used for the `NickName` attribute. The number argument is a positive real number.

See Also

`Manufacturer`, `ModelName`, `PCFileName`

Standard Include Files

Table B−1 shows the standard include files which are provided with the DDK.

Include File	Description
<dymo.h>	Defines all of the CUPS Dymo sample driver constants.
<font.defs>	Defines all of the standard fonts which are included with ESP Ghostscript and the Apple PDF RIP.
<epson.h>	Defines all of the CUPS ESC/P sample driver constants.
<escp.h>	Defines all of the DDK ESC/P driver constants.
<hp.h>	Defines all of the CUPS HP−PCL sample driver constants.
<media.defs>	Defines all of the standard media sizes listed in Appendix B of the Adobe PostScript Printer Description File Format Specification.
<pcl.h>	Defines all of the DDK HP−PCL driver constants.
<raster.defs>	Defines all of the CUPS raster format constants.
<rtl.h>	Defines all of the CUPS HP−RTL sample driver constants.

Table B−1, Standard Include Files

Printer Driver ModelNumber Constants

The CUPS DDK and sample drivers use the `cupsModelNumber` attribute in the PPD file to tailor their output to the printer. The following sections describe the constants for each driver.

The CUPS Dymo Sample Driver (dymo)

The `dymo` driver supports the Dymo Labelwriter 300, 330, and 330 Turbo printers. Table B−2 lists the constants for the `ModelNumber` directive. `ModelNumber` values should be inserted by referencing only one of these constants.

Constant	Description
DYMO_3x0	Format output for the Dymo Labelwriter 300, 330, or 330 Turbo.

Table B−2, `dymo` driver constants

The CUPS ESC/P Sample Driver (epson)

The `epson` driver supports Epson and Okidata dot−matrix, Epson Stylus Color, and Epson Stylus Photo printers. Table B−3 lists the constants for the `ModelNumber` directive. `ModelNumber` values should be inserted by referencing only one of these constants.

Constant	Description
`EPSON_9PIN`	Epson and Okidata 9−pin dot−matrix printers
`EPSON_24PIN`	Epson and Okidata 24−pin dot−matrix printers
`EPSON_COLOR`	Older Epson Stylus Color printers that use the `ESC .` graphics command
`EPSON_PHOTO`	Older Epson Stylus Photo printers that use the `ESC .` graphics command
`EPSON_ICOLOR`	Newer Epson Stylus Color printers that use the `ESC i` graphics command
`EPSON_IPHOTO`	Newer Epson Stylus Photo printers that use the `ESC i` graphics command

Table B−3, `epson` driver constants

The DDK ESC/P Driver (escp)

The `escp` driver supports all Epson inkjet printers. Table B−4 lists the constants for the `ModelNumber` directive. `ModelNumber` values should be specified as the bitwise OR of one or more of these constants.

Constant	Description
`ESCP_MICROWEAVE`	Use microweave command?
`ESCP_STAGGER`	Are color jets staggered?
`ESCP_ESCK`	Use print mode command?
`ESCP_EXT_UNITS`	Use extended unit commands?
`ESCP_EXT_MARGINS`	Use extended margin command?
`ESCP_USB`	Send USB packet mode escape
`ESCP_PAGE_SIZE`	Use page size command
`ESCP_RASTER_ESCI`	Use `ESC i` graphics command
`ESCP_REMOTE`	Use remote mode commands
`ESCP_REMOTE_AC`	Use auto−cutter command
`ESCP_REMOTE_CO`	Use cutter−operation command
`ESCP_REMOTE_EX`	Use media−position command
`ESCP_REMOTE_MS`	Use media−size command
`ESCP_REMOTE_MT`	Use media−type command
`ESCP_REMOTE_PC`	Use paper−check command
`ESCP_REMOTE_PH`	Use paper−thickness command
`ESCP_REMOTE_PP`	Use paper−path command
`ESCP_REMOTE_SN0`	Use feed−sequence−0 command
`ESCP_REMOTE_SN1`	Use platten−gap command
`ESCP_REMOTE_SN2`	Use feed−sequence−2 command
`ESCP_REMOTE_SN6`	Use eject−delay command
`ESCP_REMOTE_FP`	Use print−position command

Table B−4, `escp` driver constants

The CUPS HP−PCL Sample Driver (hp)

The `hp` driver supports HP LaserJet and DeskJet printers. Table B−5 lists the constants for the `ModelNumber` directive. `ModelNumber` values should be inserted by referencing only one of these constants.

Constant	Description
`HP_LASERJET`	HP LaserJet printers supporting PCL 3, 4, or 5
`HP_DESKJET`	HP DeskJet printers supporting PCL 3 and using the simple color graphics command (`ESC * r # U`)
`HP_JET`	HP DeskJet printers supporting PCL3GUI and using the configure raster graphics command (`ESC * g # W`)

Table B−5, `hp` driver constants

The DDK HP−PCL Driver (pcl)

The `pcl` driver supports all HP LaserJet, DeskJet, and DesignJet printers. Table B−6 lists the constants for the `ModelNumber` directive. `ModelNumber` values should be specified as the bitwise OR of one or more of these constants.

Constant	Description
PCL_PAPER_SIZE	Use paper size command (ESC & l # A)
PCL_INKJET	Use inkjet commands
PCL_RASTER_END_COLOR	Use new end−raster command (ESC * r C)
PCL_RASTER_CID	Use configure−image−data command (ESC * v # W)
PCL_RASTER_CRD	Use configure−raster−data command (ESC * g # W)
PCL_RASTER_SIMPLE	Use simple−raster−color command (ESC * r # U)
PCL_RASTER_RGB24	Use 24−bit RGB mode
PCL_PJL	Use PJL commands
PCL_PJL_PAPERWIDTH	Use PJL PAPERWIDTH/LENGTH commands
PCL_PJL_HPGL2	Use PJL ENTER HPGL2 command
PCL_PJL_PCL3GUI	Use PJL ENTER PCL3GUI command
PCL_PJL_RESOLUTION	Use PJL SET RESOLUTION command

Table B−6, `pcl` driver constants

The CUPS HP−RTL Sample Driver

The `rtl` driver supports HP DesignJet printers and ENCAD plotters. Table B−7 lists the constants for the `ModelNumber` directive. `ModelNumber` values should be specified as the bitwise OR of one or more of these constants.

Constant	Description
`RTL_PJL`	Supports PJL commands
`RTL_PJL_EXT`	Supports extended PJL commands
`RTL_END_COLOR`	Supports new end−raster command (`ESC * r C`)
`RTL_CID`	Supports configure−image−data command (`ESC * v # W`)
`RTL_CRD`	Supports configure−raster−data command (`ESC * g # W`)
`RTL_CMYK`	Supports CMYK graphics
`RTL_ENCAD`	Supports ENCAD quality modes

Table B−7, `rtl` driver constants

Color Keywords

The PPD compiler defines two types of color keywords: colorspace and color order. The following sections list the supported keywords for each type.

Colorspace Keywords

The following colorspace keywords are recognized:

- `w` − Luminance
- `rgb` − Red, green, blue
- `rgba` − Red, green, blue, alpha
- `k` − Black
- `cmy` − Cyan, magenta, yellow
- `ymc` − Yellow, magenta, cyan *
- `cmyk` − Cyan, magenta, yellow, black
- `ymck` − Yellow, magenta, cyan, black *
- `kcmy` − Black, cyan, magenta, yellow *
- `kcmycm` − Black, cyan, magenta, yellow, light−cyan, light−magenta *
- `gmck` − Gold, magenta, yellow, black *
- `gmcs` − Gold, magenta, yellow, silver *
- `white` − White ink (as black) *
- `gold` − Gold foil *
- `silver` − Silver foil *
- `ciexyz` − CIE XYZ
- `cielab` − CIE Lab
- `icc1` − ICC−based, 1 color *
- `icc2` − ICC−based, 2 colors *
- `icc3` − ICC−based, 3 colors *
- `icc4` − ICC−based, 4 colors *
- `icc5` − ICC−based, 5 colors *
- `icc6` − ICC−based, 6 colors *
- `icc7` − ICC−based, 7 colors *
- `icc8` − ICC−based, 8 colors *
- `icc9` − ICC−based, 9 colors *
- `icca` − ICC−based, 10 colors *
- `iccb` − ICC−based, 11 colors *
- `iccc` − ICC−based, 12 colors *
- `iccd` − ICC−based, 13 colors *
- `icce` − ICC−based, 14 colors *
- `iccf` − ICC−based, 15 colors *

* = This colorspace is not supported by the current Apple RIP filters and should not be used when developing printer drivers for MacOS X 10.2 or 10.3.

Color Order Keywords

The following color order keywords are recognized:

- `chunked` or `chunky` − Color values are passed together on a line as RGB RGB RGB RGB
- `banded` − Color values are passed separately on a line as RRRR GGGG BBBB *
- `planar` − Color values are passed separately on a page as RRRR RRRR RRRR ... GGGG GGGG GGGG ... BBBB BBBB BBBB *

  * = This color order is not supported by the current Apple RIP filters and should not be used when developing printer drivers for MacOS X 10.2 or 10.3.

C – CUPS Driver API Reference

This appendix provides a reference for the CUPS driver API included with the CUPS DDK.

Header File

The CUPS driver API provides a single header file called *driver.h*. You include it using the following preprocessor command:

```
#include <cups/driver.h>
```

Library

The CUPS driver API is provided in a single library called `cupsdriver`. The driver API also depends upon the CUPS imaging API and CUPS API libraries. A typical link command will look like the following:

```
cc -o rastertomyprinter rastertomyprinter.o -lcupsdriver -lcupsimage -lcups -lm ENTER
```

Contents

- Functions
- Structures
- Types
- Variables

Functions

- cupsCMYKDelete()
- cupsCMYKDoBlack()
- cupsCMYKDoCMYK()
- cupsCMYKDoGray()
- cupsCMYKDoRGB()
- cupsCMYKLoad()
- cupsCMYKNew()
- cupsCMYKSetBlack()
- cupsCMYKSetCurve()
- cupsCMYKSetGamma()
- cupsCMYKSetInkLimit()
- cupsCMYKSetLtDk()
- cupsCheckBytes()
- cupsCheckValue()
- cupsDitherDelete()
- cupsDitherLine()
- cupsDitherNew()
- cupsFindAttr()
- cupsLutDelete()
- cupsLutLoad()
- cupsLutNew()
- cupsPackHorizontal()
- cupsPackHorizontal2()
- cupsPackHorizontalBit()
- cupsPackVertical()
- cupsRGBDelete()
- cupsRGBDoGray()
- cupsRGBDoRGB()
- cupsRGBLoad()
- cupsRGBNew()

cupsCMYKDelete()

Description

Delete a color separation.

Syntax

```
void
cupsCMYKDelete(
    cups_cmyk_t * cmyk);
```

Arguments

Name	Description
cmyk	Color separation

Returns

Nothing.

cupsCMYKDoBlack()

Description

Do a black separation...

Syntax

```
void
cupsCMYKDoBlack(
    const cups_cmyk_t * cmyk,
    const unsigned char * input,
    short * output,
    int num_pixels);
```

Arguments

Name	Description
cmyk	Color separation
input	Input grayscale pixels
output	Output Device−N pixels
num_pixels	Number of pixels

Returns

Nothing.

cupsCMYKDoCMYK()

Description

Do a CMYK separation...

Syntax

```
void
cupsCMYKDoCMYK(
    const cups_cmyk_t * cmyk,
    const unsigned char * input,
    short * output,
    int num_pixels);
```

Arguments

Name	Description
cmyk	Color separation
input	Input grayscale pixels
output	Output Device−N pixels
num_pixels	Number of pixels

Returns

Nothing.

cupsCMYKDoGray()

Description

Do a grayscale separation...

Syntax

```
void
cupsCMYKDoGray(
    const cups_cmyk_t * cmyk,
    const unsigned char * input,
    short * output,
    int num_pixels);
```

Arguments

Name	Description
cmyk	Color separation
input	Input grayscale pixels
output	Output Device−N pixels
num_pixels	Number of pixels

Returns

Nothing.

cupsCMYKDoRGB()

Description

Do an sRGB separation...

Syntax

```
void
cupsCMYKDoRGB(
    const cups_cmyk_t * cmyk,
    const unsigned char * input,
    short * output,
    int num_pixels);
```

Arguments

Name	Description
cmyk	Color separation
input	Input grayscale pixels
output	Output Device−N pixels
num_pixels	Number of pixels

Returns

Nothing.

cupsCMYKLoad()

Description

Load a CMYK color profile from PPD attributes.

Syntax

```
cups_cmyk_t *
cupsCMYKLoad(
    ppd_file_t * ppd,
    const char * colormodel,
    const char * media,
    const char * resolution);
```

Arguments

Name	Description
ppd	PPD file
colormodel	ColorModel value
media	MediaType value
resolution	Resolution value

Returns

CMYK color separation

cupsCMYKNew()

Description

Create a new CMYK color separation.

Syntax

```
cups_cmyk_t *
cupsCMYKNew(
    int num_channels);
```

Arguments

Name	Description
num_channels	Number of color components

Returns

New CMYK separation or NULL

cupsCMYKSetBlack()

Description

Set the transition range for CMY to black.

Syntax

```
void
cupsCMYKSetBlack(
    cups_cmyk_t * cmyk,
    float lower,
    float upper);
```

Arguments

Name	Description
cmyk	CMYK color separation
lower	No black ink
upper	Only black ink

Returns

Nothing.

cupsCMYKSetCurve()

Description

Set a color transform curve using points.

Syntax

```
void
cupsCMYKSetCurve(
    cups_cmyk_t * cmyk,
    int channel,
    int num_xypoints,
    const float * xypoints);
```

Arguments

Name	Description
cmyk	CMYK color separation
channel	Color channel
num_xypoints	Number of X,Y points
xypoints	X,Y points

Returns

Nothing.

cupsCMYKSetGamma()

Description

Set a color transform curve using gamma and density.

Syntax

```
void
cupsCMYKSetGamma(
    cups_cmyk_t * cmyk,
    int channel,
    float gamval,
    float density);
```

Arguments

Name	Description
cmyk	CMYK color separation
channel	Ink channel
gamval	Gamma correction
density	Maximum density

Returns

Nothing.

cupsCMYKSetInkLimit()

Description

Set the limit on the amount of ink.

Syntax

```
void
cupsCMYKSetInkLimit(
    cups_cmyk_t * cmyk,
    float limit);
```

Arguments

Name	Description
cmyk	CMYK color separation
limit	Limit of ink

Returns

Nothing.

cupsCMYKSetLtDk()

Description

Set light/dark ink transforms.

Syntax

```
void
cupsCMYKSetLtDk(
    cups_cmyk_t * cmyk,
    int channel,
    float light,
    float dark);
```

Arguments

Name	Description
cmyk	CMYK color separation
channel	Dark ink channel (+1 for light)
light	Light ink only level
dark	Dark ink only level

Returns

Nothing.

cupsCheckBytes()

Description

Check to see if all bytes are zero.

Syntax

```
int
cupsCheckBytes(
    const unsigned char * bytes,
    int length);
```

Arguments

Name	Description
bytes	Bytes to check
length	Number of bytes to check

Returns

1 if they match

cupsCheckValue()

Description

Check to see if all bytes match the given value.

Syntax

```
int
cupsCheckValue(
    const unsigned char * bytes,
    int length,
    const unsigned char value);
```

Arguments

Name	Description
bytes	Bytes to check
length	Number of bytes to check
value	Value to check

Returns

1 if they match

cupsDitherDelete()

Description

Free a dithering buffer. Returns 0 on success, −1 on failure.

Syntax

```
void
cupsDitherDelete(
    cups_dither_t * d);
```

Arguments

Name	Description
d	Dithering buffer

Returns

Nothing.

cupsDitherLine()

Description

Dither a line of pixels...

Syntax

```
void
cupsDitherLine(
    cups_dither_t * d,
    const cups_lut_t * lut,
    const short * data,
    int num_channels,
    unsigned char * p);
```

Arguments

Name	Description
d	Dither data
lut	Lookup table
data	Separation data
num_channels	Number of components
p	Pixels

Returns

Nothing.

cupsDitherNew()

Description

Create an error−diffusion dithering buffer.

Syntax

```
cups_dither_t *
cupsDitherNew(
    int width);
```

Arguments

Name	Description
width	Width of output in pixels

Returns

New state array

cupsFindAttr()

Description

Find a PPD attribute based on the colormodel, media, and resolution.

Syntax

```
ppd_attr_t *
cupsFindAttr(
    ppd_file_t * ppd,
    const char * name,
    const char * colormodel,
    const char * media,
    const char * resolution,
    char * spec,
    int specsize);
```

Arguments

Name	Description
ppd	PPD file
name	Attribute name
colormodel	Color model
media	Media type
resolution	Resolution
spec	Final selection string
specsize	Size of string buffer

Returns

Matching attribute or NULL

cupsLutDelete()

Description

Free the memory used by a lookup table.

Syntax

```
void
cupsLutDelete(
    cups_lut_t * lut);
```

Arguments

Name	Description
lut	Lookup table to free

Returns

Nothing.

cupsLutLoad()

Description

Load a LUT from a PPD file.

Syntax

```
cups_lut_t *
cupsLutLoad(
    ppd_file_t * ppd,
    const char * colormodel,
    const char * media,
    const char * resolution,
    const char * ink);
```

Arguments

Name	Description
ppd	PPD file
colormodel	Color model
media	Media type
resolution	Resolution
ink	Ink name

Returns

New lookup table

cupsLutNew()

Description

Make a lookup table from a list of pixel values. Returns a pointer to the lookup table on success, NULL on failure.

Syntax

```
cups_lut_t *
cupsLutNew(
    int num_values,
    const float * values);
```

Arguments

Name	Description
num_values	Number of values
values	Lookup table values

Returns

New lookup table

cupsPackHorizontal()

Description

Pack pixels horizontally...

Syntax

```
void
cupsPackHorizontal(
    const unsigned char * ipixels,
    unsigned char * obytes,
    int width,
    const unsigned char clearto,
    const int step);
```

Arguments

Name	Description
ipixels	Input pixels
obytes	Output bytes
width	Number of pixels
clearto	Initial value of bytes
step	Step value between pixels

Returns

Nothing.

cupsPackHorizontal2()

Description

Pack 2−bit pixels horizontally...

Syntax

```
void
cupsPackHorizontal2(
    const unsigned char * ipixels,
    unsigned char * obytes,
    int width,
    const int step);
```

Arguments

Name	Description
ipixels	Input pixels
obytes	Output bytes
width	Number of pixels
step	Stepping value

Returns

Nothing.

cupsPackHorizontalBit()

Description

Pack pixels horizontally by bit...

Syntax

```
void
cupsPackHorizontalBit(
    const unsigned char * ipixels,
    unsigned char * obytes,
    int width,
    const unsigned char clearto,
    const unsigned char bit);
```

Arguments

Name	Description
ipixels	Input pixels
obytes	Output bytes
width	Number of pixels
clearto	Initial value of bytes
bit	Bit to check

Returns

Nothing.

cupsPackVertical()

Description

Pack pixels vertically...

Syntax

```
void
cupsPackVertical(
    const unsigned char * ipixels,
    unsigned char * obytes,
    int width,
    const unsigned char bit,
    const int step);
```

Arguments

Name	Description
ipixels	Input pixels
obytes	Output bytes
width	Number of input pixels
bit	Output bit
step	Number of bytes between columns

Returns

Nothing.

cupsRGBDelete()

Description

Delete a color separation.

Syntax

```
void
cupsRGBDelete(
    cups_rgb_t * rgbptr);
```

Arguments

Name	Description
rgbptr	Color separation

Returns

Nothing.

cupsRGBDoGray()

Description

Do a grayscale separation...

Syntax

```
void
cupsRGBDoGray(
    cups_rgb_t * rgbptr,
    const unsigned char * input,
    unsigned char * output,
    int num_pixels);
```

Arguments

Name	Description
rgbptr	Color separation
input	Input grayscale pixels
output	Output Device−N pixels
num_pixels	Number of pixels

Returns

Nothing.

cupsRGBDoRGB()

Description

Do a RGB separation...

Syntax

```
void
cupsRGBDoRGB(
    cups_rgb_t * rgbptr,
    const unsigned char * input,
    unsigned char * output,
    int num_pixels);
```

Arguments

Name	Description
rgbptr	Color separation
input	Input RGB pixels
output	Output Device−N pixels
num_pixels	Number of pixels

Returns

Nothing.

cupsRGBLoad()

Description

Load a RGB color profile from a PPD file.

Syntax

```
cups_rgb_t *
cupsRGBLoad(
    ppd_file_t * ppd,
    const char * colormodel,
    const char * media,
    const char * resolution);
```

Arguments

Name	Description
ppd	PPD file
colormodel	Color model
media	Media type
resolution	Resolution

Returns

New color profile

cupsRGBNew()

Description

Create a new RGB color separation.

Syntax

```
cups_rgb_t *
cupsRGBNew(
    int num_samples,
    cups_sample_t * samples,
    int cube_size,
    int num_channels);
```

Arguments

Name	Description
num_samples	Number of samples
samples	Samples
cube_size	Size of LUT cube
num_channels	Number of color components

Returns

New color separation or NULL

Structures

- cups_cmyk_s
- cups_dither_s
- cups_lut_s
- cups_rgb_s
- cups_sample_s

cups_cmyk_s

Description

Simple CMYK lookup table

Definition

```
struct cups_cmyk_s
{
  unsigned char black_lut[256];
  short * channels[CUPS_MAX_CHAN];
  unsigned char color_lut[256];
  int ink_limit;
  int num_channels;
};
```

Members

Name	Description
black_lut[256]	Black generation LUT
channels[CUPS_MAX_CHAN]	Lookup tables
color_lut[256]	Color removal LUT
ink_limit	Ink limit
num_channels	Number of components

cups_dither_s

Description

Dithering State

Definition

```
struct cups_dither_s
{
  int errors[96];
  int row;
  int width;
};
```

Members

Name	Description
errors[96]	Error values
row	Current row
width	Width of buffer

cups_lut_s

Description

Lookup Table for Dithering

Definition

```
struct cups_lut_s
{
  int error;
  short intensity;
  short pixel;
};
```

Members

Name	Description
error	Error from desired value
intensity	Adjusted intensity
pixel	Output pixel value

cups_rgb_s

Description

Color separation lookup table

Definition

```
struct cups_rgb_s
{
  unsigned char black[CUPS_MAX_RGB];
  int cache_init;
  unsigned char **** colors;
  int cube_index[256];
  int cube_mult[256];
  int cube_size;
  int num_channels;
  unsigned char white[CUPS_MAX_RGB];
};
```

Members

Name	Description
black[CUPS_MAX_RGB]	Cached black (sRGB = 0,0,0)
cache_init	Are cached values initialized?
colors	4−D array of sample values
cube_index[256]	Index into cube for a given sRGB value
cube_mult[256]	Multiplier value for a given sRGB value
cube_size	Size of color cube (2−N) on a side
num_channels	Number of colors per sample
white[CUPS_MAX_RGB]	Cached white (sRGB = 255,255,255)

cups_sample_s

Description

Color sample point

Definition

```
struct cups_sample_s
{
  unsigned char colors[CUPS_MAX_RGB];
  unsigned char rgb[3];
};
```

Members

Name	Description
`colors[CUPS_MAX_RGB]`	Color values
`rgb[3]`	sRGB values

Types

- cups_cmyk_t
- cups_dither_t
- cups_lut_t
- cups_rgb_t
- cups_sample_t

cups_cmyk_t

Description

Simple CMYK lookup table

Definition

```
typedef struct cups_cmyk_s cups_cmyk_t;
```

cups_dither_t

Description

Dithering State

Definition

```
typedef struct cups_dither_s cups_dither_t;
```

cups_lut_t

Description

Lookup Table for Dithering

Definition

```
typedef struct cups_lut_s cups_lut_t;
```

cups_rgb_t

Description

Color separation lookup table

Definition

```
typedef struct cups_rgb_s cups_rgb_t;
```

cups_sample_t

Description

Color sample point

Definition

```
typedef struct cups_sample_s cups_sample_t;
```

Variables

- cups_scmy_lut[256]
- cups_srgb_lut[256]

cups_scmy_lut[256]

Description

sRGB gamma lookup table (inverted)

Definition

```
extern const unsigned char cups_scmy_lut[256];
```

cups_srgb_lut[256]

Description

sRGB gamma lookup table

Definition

```
extern const unsigned char cups_srgb_lut[256];
```

D – Man Pages

This appendix provides copies of all of the man pages included with the CUPS DDK.

Man Page Index

- cupsprofile(1)
- ppdc(1)
- ppdcfile(5)
- ppdi(1)
- ppdpo(1)

cupsprofile(1)

Name

cupsprofile − cups simple profiling tool

Synopsis

cupsprofile

Description

cupsprofile creates simple color profiles for CUPS−based drivers using a manual four−pass process. The first pass optimizes the density for the current media. The second pass optimizes the gamma correction for the current media and densities. The third pass optimizes the primary colors for the current media, densities, and gamma correction. The fourth and final pass prints a confirmation/validation page using all of the values entered.

See Also

ppdc(1), ppdi(1), ppdpo(1), ppdcfile(5), CUPS Driver Developer Kit Manual.

Copyright

Copyright 1993−2004 by Easy Software Products, All Rights Reserved.

ppdc(1)

Name

ppdc − cups ppd compiler

Synopsis

ppdc [−I *include−directory*] [−c *message−catalog*] [−d *output−directory*] [−l *language*] [−v] *source−file*

Description

ppdc compiles PPDC source files into one or more PPD files.

The *−I* option specifies an alternate include directory; multiple *−I* options can be supplied to add additional directories.

The *−c* option specifies a message catalog file in GNU gettext source format (filename.po) to be used for localization.

The *−d* option specifies the output directory for PPD files. The default output directory is "ppd".

The *−l* option specifies the language to use when localizing the PPD file(s). The default language is "en" (English).

The *−v* option provides more verbose output, basically a running status of which files are being loaded or written.

See Also

cupsprofile(1), ppdi(1), ppdpo(1), ppdcfile(5), CUPS Driver Developer Kit Manual.

Copyright

Copyright 1993−2004 by Easy Software Products, All Rights Reserved.

ppdcfile(5)

Name

ppdcfile − cups ppd compiler source file format

Description

The CUPS PPD compiler reads meta files that contain descriptions of one or more PPD files to be generated by *ppdc*. This man page provides a quick reference to the supported keywords and should be used in conjunction with the CUPS Driver Development Kit Developers Manual.

The source file format is plain ASCII text that can be edited using your favorite text editor. Comments are supported using the C (/* ... */) and C++ (// ...) comment mechanisms.

More to follow...

See Also

cupsprofile(1), ppdc(1), ppdi(1), ppdpo(1), CUPS Driver Developer Kit Manual.

Copyright

Copyright 1993−2004 by Easy Software Products, All Rights Reserved.

ppdi(1)

Name

ppdi − import ppd files

Synopsis

ppdi [−I *include−directory*] [−o *source−file*] *ppd−file* [*ppd−file2 ... ppd−fileN*]

Description

ppdi imports one or more PPD files into a PPD compiler source file. Multiple languages of the same PPD file are merged into a single printer definition to facilitate accurate changes for all localizations.

The *−o* option specifies the PPD source file to update. If the source file does not exist, a new source file is created. Otherwise the existing file is merged with the new PPD file(s) on the command−line. If no source file is specified, the filename "ppdi.drv" is used.

See Also

cupsprofile(1), ppdc(1), ppdpo(1), ppdcfile(5), CUPS Driver Developer Kit Manual.

Copyright

Copyright 1993−2004 by Easy Software Products, All Rights Reserved.

ppdpo(1)

Name

ppdpo − cups message catalog generator

Synopsis

ppdpo [−I *include−directory*] [−o *output−file*] *source−file*

Description

ppdpo extracts UI strings from PPDC source files and writes them in a GNU gettext format message catalog source file for translation.

The *−I* option specifies an alternate include directory; multiple *−I* options can be supplied to add additional directories.

The *−o* option specifies the output file.

The *−v* option provides additional progress information.

See Also

cupsprofile(1), ppdc(1), ppdi(1), ppdcfile(5), CUPS Driver Developer Kit Manual.

Copyright

Copyright 1993−2004 by Easy Software Products, All Rights Reserved.

E − Release Notes

This appendix provides a description of the changes in each release of the CUPS DDK.

Changes in v1.0rc1

- Much improved documentation.
- Added support for cupsAllXY and cupsAllGamma CMYK transform attributes.
- Updated ESC/P driver to support both the old ESC . and new ESC i graphics commands.
- The ESC/P driver no longer uses the individual remote mode model number bits, but instead relies on the CUPS attributes instead.
- The ESC/P driver now supports automatic paper feed value computation.
- The ESC/P driver now uses a more intuitive value for the row count.
- Added the CUPS simple profile generator, cupsprofile.
- Added driver types for the CUPS sample drivers.
- Reorganized the build system and directory structure for DDK components.
- Changed the default PCL media type to "PLAIN" instead of "Plain".
- The PCL driver did not set the default PPD options.

Changes in v0.20

- First public release.
- Updated dither lookup code to support density values beyond 2.0.
- Added PCL command filter for cleaning the print heads.
- The PCL driver did not support grayscale/black output with Mode 10 compression.

- The PCL driver did not support 7−color modes.

Changes in v0.15

- The cupsCheckValue() function did its comparisons incorrectly, causing missing lines.
- The PPD compiler internally limited the size of code in PPD files to 1024 bytes; this has been increased to 10240 bytes (10k)
- The ESC/P driver did not support the softweaved modes properly.
- The PCL driver did not initialize the seed buffer correctly for mode 3 compression, causing spooky prints on PCL color devices in RGB mode.
- The PCL driver did not set the DISPLAY variable in its PJL JOB command, so some printers did not display the job info on the printer's LCD.
- Initial prototype of GUI development environment (ppddevel).
- First draft of PPDC manual.

Changes in v0.14

- Improved the performance of the ppdi utility.
- The ppdc utility now converts the PCFileName string to lowercase when creating PPD files.
- Added new "Fonts *" directive to include base fonts; otherwise PS PPD files would get non−PS base fonts.
- Added new "CustomMedia" directive for non−standard media sizes/commands.
- Generated PPD files now use attributes instead of the standard values if defined (e.g. PSVersion's, Product's, etc.)
- The advanced PCL driver now uses non−standard cupsXYZ option names for all finishing options to avoid conflicts with HP and Lexmark PPD plug−ins on OSX.
- The advanced ESC/P driver now sets the correct top margin and position.

Changes in v0.13

- Added the advanced PCL driver.
- The cupsCMYKLoad() function looked for the cupsBlackGeneration attribute instead of cupsBlackGamma for the black gamma correction and density lookup table.
- Fixed some spots where the ppdcDriver class did not localize the UI strings using the current message catalog.
- UIConstraints for Custom options now correctly map to NonUIConstraints in the output PPD file.

Changes in v0.12

- Added hooks for cut method and pressure on high−end EPSON Stylus Pro models.
- The CMYK LUT code incorrectly required the cupsBlackGeneration attribute.

Changes in v0.11

- Added hooks for keeping track of the driver "type" so that known drivers (ESC/P and PCL) can have meaningful ModelNumber values in the driver info/source files when saved.
- Now support localization via GNU gettext−like .po files.
- Changed the names of all CMYK lookup table attributes to be shorter and less specific to the CMYK code.

- Fixed the CMYK lookup table XY code − need to do a reverse transform from the provided data to get the proper results.
- The Lab color management code now generates 8−bit N−channel data for use with the CMYK lookup table stuff. This should make profiling a lot easier.
- The ESC/P driver now uses more PPD attributes for printer−specific stuff.

Changes in v0.10

- Initial private release.