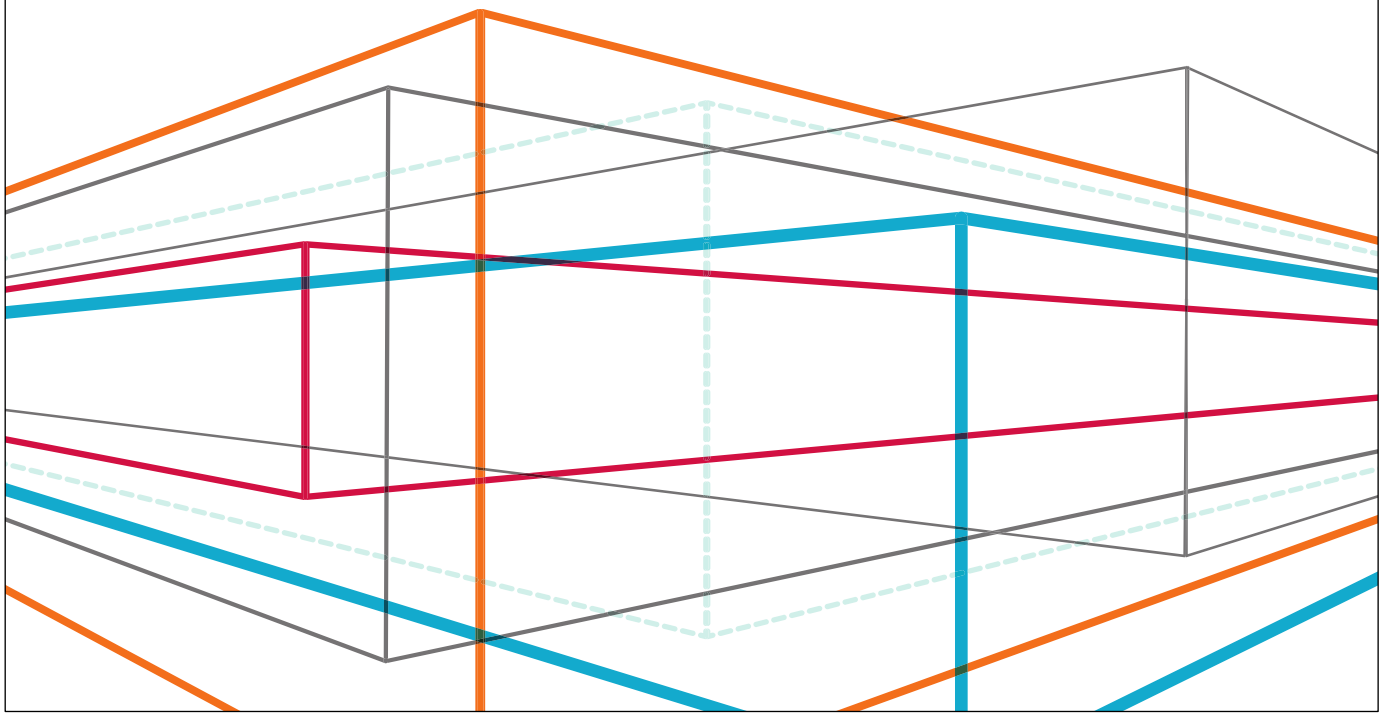
Technical Paper:
Zebra ZXP Series 3®
Linux Printer Driver

ZEBRA TECHNOLOGIES

SEE MORE. DO MORE.



Contents

Zebra ZXP Series 3 Printer Driver for Linux

Introduction

Purpose of the Document

The purpose of this document is to describe the technical implementation of the Zebra ZXP Series 3 and Series 1 Printer Driver for Linux.

(***Note:*** This Linux driver implementation supports Zebra ZXP Series 3 and ZXP Series 1® Card Printers only. Contact Zebra for Linux printer driver implementation details for other Zebra printers.)

System Environment

Hardware:

- Printer: ZXP Series 3 or ZXP Series 1 Printer
- Communication Interface: USB, Ethernet

Software:

- Operating System: Ubuntu Linux, Red Hat Linux (RHEL)

History of Common UNIX Printing System (CUPS)

Michael Sweet, who owned Easy Software Products, started developing CUPS in 1997. The first public betas appeared in 1999. The original design of CUPS used a Line Printer Daemon or LPD protocol implementation, but due to limitations in LPD and vendor incompatibilities, the Internet Printing Protocol (IPP) was chosen instead. CUPS was quickly adopted as the default printing system for most Linux distributions. In March 2002, Apple Inc. adopted CUPS as the printing system for Mac OS X 10.2. In February 2007, Apple Inc. hired chief developer, Michael Sweet, and purchased the CUPS source code.

Basic Understanding of CUPS

CUPS Driver Implementation

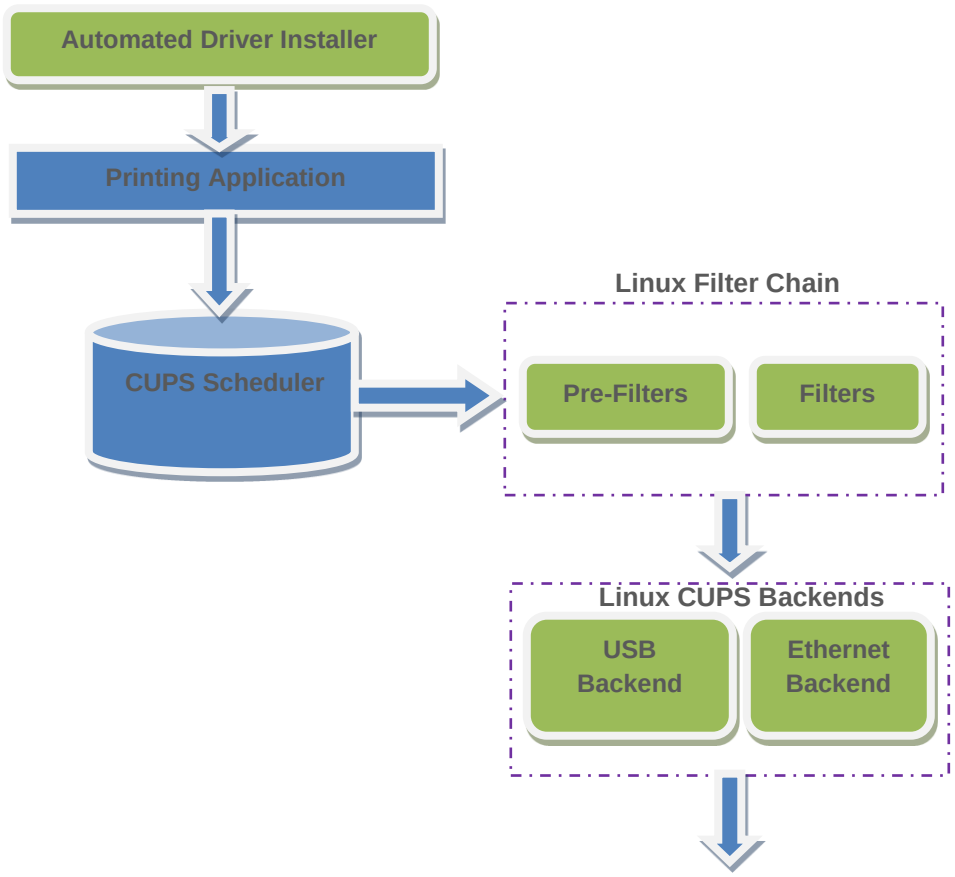
CUPS is a modular printing system for a UNIX-like computer operating system. A computer running CUPS is a host that accepts print jobs from client computers, processes them, and sends them to the appropriate printer. The architecture consists of a printer spooler, a filter system that converts the print data to a format that the printer understands, and a backend system that sends this data to the print device.

CUPS uses the Internet Printing Protocol (IPP) for managing print jobs and queues. It also supports the traditional command line interfaces for System V and Printing systems, and LPD protocol. It provides a mechanism that allows print jobs to be sent to printers in a standard process.

- Print data from an application goes to the printer scheduler, which sends the data to the filter chain.
- The filter chain contains one or more pre-filters, which process the data and send the data to the next filter.
- This filter processes the data and converts the print job into a format a specific printer understands.
- Finally, the filter system passes the data to the CUPS Backend system, which sends the print data to the device or network connection.

A CUPS driver implementation allows printer manufacturers and driver developers to easily create unique or specific drivers that work natively on the print server. Because processing occurs on the server, it allows for easier network-based printing than with other UNIX printing systems. With the addition of Samba, users can address printers on remote Windows computers, and generic PostScript drivers can be used for network based printing.

Architecture of the CUPS-Based Linux Printer Driver developed for the Zebra Printers



CUPS Scheduler

The CUPS scheduler implements IPP over HTTP/1.1. A helper application (cups-lpd) converts LPD requests to IPP. The scheduler also provides a web-based interface for managing print jobs, the configuration of the server, and for documentation about CUPS itself.

An authorization module controls which IPP and HTTP messages pass through the system. Once the IPP/HTTP packets are authorized, they are sent to the client module, which listens for and processes incoming connections. The client module also executes external CGI programs, as needed, to support web-based printers, classes, job status monitoring, and administration. Once this module has processed its requests, it sends them to the IPP module, which performs Uniform Resource Identifier (URI) validation to prevent a client from sidestepping any access controls or authentication on the HTTP server. The URI is a text string that indicates a name or address that refers to an abstract or physical resource on a network. The table below is an example of the URI used with a Zebra ZXP Series 3 printer:

Zebra ZXP Series 3 Printer	URI
USB	zxp3usb://ZXP11/Zebra-ZXP31-USB-Printer-Serial=123456478
Ethernet	zxp3socket://<IP Address>:9100

The scheduler allows for classes of printers. Applications send requests to groups of printers in a class, allowing the scheduler to direct the job to the first available printer in that class. A jobs module manages print jobs, sending them to the filter and backend processes for final conversion and printing, as well as monitoring the status messages from those processes.

The CUPS scheduler utilizes a configuration module, which parses configuration files, initializes CUPS data structures, and starts and stops the CUPS program. The configuration module stops CUPS services during configuration file processing, and then restarts the service when processing is complete.

A logging module handles the logging of scheduler events for access, error, and page log files. The main module handles timeouts and the dispatch of I/O requests for client connections, watches for signals, handles child process errors and exits, and reloads the server configuration files as needed.

Other modules used by the scheduler include:

- The Multipurpose Internet Mail Extensions or MIME module handles a MIME type and conversion database used in the filtering process to convert print data to a format suitable for a print device.
- A PPD module handles a list of Postscript Printer Description (PPD) files.
- A devices module manages a list of devices available in the system.
- A printer module manages printers and PPDs within the CUPS system.

CUPS Filters

CUPS allows users to send different data to the CUPS server, to convert the data into a format the printer understands, and to print. CUPS can process a variety of data formats on the print server. It converts the print job data into the final language/format of the printer via a series of filters.

In this implementation, the CUPS filter converts the document from raster format to a Card printer family specific format. In this case, the conversion embeds ZXP Series 3 specific printer supported commands, planar/pixel separation, and compression/encryption. Any image rotation, sharpening, image processing operations like brightness, contrast, gamma, etc., happens here.

The filtering process works by taking input data pre-formatted with six arguments:

- the job ID of the print job
- the username
- the job name
- the number of copies to print
- any print options
- the filename (though this is unnecessary if it has been redirected from standard input)

CUPS determines the type of data that is being input and the filter to be used through the use of the MIME databases. For instance, image data is detected and processed through a particular filter, while HTML data is detected and processed through another filter.

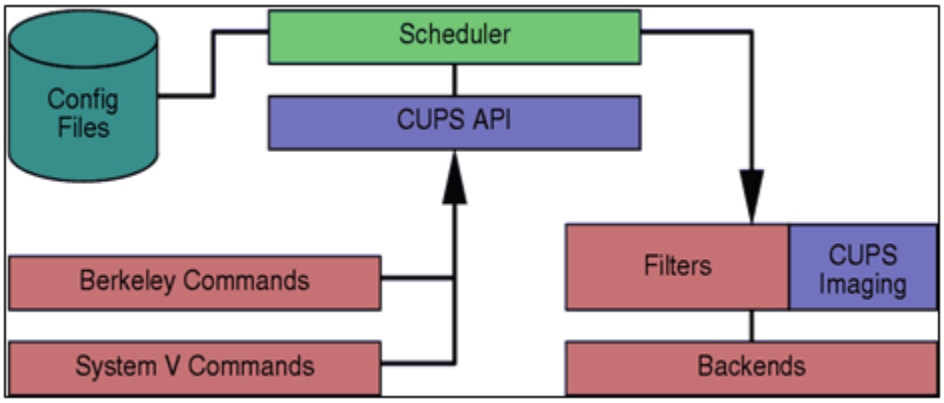
A generic CUPS converts supplied data either into PostScript data or directly into raster data. A pre-filter may be applied to convert PostScript data to raster. After the pre-filtering is complete, the data is either sent directly to a CUPS backend or another intermediary CUPS-raster format. In the case of the ZXP Series 3 driver implementation, this intermediary raster format passes it on to a final filter, which converts the raster data to a Card printer-specific format.

CUPS Backends

The backends are the ways in which CUPS sends data to printers. There are several backends available for CUPS: parallel, serial, and USB ports, as well as network backends that operate via the IPP, JetDirect (AppSocket), LPD, and Server Message Block (SMB) protocols.

The backend handles communications with the printer, sends print data from the last filter to the printer, and relays back-channel data from the printer to the upstream filters. CUPS includes backend programs for common direct-connect interfaces and network protocols. For USB printers, the USB module provided by CUPS is the backend.

The backend filter communicates with the printer to query the status, and to send control commands using CUPS APIs. They provide bi-directional communication support for the printer. The Zebra customized CUPS backend receives the errors and status messages from the printer, and sends these messages to the Error Handling module. The standard CUPS backend does not have Error Handling modules, and is therefore not suitable for the bi-directional print commands necessary in a Card specific print job.



CUPS Postscript Printing Document (PPD)

A PPD also contains the PostScript code (commands) used to invoke features for the print job. As such, PPDs function as drivers for all PostScript printers, by providing a unified interface for the printer's capabilities and features. For example, a generic PPD file for all models of Zebra ZXP Series 3 printer contains the following information:

**%*
**% Zebra Card Printer PPD for Common UNIX Printing System (CUPS).*
**%*
**%*

**FormatVersion: "4.3"*
**FileVersion: "1.0.0.9"*
**PCFileName: "ZXPS31.PPD"*
**Product: "(Card Printer)"*
**LanguageVersion: English*
**LanguageEncoding: ISOLatin1*
**Manufacturer: "Zebra"*
**ModelName: "ZXPS31"*
**ShortNickName: "ZXPS31"*
**NickName: "ZXPS31"*
**PSVersion: "(3010.000) 550"*
**LanguageLevel: "2"*
**ColorDevice: True*
**DefaultColorSpace: RGB*
**FileSystem: False*
**Throughput: "3"*

This information specifies that the raster driver understands PostScript Level 2, is a color device, and so forth. The PPD describes allowable paper sizes, memory configurations, minimum font set for the printer, and even specifies a tree-based user interface for printer-specific configuration.

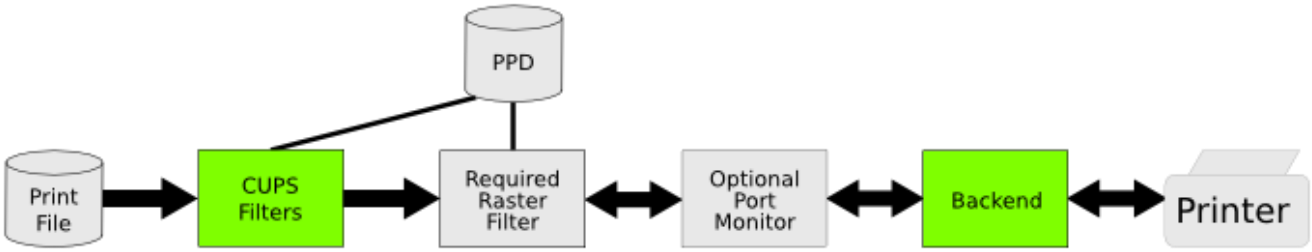
ZXP Series 3 Card Printer Linux Driver Implementation

The printer graphics perform the rendering operations as per the CUPS Architecture Specification. The Zebra Linux driver has a raster module responsible for rendering image data specific to the Zebra ZXP Series 3 printer.

Zebra Filter

A CUPS raster printer driver consists of a PPD file that describes the features and capabilities of the device, one or more filter programs that prepare print data for the device, and zero or more support files for color management, online help, and so forth. The PPD file includes references to all of the filters and support files used by the driver.

Every time a user prints something, the scheduler program determines the format of the print job, and the programs required to convert that job into something the printer understands. CUPS includes filter programs for many common formats. For example, CUPS has a filter to convert Portable Document Format (PDF) files into CUPS raster data. The following figure shows the data flow of a typical print job.



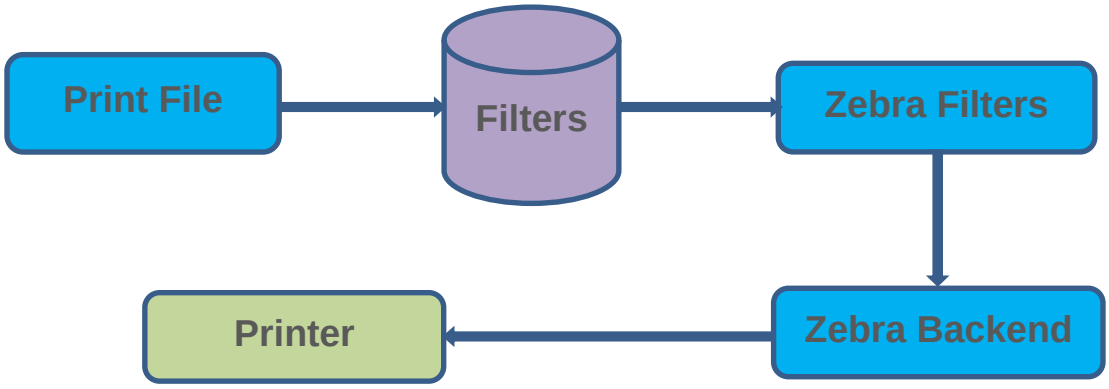
The raster filter converts CUPS raster data into a format the printer understands. The Zebra Filter will send the Zebra ZXP Series 3 printer commands along with the YMCK color buffer data to the backend.

Zebra Backend Communications

The Zebra CUPS backends are the modules with which CUPS sends data to printers. Backends take the print data from the last filter in the filter system, and send it to the printer for printing. The backend manages bi-directional communications with the printer. Zebra backends include backend programs for common direct-connect interfaces and network protocols, such as a USB connected printer.

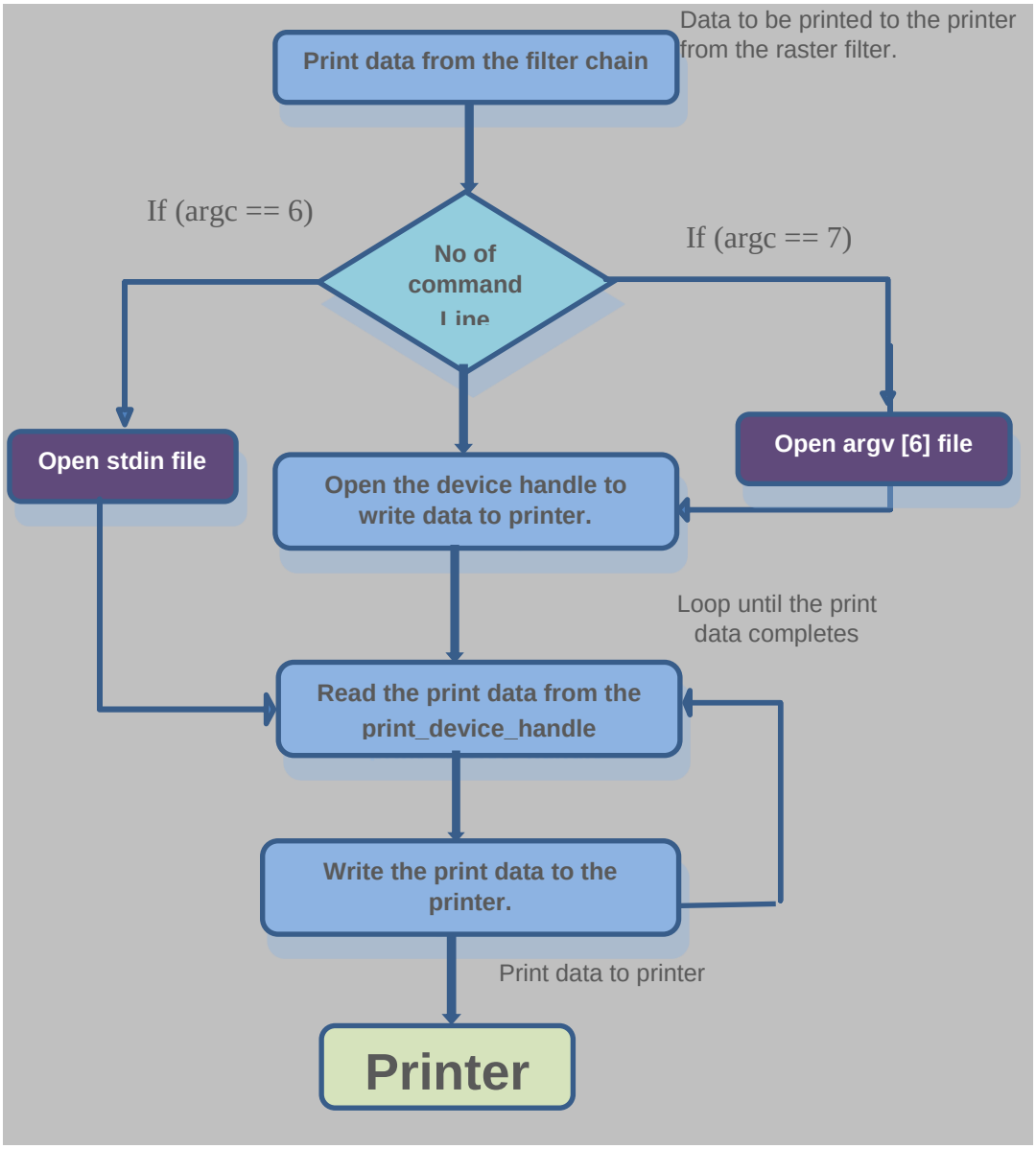
To support the USB printer and Ethernet interfaces, specific backend functionality can be added to the Zebra backend. The USB backend uses the Linux /dev/usb/lp0 device file, and the Ethernet backend uses the TCP/IP sockets to communicate with the Ethernet printer.

Zebra printer communications with the backends can be illustrated as follows:



Zebra USB Backend

Zebra USB printer is supported by adding the Zebra USB backend to the CUPS backend that supports the bi-directional print functionally required. After the filtering process is complete, the Zebra USB backend forwards the data to the printer hardware, and receives any error(s) and the printer status from the printer.



Zebra Ethernet Backend

Ethernet Communications are accomplished with the help of Ethernet backends. Zebra Ethernet backend is supported by adding the Socket backend to the CUPS backend. Ethernet is supported by the Linux TCP/IP Sockets. Print data is sent to the print filters using the CUPS print scheduler IPP protocol. The CUPS IPP is implemented on top of HTTP 1.1 functionality for both USB and Ethernet backends.

In the Ethernet backend, a port to the printer is opened through TCP/IP sockets, and two-way communication to the printer is established using socket communication. The Socket backend uses the TCP port 9100 for communication with the printer. The Socket backend checks the command line arguments to discover the device host name, port, and print file to be sent to the printer. Including the print filename in the command line arguments is optional. If the Socket backend does not get the filename in the arguments, it means that the backend is getting the print data from the standard input (stdin). In this case, the backend extracts the print data from the stdin and transmits it to the printer. If the print data is in the stdin, the argc value is six; otherwise, the value is seven. The argv[0], obtained from the command line arguments, contains the device URI that is the host name of the device, and the port number with which communication to the printer is established.

Advantages of the Zebra Linux Driver Implementation

The standard CUPS backend is capable of sending the data to the printer. However, this is insufficient for communicating a complete Card specific print job to a Card printer.

In the Zebra ZXP Series 3 printer, the data needs to be sent in communication blocks with the proper printer commands. The printer status needs to be checked at the end of each command. If any error occurs, the error information is sent to the user. The standard CUPS backend cannot accommodate this end user interaction.

For the Zebra ZXP Series 3 Card printer, Zebra developed a custom CUPS Backend. The Zebra ZXP Series 3printer custom CUPS backend performs the following actions:

- Sends the data to the printer in a Card printer specific data communication protocol understood by the printer device firmware.
- Checks the printer status at the end of every command.
- Reports error and status information back to the user (such as out or low ribbon media, card jam, door open, etc.).
- Acts according to user specified error or warning inputs

In addition, Zebra has implemented a Plug n Play USB communication feature support through a USB Printer Installer application that handles the Plug n Play through udev rules. The Zebra ZXP Series 3 Udev rules can be found in the */lib/udev/rules.d* directory. When a user inserts the USB printer, Zebra ZXP3 USB Printer Installer calls the udev rules, and it adds the printer if it is new. If it is already installed, it enables the printer in the Printer Control Panel.

Driver Exceptions and Errors

Exceptions

Various components of the print spooler handle the following exception scenarios:

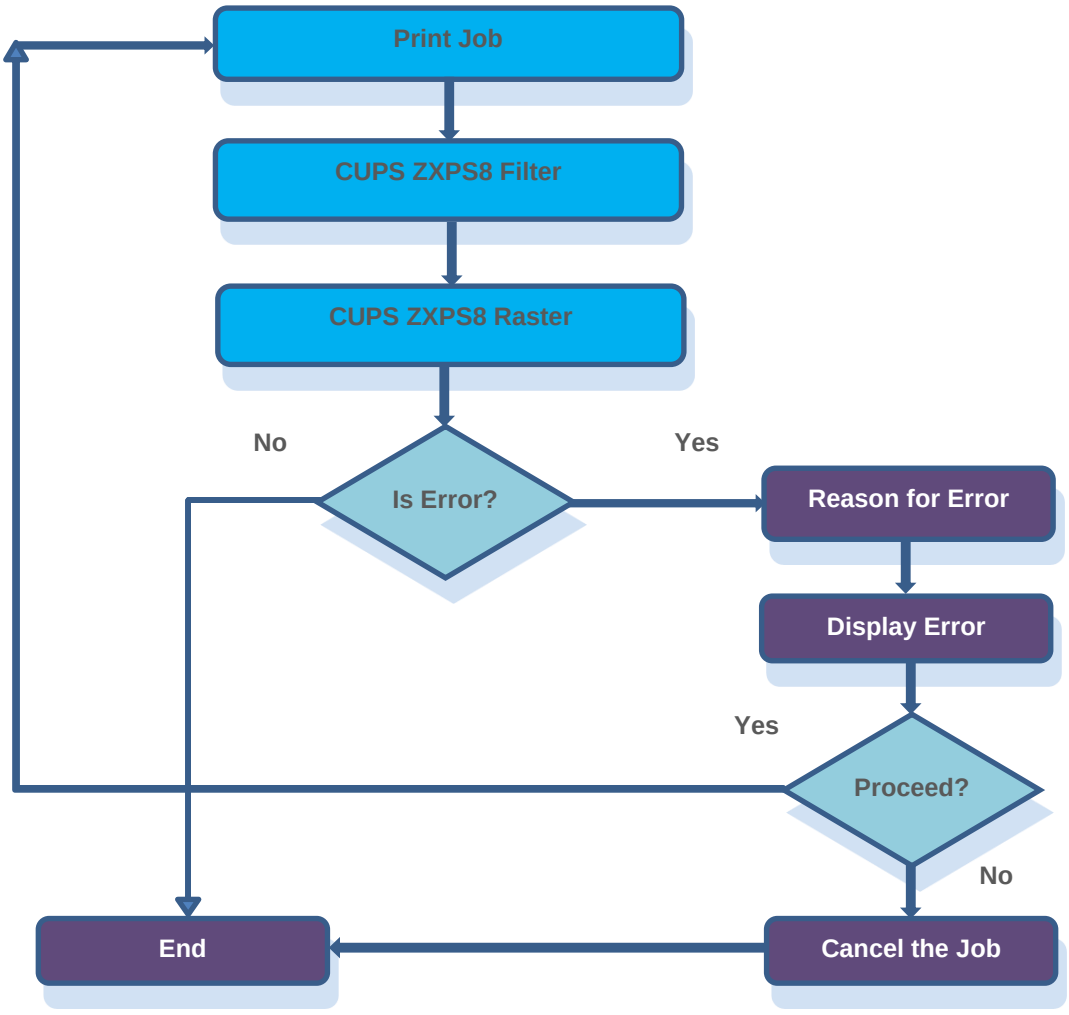
- Incorrect printer settings
- The printer driver validates the UI settings values during printing. If a setting is not valid, it returns the appropriate error code, and the request fails.
- The printer UI performs UI validation to block the user from choosing any invalid combination of print settings.

Exceptions (continued)

- Error in print monitor modules:
- Any error in the print monitor module results in the failure of the print job as well as a spooler error. On restart, the CUPS spooler continues to perform printing, since the job spool file will not be deleted until the document print is complete.

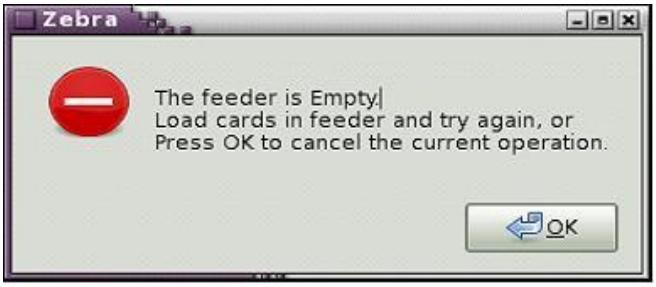
Errors

The component Error Finder always watches the port for an error by sending a GetStatus message to the printer port. Whenever it gets an error, it collects the appropriate error number from the printer's port. After collecting the number, it calls the Error Number Processing component, and based on the error number, it locates the correct error description. The error handling can only be handled in the customized Zebra backend. (The CUPS default backend will not support error handling.)



After collecting the Error Description, the component Error Display Mechanism displays the Error report for the user via a User Interface dialog. Then, it waits for a user response to that error report. Based on the response, either it continues the job or quits the current job. This error cycle continues until the termination of the current job in the printer. The previous flowchart describes how error handling works in CUPS.

For example, while printing, suppose the Feeder Empty Error occurs. First, the Error Finder receives Error 4. It forwards the error number to Error Processing; this is like a Switch case statement. It collects the error description based on the error number. The Error Display Mechanism displays the Error report. For this error, the error dialog will display the following:



Driver Implementation Steps

Implementing PPD

PPD is the entry point to the CUPS Printing. Once the user prints the job, the CUPS Scheduler searches for the respective printer PPD, and runs the filter program specified in the PPD. The Post Filter runs after execution of the Pre-filter.

Implementing Filters

The filter is the program which converts the raster data into the printer understandable format. In the filters, the Print Data needs to rasterize depending on the Ribbon Type, Ribbon Combination, Monochrome or Color settings. These settings are stored by the Printer Properties GUI. Composing the Image buffer involves Zebra's various printing algorithm and Lookup Table (LUT) files. These LUT files will be stored during the installation process in the */usr/local/ZebraZXP3Driver/LUTFiles/.* The Print Data composing will be done for the number of copies for all the print data. The Zebra filter binary is located in the CUPS filter path:

/usr/lib/cups/filte/ rastertozxp3

/usr/lib/cups/filte/ pstozebraps

The Zebra filter, with the help of the Zebra backend, automatically detects and prepares the raster buffer (color or monochrome) based on the ribbon type currently used in the ZXP Series 3 printer.

Zebra Backend Implementation

The Zebra ZXP Series 3 printers use Zebra specific USB and Socket backends for the USB printer and the Ethernet printer.

- The USB backend uses the libUSB for communicating with the USB printer.
- The Socket backend uses the TCP/IP Sockets for communicating with the Ethernet printer.
- The Zebra backend binary file is located in the CUPS Backend path:

/usr/lib/cups/backend/zxps3usb

/usr/lib/cups/backend/zxps3socket

Zebra Driver Installer

The Zebra ZXP Series 3 Linux Printer Driver is based on the module CUPS architecture. All executable files should be placed in the CUPS lib path or usr path of the Linux operating system. Most of the printer drivers use rpm- or deb-based installation.

To facilitate the proper setup and configuration, the Zebra driver uses an “Install Jammer”-based Installation Wizard for creating the Linux installation module. This is more user friendly than other packages and ensures that all Zebra required custom driver components get installed correctly.

Technical Implementation Considerations

Integration of Components

The integration of the printer driver components must be implemented in an incremental manner as described below:

- First, install the printer using the driver provided GUI, which will install the required USB and Ethernet communication modules. This will create the necessary printer driver file port or any other virtual port or USB.
- Next, install the Zebra CUPS Filter, so that the rasterization of the print data can be performed.
- Next, install the Zebra CUPS Backend for USB module, then add the Ethernet module, so that printing over the Ethernet is possible. (This allows you to print normal print jobs.)
- Install Jammer Based-Installation packs, all of the CUPS modules, udev rules, and other necessary script files, so that all the driver modules can be installed.

Identification of Integration Sequence

1. Basic parts of the printer driver (StartPage, EndPage, DoPrinting) are implemented.
2. Printer UI (all features) are implemented and integrated with the driver.
3. Zebra CUPS Backend features are implemented and integrated with modules generated in steps 1-2.
4. Printer driver (all printing related features) are implemented and integrated in modules in steps 2-4.
5. Once all are available, install Jammer-based installation.

Advanced Setup and Configuration

For developing the CUPS-based Zebra ZXP Series 3 Linux Driver, the CUPS library must be installed in the Development System. If CUPS is not found in the Development System, it may be installed using the command ‘*yum install cups'* for the RedHat-based system. And, for the Debian-based system, use *'sudo apt-get install cups*'.

Apart from CUPS, the Install Jammer application must be installed in the Development System. For compiling the Property Pages UI code, Qt must be installed in the Development System along with the Qt library.

Common Name	Filename	Directory
Zebra Filter Compiled	Prefilter filename: pstozxp3ps, Raster filename: rastertozxp3	/usr/lib/cups/Filter
Zebra Backend Binary	zxp3usb and zxp3socket	/usr/lib/cups/Backend.
Zebra PPD files		/usr/share/cups/model
CUPS		/etc/cups/ppd (once the printer is added)
Zebra Rules Files		/lib/udev/rules.d
Zebra PnP Binary	Zebra ZXP3 USB Printer Installer	/lib/udev/
Zebra Error Handling Binary	ZXP3ErrorNotify	/usr/bin/
All other supported files and scripts		/usr/local/ZebraZXP3Driver
MIME type files	pstozxp3ps.convs	/usr/share/cups/mime

Once the Printer is added to the printer panel, the user performs a test print, which is available in the Printer UI Dialog.

Note: If you wish to set up a Development environment in Ubuntu Linux, please install the following packages for compiling the Zebra ZXP Series 3 Printer Driver:

```
sudo apt-get install libgtk2.0-dev
sudo apt-get install cups libcups2-dev
sudo apt-get install libcupsimage2-dev
sudo apt-get install libusb-dev
sudo apt-get install libudev-dev
sudo apt-get install libnotify-dev
sudo apt-get install libqt4-dev
```

Driver and Print Job Management Considerations

The Linux CUPS Daemon manages the Scheduler. All print jobs from the application are queued by CUPS and sent to the CUPS Scheduler. If any error occurs in the Filter or in the backend, the CUPS Scheduler will stop the print process. All errors are logged in the CUPS error log file, and are located in the */var/log/cups/error_log*. The CUPS debug log may be enabled in the cupsd.conf, which is available in the */etc/cups/cupsd.conf*. The various log level options are debug, debug2, warn, and error. The job may be managed through the CUPS web interface also. CUPS web interface is available in the http://localhost:631 location.

Difference from Normal Implementation

The Printer Linux Driver must contain the filter and the PPD file, and the Plug n Play USB communication and error handling is optional.

In the Zebra ZXP Series 3 printer, the driver overrides the default CUPS backend and the default CUPS URI. To view the Zebra ZXP Series 3 Printer Linux CUPS URIs, see the table below:

Zebra ZXP Series 3 Printer	URI
USB	zxp3usb://ZXP11/Zebra-ZXP31-USB-Printer-Serial=123456478
Socket	zxp3socket://*ZXP11/10.1.6.91:9100*

Considerations in Successful Zebra Linux Driver Deployment

The following must be considered during the Printer driver development:

- Zebra Custom filter must fulfill the CUPS filter requirements.
- Zebra Custom backend must fulfill the CUPS backend requirements.
- The syntax of CUPS PPD must be checked.
- The Pre-filter (if used), Post-filter binaries must be mentioned in the CUPS PPD.
- Zebra CUPS PPD must be copied to*: /usr/share/cups/model* during installation
- Zebra Filter must be copied to the CUPS filter folder: */usr/lib/cups/filter*
- Zebra Backend must be copied to the CUPS backend folder: */usr/lib/cups/backend*
- Check the Filter and Backend permission in the target folder: */usr/lib/cups/filter, /usr/lib/cups/backend*
- Compile Zebra Printer Properties GUI with the Qt. Ensure that the target machine contains Qt Runtime libraries to run the Zebra Printer Properties GUI.
- Installation and Uninstallation requires Linux root permission.
- PnP module should be copied to Udev folder: */lib/udev*
- PnP Udev rules should be copied to*: /lib/udev/rules.d*

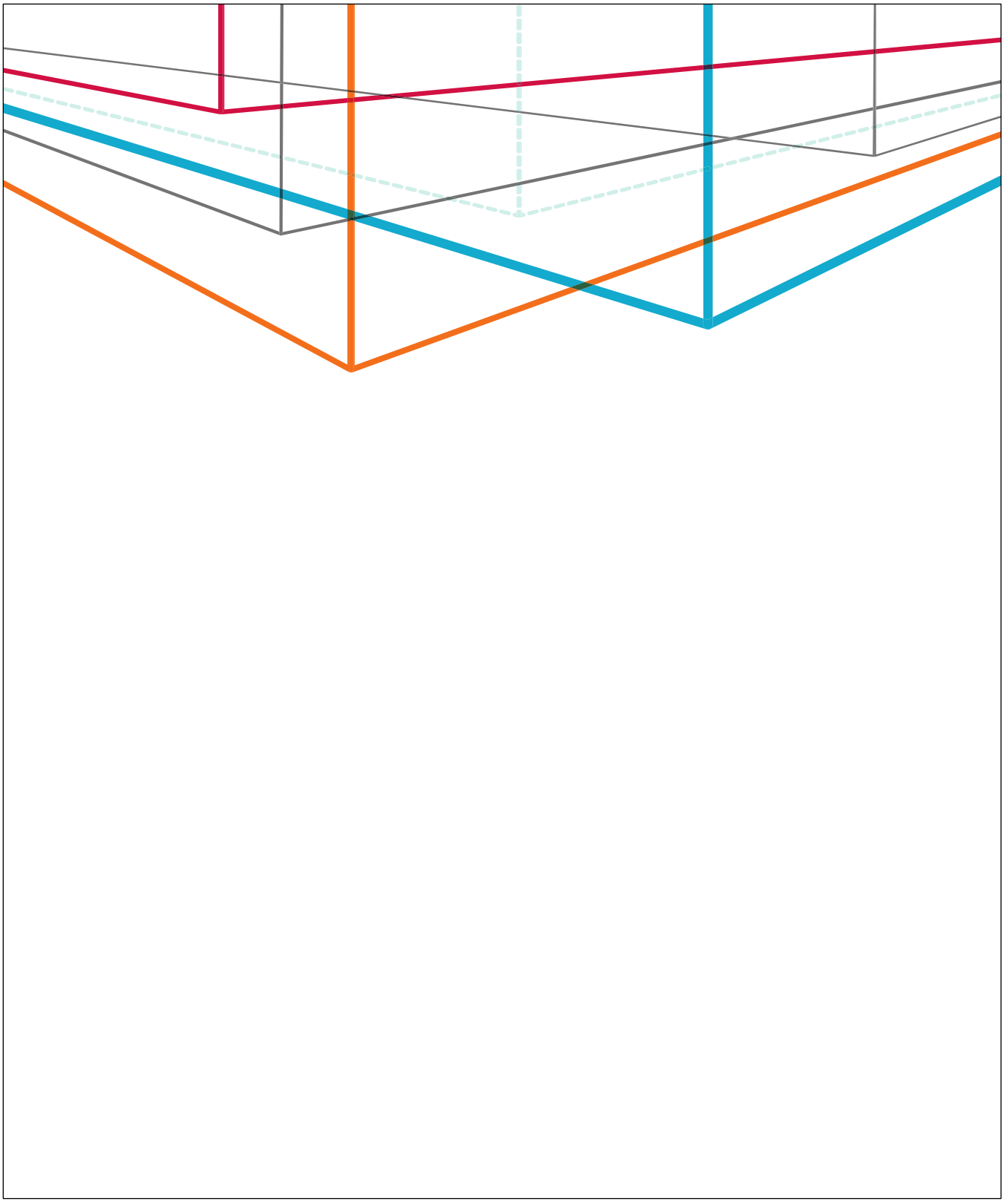
Appendix

Various file paths (see table below) are used in the Linux CUPS Printer Driver:

Modules	**Path**
CUPS Filter	/usr/lib/cups/filter/ rastertozxp3
CUPS USB Backend	/usr/lib/cups/backend/zxp3usb
CUPS Socket Backend	/usr/lib/cups/backend/zxp3socket
CUPS PPD	/usr/share/cups/model/ZebraZXPS<>Printer.ppd
Udev Zebra ZXPS3 PnP Binary	/lib/udev/ ZXPS3-USB-Printer-Installer
Udev Rules	/lib/udev/20-ZebraZXP3Series.rules
Udev Rules	/lib/udev/70-ZebraZXP3Series.rules
Udev Rules	/lib/udev/zebrazxp3.rules
Zebra Printer Driver Installation Folder	/usr/local/ZebraZXP3Driver
MIME Type	/usr/share/cups/mime/pstozxp3ps.convs
Error Handling Binary	/usr/bin/ ZXP3ErrorNotify

Various Acronyms used in this document:

- CUPS → Common Unix Printing System
- PPD → Postscript Printing Document
- PnP → Plug and Play
- MIME → Multi-Purpose Internet Mail Extensions
- TCP → Transmission Control Protocol
- IP → Internet Protocol



Corporate Headquarters
+1 800 423 0442
inquiry4@zebra.com

Asia-Pacific Headquarters
+65 6858 0722
apacchannelmarketing@zebra.com

EMEA Headquarters
+44 (0)1628 556000
mseurope@zebra.com

Latin America Headquarters
+1 847 955 2283
inquiry4@zebra.com

Other Locations / USA: California, Georgia, Illinois, Rhode Island, Texas, Wisconsin **Europe:** France, Germany, Italy, the Netherlands, Poland, Spain, Sweden, Turkey, United Kingdom **Asia Pacific:** Australia, China, Hong Kong, India, Indonesia, Japan, Malaysia, Philippines, Singapore, South Korea, Taiwan, Thailand, Vietnam **Latin America:** Argentina, Brazil, Colombia, Florida (LA Headquarters in USA), Mexico **Africa/Middle East:** Dubai, South Africa

(11/14)