Free Standards Printing Working Group
Working Draft V1
Catagory: proposed standard

Ben Woodard
Red Hat Inc.
February 7, 2002

API and SPI for handling printer capabilities

Status of this Memo

This memo is a working draft for use within the Free Standards Printing Working Group. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

Table of Contents

1. Introduction

This document defines a proposed standard for manipulating printer capabilities information from the programatic point of view. In other words, it specifies the data structures and function calls which elements of the print system use to pass around information about the printer capabilities. It takes great pains to avoid either defining an on disk format or network protocol to pass this capabilities information around. Other standards, such as Adobe's PPD file format, the IETF's PWG's UPDF, and IPP already suitably define those standards.

2. Overview

The world of printing has changed over the past few years. In the past, printers had very few capabilities that an application needed to worry about. One extreme of this is the ASCII only text printer. Whose sole ability was inscribe an ASCII data stream onto paper. These days, printers are dramatically more featureful, they have duplexers, finishers, different imaging models, and a whole host of features that an application may want to make use of.

Many different attempts have been made to describe the format in which printer capabilities can be encoded either for persistant storage or for transport across a network protocol. However, to our knowlege no

attempt has been made in the UNIX world to provide an API to access this information. As applications become more sophisticated and no longer emit simple text, they need to have information about what the destination printer can do, and how to make the printer do that. Also, some of the features are user selections and so information about how to present these options to a user is also necessary.

Also since there are many different sources for this information, some of them network protocols, some of them static representations such as disk files, and some of them processes which directly communicate with a printing device, a standard must also be defined which defines how the sources of this information should provide their information to the agent which collects the information and organizes it into a consistant view for the application.

3. Existing Definitions

BSD -- Berkley Software Distribution. The first publicly released version of UNIX distributed in source code format.

BSD LPD -- The original print spooler for the BSD UNIX operating system.

IPP - IPP IS an acronym for Internet Printing Protocol. It was developed by the IETF's PWG. This protocol specifies several things. The most fundemental is how a print job and all the associated metadata can be sent from one machine to another. It also covers discovery of available printers. This protocol is largly seen as the successor to the LPD Protocol.

LPD Protocol -- The internet protocol used by BSD's LPD and a whole host of other print spoolers and printing devices to transmit print jobs between internet connected devices.

PDL - PDL is an acronym for Page Description Language. It is a structure of data which describes how to put ink on a piece of paper or a series of pieces of paper.

PostScript -- Adobe Systems Inc. defined a device independant PDL that has been widely deployed on laser printers as well as other printing devices and other non-printing related systems.

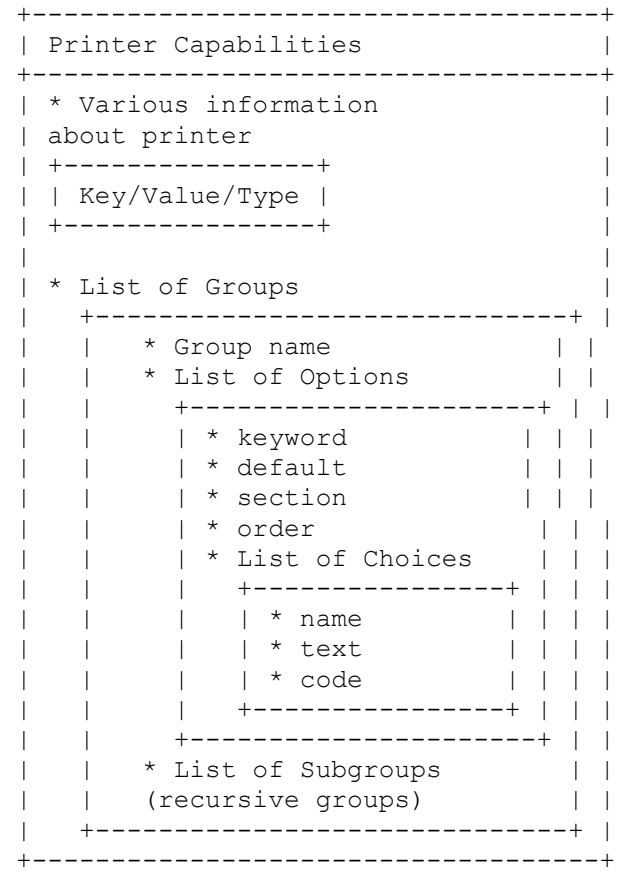
PPD files -- Adobe Systems Inc. Defined a de

Printer Capabilities -- Printer capabilities are three distinct things. They are the physical limiations and features of the printing device, information on how to access these features and information on how to present this information to the user.

UPDF -- An acronym for Universal Printer Description Format. Whereas PPD files are only applicable to PostScript printers, UPDF is an evolving standard that codifies similar information for all types of printers including PostScript and simple ink jet printers.

4. Data Structures and Enumerations

The overall structure of the data is as follows:



The overall printer capabilites structure contains a few hard coded elements and a few lists of information that will always be needed. It will also contain a dynamic key/value/type structure where arbitrary information can be stored.

Also inside of the capabilities structure there is a list of groups of options. These groups can have recursive subgroups and also a list of options. Having a heirarchy of groups and subgroups allows the printer vendor to provide some logical structure to the capabilities for display purposes. It also allows allows different contributors of the overall printer capabilities information to store their groupings of options together.

The options are the actual features that are set by the user. They need some identifying unique keyword, a default action, a section in which they will be emmitted as well as some representation of a value. In many cases this representation will be a list of choices, a string, or a number.

The data structures that are presented below in their most minimal form. Different implementations can add additional members to the structures as their implementation requires but they must support at least specified items.

4.1 UIType Enumeration

When an option is displayed, it is represented with one of several kinds of widgets and those widgets put constraints on what kind of value the option can take on. For example, a pickone kind of option would allow the user to pick one of discrete list of values whereas an UI type of string would allow the input of a string.

```
  typedef enum {                // *** UI types ***
    LPSCAP_UI_BOOLEAN,          // True or False option
    LPSCAP_UI_PICKONE,          // Pick one from a list
    LPSCAP_UI_PICKMANY,                 // Pick zero or more from a list
    LPSCAP_UI_STRING,
    LPSCAP_UI_INT,
    LPSCAP_UI_FLOAT
  } LPSCapUIType;
```

4.2 SectionOrder Enumeration

When constructing the print data, their are several logical points within a print job. Many options only make sense in the context of these transition points within a print job. This enumeration specifies where a particular printer control command actually makes sense.

```
  typedef enum {                // *** Order dependency sections ***
    LPSCAP_ORDER_ANY,           // Option code can be anywhere in the
file
    LPSCAP_ORDER_DOCUMENT,      // ... must be in the DocumentSetup
section
    LPSCAP_ORDER_EXIT,          // ... must be sent prior to the document
    LPSCAP_ORDER_JCL,           // ... must be sent as a JCL command
    LPSCAP_ORDER_PAGE,          // ... must be in the PageSetup section
    LPSCAP_ORDER_PROLOG                 // ... must be in the Prolog
section
  } LPSCapSectionOrder;
```

4.3 Choice Data Structure

In many cases the value of an option will be a one of many possible choices. The Choice data structure represents the an individual choice.

```
  typedef struct {              // *** Option choices ***
    boolean marked;             // 0 if not selected, 1 otherwise
    char *choice;               // Computer-readable option name: 41
chars
    char *text;                 // Human-readable option name: 81 chars
    char *code;                 // Code to send for this option NOT NULL
term
    unsigned codelen;           // length of code.
  } LPSCapChoice;
```

The computer readable version of the string is a normal NULL terminated C string.

The text version is a UTF4 encoded multibyte character NULL terminated string designed to be readable to the user. At the time that the capabilities structure is prepared for the user, the appropriate translation of the human readable strings is selected and inserted

into the structure.

The code is the stream of data to send to the printer to activate this choice for the option that contains it. This string is an 8 bit clean string of length codelen rather than a NULL terminated string. In most cases, an application will not care about how to turn a feature on but will rather, pass the fact that this option is set to a particular value in a print job attribute. Only applications which feel that they need very tight control over the generation of printer data should make use of the code variable.

4.2 Option Data Structure

The option data structure is the logical representation of one feature or one user selectable item.

```
  typedef struct {             // *** Options ***
    boolean conflicted;        // 0 if no conflicts exist, 1 otherwise
    boolean emitted;           // 1 already emitted 0 otherwise

    char *keyword;             // Option keyword name ("PageSize", etc.)
    char *defchoice;           // Default option choice
    char *text;                // Human-readable text

    LPSCapUIType ui;           // Type of UI option
    LPSCapSectionOrder section;        // Section for command
    float order;               // Order number
    LPSCapChoice *choices;     // should be a GSList of PpdChoice
objects
    char *value;               // other values
  } LPSCapOption;
```

5. API Functions
6. Sample usage
7. Interface with the Spooler API
8. SPI