Linux Standard Base Printing Specification 4.1

Linux Standard Base Printing Specification 4.1

Contents

List of Tables

Foreword

This is version 4.1 of the Linux Standard Base Printing Specification. This specification is one of a series of volumes under the collective title *Linux Standard Base*:

- Core
- C++
- Desktop
- Languages
- Printing

Note that the Core, C++ and Desktop volumes consist of a generic volume augmented by an architecture-specific volume.

Status of this Document

This is a released specification. Other documents may supersede or augment this specification. A list of current Linux Standard Base (LSB) specifications is available at http://refspecs.linuxfoundation.org (http://refspecs.linuxfoundation.org/).

If you wish to make comments regarding this document in a manner that is tracked by the LSB project, please submit them using our public bug database at http://bugs.linuxbase.org. Please enter your feedback, carefully indicating the title of the section for which you are submitting feedback, and the volume and version of the specification where you found the problem, quoting the incorrect text if appropriate. If you are suggesting a new feature, please indicate what the problem you are trying to solve is. That is more important than the solution, in fact.

If you do not have or wish to create a bug database account then you can also e-mail feedback to `<lsb-discuss@lists.linuxfoundation.org>` (subscribe (http://lists.linux-foundation.org/mailman/listinfo/lsb-discuss), archives (http://lists.linux-foundation.org/pipermail/lsb-discuss/)), and arrangements will be made to transpose the comments to our public bug database.

Introduction

The LSB defines a binary interface for application programs that are compiled and packaged for LSB-conforming implementations on many different hardware architectures. A binary specification must include information specific to the computer processor architecture for which it is intended. To avoid the complexity of conditional descriptions, the specification has instead been divided into generic parts which are augmented by one of several architecture-specific parts, depending on the target processor architecture; the generic part will indicate when reference must be made to the architecture part, and vice versa.

This document should be used in conjunction with the documents it references. This document enumerates the system components it includes, but descriptions of those components may be included entirely or partly in this document, partly in other documents, or entirely in other reference documents. For example, the section that describes system service routines includes a list of the system routines supported in this interface, formal declarations of the data structures they use that are visible to applications, and a pointer to the underlying referenced specification for information about the syntax and semantics of each call. Only those routines not described in standards referenced by this document, or extensions to those standards, are described in the detail. Information referenced in this way is as much a part of this document as is the information explicitly included here.

The specification carries a version number of either the form `x.y` or `x.y.z`. This version number carries the following meaning:

1. The first number (*x*) is the major version number. Versions sharing the same major version number shall be compatible in a backwards direction; that is, a newer version shall be compatible with an older version. Any deletion of a library results in a new major version number. Interfaces marked as deprecated may be removed from the specification at a major version change.
2. The second number (*y*) is the minor version number. Libraries and individual interfaces may be added, but not removed. Interfaces may be marked as deprecated at a minor version change. Other minor changes may be permitted at the discretion of the LSB workgroup.
3. The third number (*z*), if present, is the editorial level. Only editorial changes should be included in such versions.

Since this specification is a descriptive Application Binary Interface, and not a source level API specification, it is not possible to make a guarantee of 100% backward compatibility between major releases. However, it is the intent that those parts of the binary interface that are visible in the source level API will remain backward compatible from version to version, except where a feature marked as "Deprecated" in one release may be removed from a future release. Implementors are strongly encouraged to make use of symbol versioning to permit simultaneous support of applications conforming to different releases of this specification.

LSB is a trademark of the Linux Foundation. Developers of applications or implementations interested in using the trademark should see the Linux Foundation Certification Policy for details.

I Introductory Elements

1 Scope

The LSB-Printing module defines the printing components found on an LSB conforming system.

2 Normative References

The specifications listed below are referenced in whole or in part by the LSB-Printing Module Standard. Such references may be normative or informative; a reference to specification shall only be considered normative if it is explicitly cited as such. The LSB-Printing Module may make normative references to a portion of these specifications (that is, to define a specific function or group of functions); in such cases, only the explicitly referenced portion of the specification is to be considered normative.

Table 2-1 Normative References

Name	Title	URL
CUPS API Reference	CUPS 1.2 API Reference	http://www.cups.org/documentation.php/doc-1.2/
Filesystem Hierarchy Standard	Filesystem Hierarchy Standard (FHS) 2.3	http://www.pathname.com/fhs/
ISO C (1999)	ISO/IEC 9899: 1999, Programming Languages --C	
PPD Specification	PostScript Printer Description File Format Specification version 4.3	http://partners.adobe.com/public/developer/en/ps/5003.PPD_Spec_v4.3.pdf
PPD Specification Update	Update to PPD Specification Version 4.3	http://partners.adobe.com/public/developer/en/ps/5645.PPD_Update.pdf

3 Requirements

3.1 Relevant Libraries

The libraries listed in Table 3-1 shall be available on a Linux Standard Base system, with the specified runtime names. This list may be supplemented or amended by the architecture-specific specification.

Table 3-1 Standard Library Names

Library	Runtime Name
libcups	libcups.so.2
libcupsimage	libcupsimage.so.2

These libraries will be in an implementation-defined directory which the dynamic linker shall search by default.

4 Terms and Definitions

For the purposes of this document, the terms given in *ISO/IEC Directives, Part 2, Annex H* and the following apply.

archLSB

Some LSB specification documents have both a generic, architecture-neutral part and an architecture-specific part. The latter describes elements whose definitions may be unique to a particular processor architecture. The term archLSB may be used in the generic part to refer to the corresponding section of the architecture-specific part.

Binary Standard, ABI

The total set of interfaces that are available to be used in the compiled binary code of a conforming application, including the run-time details such as calling conventions, binary format, C++ name mangling, etc.

Implementation-defined

Describes a value or behavior that is not defined by this document but is selected by an implementor. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence of the value or behavior. An application that relies on such a value or behavior cannot be assured to be portable across conforming implementations. The implementor shall document such a value or behavior so that it can be used correctly by an application.

Shell Script

A file that is read by an interpreter (e.g., awk). The first line of the shell script includes a reference to its interpreter binary.

Source Standard, API

The total set of interfaces that are available to be used in the source code of a conforming application. Due to translations, the Binary Standard and the Source Standard may contain some different interfaces.

Undefined

Describes the nature of a value or behavior not defined by this document which results from use of an invalid program construct or invalid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

Unspecified

Describes the nature of a value or behavior not specified by this document which results from use of a valid program construct or valid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

In addition, for the portions of this specification which build on IEEE Std 1003.1-2001, the definitions given in *IEEE Std 1003.1-2001, Base Definitions, Chapter 3* apply.

5 Documentation Conventions

Throughout this document, the following typographic conventions are used:

`function()`

the name of a function

command

the name of a command or utility

`CONSTANT`

a constant value

`parameter`

a parameter

`variable`

a variable

Throughout this specification, several tables of interfaces are presented. Each entry in these tables has the following format:

name

the name of the interface

(symver)

An optional symbol version identifier, if required.

[*`refno`*]

A reference number indexing the table of referenced specifications that follows this table.

For example,

forkpty(GLIBC_2.0) [SUSv3]

refers to the interface named `forkpty()` with symbol version `GLIBC_2.0` that is defined in the `SUSv3` reference.

Note: For symbols with versions which differ between architectures, the symbol versions are defined in the architecture specific parts of ISO/IEC 23360 only.

6 PPD Format Extensions

The Postscript Printer Description (PPD) format is used in a text file to describe device capabilities for a printing device. PPD files shall conform to the format described by PPD Specification and PPD Specification Update. In addition, several extensions to the standard attribute list are recognized, as listed below. The "cupsVersion" attribute is required in a compliant PPD, while the other attributes are optional.

cupsColorProfile

This string attribute specifies an sRGB-based color profile consisting of gamma and density controls and a 3x3 CMY color transform matrix.

The attribute has the following parameter usage:

*cupsColorProfile Resolution/MediaType: "density gamma m00 m01 m02 m10 m11 m12 m20 m21 m22"

The Resolution and MediaType values may be "-" to act as a wildcard. Otherwise, they must match one of the Resolution or MediaType attributes defined in the PPD file.

The density and gamma values define the gamma and density adjustment function such that (in terms of C math):

f(x) = density * pow(x, gamma)

The m00 through m22 values define a 3x3 transformation matrix for the CMY color values. The density function is applied after the CMY transformation:

| m00 m01 m02 | | m10 m11 m12 | | m20 m21 m22 |

cupsFax

This boolean attribute specifies whether the PPD defines a facsimile device. The default is false.

cupsFilter

The attribute has the following parameter usage:

*cupsFilter: "source/type cost program"

This string attribute provides a conversion rule from the given source type to the printer's native format using the filter "program". A source type is specified according to the conventions of the MIME specification, using "type/subtype" nomenclature, and may refer to a standard MIME type or a CUPS-specific MIME type using the prefix "vnd.cups-" in the subtype. If a printer supports the source type directly, the special filter program "-" may be specified. The cost is an arbitrary positive integer, used to calculate the relative impact a print job has on system load.

cupsManualCopies

This boolean attribute notifies the RIP filters that the destination printer does not support copy generation in hardware. The default value is false.

cupsModelNumber

This integer attribute specifies a printer-specific model number. This number can be used by a filter program to adjust the output for a specific model of printer.

cupsVersion

The attribute has the following parameter usage:

*cupsVersion: "major.minor"

This required attribute describes which version of the CUPS PPD file extensions was used. Currently it must be the string "1.0" or "1.1". The strings "1.2" and "1.3" represent newer versions of the CUPS PPD API that are not covered in this version of the specification, and are currently not allowed, although they may be found in non-conforming PPDs which use a newer version of the CUPS PPD specification.

FoomaticIDs

The attribute has the following parameter usage:

*FoomaticIDs printer driver

The parameters correspond to the IDs in the Foomatic database for the printer and driver, respectively.

FoomaticNoPageAccounting

This boolean attribute tells foomatic-rip whether or not to insert accounting information into the PostScript data stream. By default, foomatic-rip will insert this information.

FoomaticRIPCommandLine

The attribute has the following parameter usage:

*FoomaticRIPCommandLine "code"

This attribute defines the command line in the "code" parameter for the renderer that is called by foomatic-rip. The command must take PostScript on standard input and provide the job data stream in the printer's native language on standard output. The command must exit with status 0 if the conversion was successful and exit with another status if an error occurs. The "code" parameter may contain option setting wildcards, as described below under "FoomaticRIPOption".

FoomaticRIPDefault

The attribute has the following parameter usage:

*FoomaticRIPDefaultOptionName value

This attribute sets a default for a Foomatic option. The name of the attribute should contain the name of the option appended to "FoomaticRIPDefault", with the desired default value as the only parameter.

This option is only used to provide numeric options in the PPD, which are not supported by the Adobe spec, via enumerated options, and should not be used except for that purpose.

FoomaticRIPOption

The attribute has the following parameter usage:

*FoomaticRIPOption name: type style spot [order]

This attribute sets options for the command line specified in the "FoomaticRIPCommandLine" attribute. The "name" parameter specifies the option name, the "type" parameter specifies the option type, the "style" parameter specifies one of "CmdLine", "JCL", "PS", or "Composite", and the "spot" parameter specifies a letter, which is prepended with a "%" and used in the "FoomaticRIPCommandLine" attribute to indicate where the option should go in the command line. The optional "order" parameter indicates an order number for one-choice options.

FoomaticRIPOptionAllowedChars

The attribute has the following parameter usage:

*FoomaticRIPOptionAllowedChars name: "code"

This option sets a list of allowed characters in a string option. The "name" parameter identifies the option, while the "code" parameter is a list of allowed characters.

FoomaticRIPOptionAllowedRegExp

The attribute has the following parameter usage:

*FoomaticRIPOptionAllowedRegExp name: "code"

This option causes the option named by "name" to be validated by the Perl-style regular expression in "code".

FoomaticRIPOptionMaxLength

The attribute has the following parameter usage:

*FoomaticRIPOptionMaxLength name: length

For string or password options, this attribute sets a maximum length which can be returned. The "name" parameter identifies the option, and the "length" parameter is the maximum number of characters allowed.

FoomaticRIPOptionPrototype

The attribute has the following parameter usage:

*FoomaticRIPOptionPrototype name: "code"

For string, password, or simulated numeric options, this attribute sets a code prototype to be inserted into the output. This works for options where the FoomaticRIPOption "style" parameter is set to CmdLine, JCL, or PS. The value of the option can be represented with the string "%s" in the "code" parameter.

FoomaticRIPOptionRange

The attribute has the following parameter usage:

*FoomaticRIPOptionRange name: min max

This attribute adds a minimux and maximum limit to numeric options (that are simulated by Foomatic via emumerated options). The "name" parameter identifies the option, while the "min" and "max" parameters set the minumum and maximum allowed values, respectively, for the option.

FoomaticRIPOptionSetting

The attribute has the following parameter usage:

*FoomaticRIPOptionSetting name=choice: "code"

This attribute adds code to an option, identified by "name", with a FoomaticRIPOption "style" parameter set to Composite. It inserts options for other options that are members of the Composite option "name".

FoomaticRIPPostPipe

The attribute has the following parameter usage:

*FoomaticRIPPostPipe "code"

This attribute defines the command line in the "code" parameter for the job output command used by foomatic-rip in standalone mode. The command should take printer-native data on standard input. The "code" parameter should include the preceding shell pipe symbol ("|").

II LSB Printing Libraries

7 Libraries

7.1 Interfaces for libcups

Table 7-1 defines the library name and shared object name for the libcups library

Table 7-1 libcups Definition

Library:	libcups
SONAME:	libcups.so.2

The behavior of the interfaces in this library is specified by the following specifications:

[CUPS 1.2] CUPS API Reference
[LSB] This Specification

7.1.1 CUPS Convenience ABI

7.1.1.1 Interfaces for CUPS Convenience ABI

An LSB conforming implementation shall provide the generic functions for CUPS Convenience ABI specified in Table 7-2, with the full mandatory functionality as described in the referenced underlying specification.

Table 7-2 libcups - CUPS Convenience ABI Function Interfaces

cupsAddDest [LSB]	cupsAddOption [LSB]	cupsCancelJob [LSB]	cupsDoAuthentication [CUPS 1.2]
cupsDoFileRequest [CUPS 1.2]	cupsEncodeOptions [CUPS 1.2]	cupsEncryption [LSB]	cupsFreeDests [LSB]
cupsFreeJobs [LSB]	cupsFreeOptions [LSB]	cupsGetDefault [LSB]	cupsGetDefault2 [CUPS 1.2]
cupsGetDest [LSB]	cupsGetDests [LSB]	cupsGetDests2 [CUPS 1.2]	cupsGetFd [CUPS 1.2]
cupsGetFile [CUPS 1.2]	cupsGetJobs [LSB]	cupsGetJobs2 [CUPS 1.2]	cupsGetOption [LSB]
cupsGetPPD [LSB]	cupsGetPPD2 [CUPS 1.2]	cupsGetPassword [LSB]	cupsLangEncoding [LSB]
cupsLangFlush [LSB]	cupsLangFree [LSB]	cupsLangGet [LSB]	cupsLastError [LSB]
cupsMarkOptions [LSB]	cupsParseOptions [LSB]	cupsPrintFile [LSB]	cupsPrintFile2 [CUPS 1.2]
cupsPrintFiles [LSB]	cupsPrintFiles2 [CUPS 1.2]	cupsPutFd [CUPS 1.2]	cupsPutFile [CUPS 1.2]
cupsServer [LSB]	cupsSetDests [LSB]	cupsSetDests2 [CUPS 1.2]	cupsSetEncryption [LSB]
cupsSetPassword	cupsSetServer	cupsSetUser	cupsTempFd

CB [LSB]	[LSB]	[LSB]	[LSB]
cupsUser [LSB]	httpBlocking [CUPS 1.2]	httpCheck [CUPS 1.2]	httpClearCookie [CUPS 1.2]
httpClearFields [CUPS 1.2]	httpClose [CUPS 1.2]	httpConnect [CUPS 1.2]	httpConnectEncrypt [CUPS 1.2]
httpDecode64_2 [CUPS 1.2]	httpDelete [CUPS 1.2]	httpEncode64_2 [CUPS 1.2]	httpEncryption [CUPS 1.2]
httpError [CUPS 1.2]	httpFlush [CUPS 1.2]	httpGet [CUPS 1.2]	httpGetCookie [CUPS 1.2]
httpGetDateString [CUPS 1.2]	httpGetDateTime [CUPS 1.2]	httpGetField [CUPS 1.2]	httpGetHostByName [CUPS 1.2]
httpGetSubField [CUPS 1.2]	httpGets [CUPS 1.2]	httpHead [CUPS 1.2]	httpInitialize [CUPS 1.2]
httpMD5 [CUPS 1.2]	httpMD5Final [CUPS 1.2]	httpMD5String [CUPS 1.2]	httpOptions [CUPS 1.2]
httpPost [CUPS 1.2]	httpPut [CUPS 1.2]	httpReconnect [CUPS 1.2]	httpSetCookie [CUPS 1.2]
httpSetField [CUPS 1.2]	httpStatus [CUPS 1.2]	httpTrace [CUPS 1.2]	httpUpdate [CUPS 1.2]
httpWait [CUPS 1.2]	ippAddBoolean [CUPS 1.2]	ippAddBooleans [CUPS 1.2]	ippAddCollection [CUPS 1.2]
ippAddCollections [CUPS 1.2]	ippAddDate [CUPS 1.2]	ippAddInteger [CUPS 1.2]	ippAddIntegers [CUPS 1.2]
ippAddRange [CUPS 1.2]	ippAddRanges [CUPS 1.2]	ippAddResolution [CUPS 1.2]	ippAddResolutions [CUPS 1.2]
ippAddSeparator [CUPS 1.2]	ippAddString [CUPS 1.2]	ippAddStrings [CUPS 1.2]	ippDateToTime [CUPS 1.2]
ippDelete [CUPS 1.2]	ippDeleteAttribute [CUPS 1.2]	ippErrorString [CUPS 1.2]	ippFindAttribute [CUPS 1.2]
ippFindNextAttribute [CUPS 1.2]	ippLength [CUPS 1.2]	ippNew [CUPS 1.2]	ippPort [CUPS 1.2]
ippRead [CUPS 1.2]	ippReadFile [CUPS 1.2]	ippSetPort [CUPS 1.2]	ippTimeToDate [CUPS 1.2]
ippWrite [CUPS 1.2]	ippWriteFile [CUPS 1.2]	ppdClose [LSB]	ppdCollect [LSB]
ppdConflicts [LSB]	ppdEmit [LSB]	ppdEmitFd [LSB]	ppdEmitJCL [LSB]
ppdErrorString [LSB]	ppdFindAttr [LSB]	ppdFindChoice [LSB]	ppdFindMarkedChoice [LSB]
ppdFindNextAttr [LSB]	ppdFindOption [LSB]	ppdIsMarked [LSB]	ppdLastError [LSB]
ppdMarkDefault	ppdMarkOption	ppdOpen [LSB]	ppdOpenFd

s [LSB]	[LSB]		[LSB]
ppdOpenFile [LSB]	ppdPageLength [LSB]	ppdPageSize [LSB]	ppdPageWidth [LSB]
ppdSetConformance [LSB]			

7.2 Data Definitions for libcups

This section defines global identifiers and their values that are associated with interfaces contained in libcups. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and application developers should use this ABI to supplement - not to replace - source interface definition specifications.

This specification uses the ISO C (1999) C Language as the reference programming language, and data definitions are specified in ISO C format. The C language is used here as a convenient notation. Using a C language description of these data objects does not preclude their use by other programming languages.

7.2.1 cups/cups.h

```
#define _CUPS_CUPS_H_
#define CUPS_VERSION_MAJOR      1
#define CUPS_VERSION_MINOR      1
#define CUPS_VERSION    1.0123
#define CUPS_VERSION_PATCH      23
#define cupsLangDefault()       cupsLangGet(NULL)

typedef enum {
    CUPS_AUTO_ENCODING = -1,
    CUPS_US_ASCII = 0,
    CUPS_ISO8859_1 = 1,
    CUPS_ISO8859_2 = 2,
    CUPS_ISO8859_3 = 3,
    CUPS_ISO8859_4 = 4,
    CUPS_ISO8859_5 = 5,
    CUPS_ISO8859_6 = 6,
    CUPS_ISO8859_7 = 7,
    CUPS_ISO8859_8 = 8,
    CUPS_ISO8859_9 = 9,
    CUPS_ISO8859_10 = 10,
    CUPS_UTF8 = 11,
    CUPS_ISO8859_13 = 12,
    CUPS_ISO8859_14 = 13,
    CUPS_ISO8859_15 = 14,
    CUPS_WINDOWS_874 = 15,
    CUPS_WINDOWS_1250 = 16,
    CUPS_WINDOWS_1251 = 17,
    CUPS_WINDOWS_1252 = 18,
    CUPS_WINDOWS_1253 = 19,
    CUPS_WINDOWS_1254 = 20,
    CUPS_WINDOWS_1255 = 21,
    CUPS_WINDOWS_1256 = 22,
    CUPS_WINDOWS_1257 = 23,
    CUPS_WINDOWS_1258 = 24,
    CUPS_KOI8_R = 25,
    CUPS_KOI8_U = 26
} cups_encoding_t;
typedef struct cups_lang_s {
    struct cups_lang_s *next;
    int used;
    cups_encoding_t encoding;
    char language[16];
    cups_array_t *strings;
} cups_lang_t;
typedef enum {
    HTTP_ENCRYPT_IF_REQUESTED = 0,
    HTTP_ENCRYPT_NEVER = 1,
    HTTP_ENCRYPT_REQUIRED = 2,
    HTTP_ENCRYPT_ALWAYS = 3
} http_encryption_t;
typedef struct {
    char *name;
    char *value;
} cups_option_t;
typedef struct {
    char *name;
    char *instance;
    int is_default;
    int num_options;
    cups_option_t *options;
} cups_dest_t;
typedef enum {
    HTTP_WAITING = 0,
    HTTP_OPTIONS = 1,
    HTTP_GET = 2,
    HTTP_GET_SEND = 3,
    HTTP_HEAD = 4,
    HTTP_POST = 5,
    HTTP_POST_RECV = 6,
    HTTP_POST_SEND = 7,
    HTTP_PUT = 8,
    HTTP_PUT_RECV = 9,
    HTTP_DELETE = 10,
    HTTP_TRACE = 11,
    HTTP_CLOSE = 12,
    HTTP_STATUS = 13
} http_state_t;
typedef enum {
    HTTP_ERROR = -1,
    HTTP_CONTINUE = 100,
    HTTP_SWITCHING_PROTOCOLS = 101,
    HTTP_OK = 200,
    HTTP_CREATED = 201,
    HTTP_ACCEPTED = 202,
    HTTP_NOT_AUTHORITATIVE = 203,
    HTTP_NO_CONTENT = 204,
    HTTP_RESET_CONTENT = 205,
    HTTP_PARTIAL_CONTENT = 206,
    HTTP_MULTIPLE_CHOICES = 300,
    HTTP_MOVED_PERMANENTLY = 301,
    HTTP_MOVED_TEMPORARILY = 302,
    HTTP_SEE_OTHER = 303,
    HTTP_NOT_MODIFIED = 304,
    HTTP_USE_PROXY = 305,
    HTTP_BAD_REQUEST = 400,
    HTTP_UNAUTHORIZED = 401,
    HTTP_PAYMENT_REQUIRED = 402,
    HTTP_FORBIDDEN = 403,
    HTTP_NOT_FOUND = 404,
    HTTP_METHOD_NOT_ALLOWED = 405,
    HTTP_NOT_ACCEPTABLE = 406,
    HTTP_PROXY_AUTHENTICATION = 407,
    HTTP_REQUEST_TIMEOUT = 408,
    HTTP_CONFLICT = 409,
    HTTP_GONE = 410,
    HTTP_LENGTH_REQUIRED = 411,
    HTTP_PRECONDITION = 412,
    HTTP_REQUEST_TOO_LARGE = 413,
    HTTP_URI_TOO_LONG = 414,
    HTTP_UNSUPPORTED_MEDIATYPE = 415,
    HTTP_UPGRADE_REQUIRED = 426,
    HTTP_SERVER_ERROR = 500,
    HTTP_NOT_IMPLEMENTED = 501,
    HTTP_BAD_GATEWAY = 502,
    HTTP_SERVICE_UNAVAILABLE = 503,
    HTTP_GATEWAY_TIMEOUT = 504,
    HTTP_NOT_SUPPORTED = 505
} http_status_t;
typedef enum {
    HTTP_0_9 = 9,
    HTTP_1_0 = 100,
    HTTP_1_1 = 101
} http_version_t;
typedef enum {
    HTTP_KEEPALIVE_OFF = 0,
    HTTP_KEEPALIVE_ON = 1
} http_keepalive_t;
typedef enum {
    HTTP_ENCODE_LENGTH = 0,
    HTTP_ENCODE_CHUNKED = 1
} http_encoding_t;
typedef enum {
    IPP_JOB_PENDING = 3,
    IPP_JOB_HELD = 4,
    IPP_JOB_PROCESSING = 5,
    IPP_JOB_STOPPED = 6,
    IPP_JOB_CANCELLED = 7,
    IPP_JOB_ABORTED = 8,
    IPP_JOB_COMPLETED = 9
} ipp_jstate_t;
typedef struct {
    int id;
    char *dest;
    char *title;
    char *user;
    char *format;
    ipp_jstate_t state;
    int size;
    int priority;
    time_t completed_time;
    time_t creation_time;
    time_t processing_time;
} cups_job_t;
typedef struct _cups_array_s cups_array_t;

typedef struct _http_s http_t;
extern int cupsAddDest(const char *name, const char *instance,
                       int num_dests, cups_dest_t * *dests);
extern int cupsAddOption(const char *name, const char *value,
                         int    num_options,    cups_option_t    *
*options);
extern int cupsCancelJob(const char *printer, int job);
extern   int   cupsDoAuthentication(http_t   *   http,   const   char
*method,
                                const char *resource);
extern ipp_t *cupsDoFileRequest(http_t * http, ipp_t * request,
                                const char *resource,
                                const char *filename);
extern void cupsEncodeOptions(ipp_t * ipp, int num_options,
                              cups_option_t * options);
extern http_encryption_t cupsEncryption(void);
extern void cupsFreeDests(int num_dests, cups_dest_t * dests);
extern void cupsFreeJobs(int num_jobs, cups_job_t * jobs);
extern  void  cupsFreeOptions(int  num_options,  cups_option_t  *
options);
extern const char *cupsGetDefault(void);
extern const char *cupsGetDefault2(http_t * http);
extern  cups_dest_t  *cupsGetDest(const  char  *name,  const  char
*instance,
                                int   num_dests,   cups_dest_t   *
dests);
extern int cupsGetDests(cups_dest_t * *dests);
extern int cupsGetDests2(http_t * http, cups_dest_t * *dests);
extern   http_status_t   cupsGetFd(http_t   *   http,   const   char
*resource,
                               int fd);
extern   http_status_t   cupsGetFile(http_t   *   http,   const   char
*resource,
                                 const char *filename);
extern int cupsGetJobs(cups_job_t * *jobs, const char *dest, int
myjobs,
                       int completed);
extern int cupsGetJobs2(http_t * http, cups_job_t * *jobs,
                        const   char   *dest,   int   myjobs,   int
completed);
extern   const   char   *cupsGetOption(const   char   *name,   int
num_options,
                                 cups_option_t * options);
extern const char *cupsGetPPD(const char *printer);
extern   const   char   *cupsGetPPD2(http_t   *   http,   const   char
*printer);
extern const char *cupsGetPassword(const char *prompt);
extern const char *cupsLangEncoding(cups_lang_t * lang);
extern void cupsLangFlush(void);
extern void cupsLangFree(cups_lang_t * lang);
extern cups_lang_t *cupsLangGet(const char *language);
extern ipp_status_t cupsLastError(void);
extern int cupsMarkOptions(ppd_file_t * ppd, int num_options,
                           cups_option_t * options);
extern int cupsParseOptions(const char *arg, int num_options,
                            cups_option_t * *options);
extern   int   cupsPrintFile(const   char   *printer,   const   char
*filename,
                         const char *title, int num_options,
                         cups_option_t * options);
extern int cupsPrintFile2(http_t * http, const char *printer,
                          const   char   *filename,   const   char
*title,
                          int    num_options,    cups_option_t    *
options);
extern int cupsPrintFiles(const char *printer, int num_files,
                          const char **files, const char *title,
                          int    num_options,    cups_option_t    *
options);
extern int cupsPrintFiles2(http_t * http, const char *printer,
                           int num_files, const char **files,
                           const char *title, int num_options,
                           cups_option_t * options);
extern   http_status_t   cupsPutFd(http_t   *   http,   const   char
*resource,
                               int fd);
extern  http_status_t  cupsPutFile(http_t  *  http,  const  char
*resource,
                                 const char *filename);
extern const char *cupsServer(void);
extern void cupsSetDests(int num_dests, cups_dest_t * dests);
extern int cupsSetDests2(http_t * http, int num_dests,
                         cups_dest_t * dests);
extern void cupsSetEncryption(http_encryption_t e);
extern void cupsSetPasswordCB(const char *(*cb) (const char *));
extern void cupsSetServer(const char *server);
extern void cupsSetUser(const char *user);
extern int cupsTempFd(char *filename, int len);
extern const char *cupsUser(void);
```

7.2.2 cups/http.h

```
#define HTTP_MAX_URI    1024
#define HTTP_MAX_BUFFER 2048
#define HTTP_MAX_HOST   256
#define HTTP_MAX_VALUE  256

typedef enum http_auth_e {
    HTTP_AUTH_NONE,
    HTTP_AUTH_BASIC,
    HTTP_AUTH_MD5,
    HTTP_AUTH_MD5_SESS,
    HTTP_AUTH_MD5_INT,
    HTTP_AUTH_MD5_SESS_INT,
    HTTP_AUTH_NEGOTIATE
} http_auth_t;
typedef enum http_field_e {
    HTTP_FIELD_UNKNOWN,
    HTTP_FIELD_ACCEPT_LANGUAGE,
    HTTP_FIELD_ACCEPT_RANGES,
    HTTP_FIELD_AUTHORIZATION,
    HTTP_FIELD_CONNECTION,
    HTTP_FIELD_CONTENT_ENCODING,
    HTTP_FIELD_CONTENT_LANGUAGE,
    HTTP_FIELD_CONTENT_LENGTH,
    HTTP_FIELD_CONTENT_LOCATION,
    HTTP_FIELD_CONTENT_MD5,
    HTTP_FIELD_CONTENT_RANGE,
    HTTP_FIELD_CONTENT_TYPE,
    HTTP_FIELD_CONTENT_VERSION,
    HTTP_FIELD_DATE,
    HTTP_FIELD_HOST,
    HTTP_FIELD_IF_MODIFIED_SINCE,
    HTTP_FIELD_IF_UNMODIFIED_SINCE,
    HTTP_FIELD_KEEP_ALIVE,
    HTTP_FIELD_LAST_MODIFIED,
    HTTP_FIELD_LINK,
    HTTP_FIELD_LOCATION,
    HTTP_FIELD_RANGE,
    HTTP_FIELD_REFERER,
    HTTP_FIELD_RETRY_AFTER,
    HTTP_FIELD_TRANSFER_ENCODING,
    HTTP_FIELD_UPGRADE,
    HTTP_FIELD_USER_AGENT,
    HTTP_FIELD_WWW_AUTHENTICATE,
    HTTP_FIELD_MAX
} http_field_t;
typedef enum http_uri_status_e {
    HTTP_URI_OVERFLOW,
    HTTP_URI_BAD_ARGUMENTS,
    HTTP_URI_BAD_RESOURCE,
    HTTP_URI_BAD_PORT,
    HTTP_URI_BAD_HOSTNAME,
    HTTP_URI_BAD_USERNAME,
    HTTP_URI_BAD_SCHEME,
    HTTP_URI_BAD_URI,
    HTTP_URI_OK,
    HTTP_URI_MISSING_SCHEME,
    HTTP_URI_UNKNOWN_SCHEME,
    HTTP_URI_MISSING_RESOURCE
} http_uri_status_t;
typedef enum http_uri_coding_e {
    HTTP_URI_CODING_NONE,
    HTTP_URI_CODING_USERNAME,
    HTTP_URI_CODING_HOSTNAME,
    HTTP_URI_CODING_RESOURCE,
    HTTP_URI_CODING_MOST,
    HTTP_URI_CODING_QUERY,
    HTTP_URI_CODING_ALL
} http_uri_coding_t;
typedef union _http_addr_u {
    struct sockaddr addr;
    struct sockaddr_in ipv4;
    struct sockaddr_in6 ipv6;
    struct sockaddr_un un;
    char pad[256];
} http_addr_t;
typedef struct http_addrlist_s {
    struct http_addrlist_s *next;
    http_addr_t addr;
} http_addrlist_t;
extern void httpBlocking(http_t * http, int b);
extern int httpCheck(http_t * http);
extern void httpClearCookie(http_t * http);
extern void httpClearFields(http_t * http);
extern void httpClose(http_t * http);
extern http_t *httpConnect(const char *host, int port);
extern http_t *httpConnectEncrypt(const char *host, int port,
                                  http_encryption_t encryption);
extern  char  *httpDecode64_2(char  *out,  int  *outlen,  const  char
*in);
extern int httpDelete(http_t * http, const char *uri);
extern  char  *httpEncode64_2(char  *out,  int  outlen,  const  char
*in,
                            int inlen);
extern int httpEncryption(http_t * http, http_encryption_t e);
extern int httpError(http_t * http);
extern void httpFlush(http_t * http);
extern int httpGet(http_t * http, const char *uri);
extern const char *httpGetCookie(http_t * http);
extern const char *httpGetDateString(time_t t);
extern time_t httpGetDateTime(const char *s);
extern  const  char  *httpGetField(http_t  *  http,  http_field_t
field);
extern struct hostent *httpGetHostByName(const char *name);
extern char *httpGetSubField(http_t * http, http_field_t field,
                             const char *name, char *value);
extern char *httpGets(char *line, int length, http_t * http);
extern int httpHead(http_t * http, const char *uri);
extern void httpInitialize(void);
extern char *httpMD5(const char *, const char *, const char *,
char *);
extern char *httpMD5Final(const char *, const char *, const char
*,
                          char *);
extern char *httpMD5String(const unsigned char *, char *);
extern int httpOptions(http_t * http, const char *uri);
extern int httpPost(http_t * http, const char *uri);
extern int httpPut(http_t * http, const char *uri);
extern int httpReconnect(http_t * http);
extern void httpSetCookie(http_t * http, const char *cookie);
extern void httpSetField(http_t * http, http_field_t field,
                         const char *value);
extern const char *httpStatus(http_status_t status);
extern int httpTrace(http_t * http, const char *uri);
extern http_status_t httpUpdate(http_t * http);
extern int httpWait(http_t * http, int msec);
```

7.2.3 cups/ipp.h

```
#define IPP_MAX_NAME    256
#define IPP_MAX_LENGTH  32767
#define IPP_PORT        631
#define IPP_MAX_VALUES  8
#define CUPS_ADD_CLASS  CUPS_ADD_MODIFY_CLASS
#define CUPS_ADD_PRINTER        CUPS_ADD_MODIFY_PRINTER
#define IPP_ERROR_JOB_CANCELLED IPP_ERROR_JOB_CANCELED
#define IPP_JOB_CANCELLED       IPP_JOB_CANCELED
#define IPP_VERSION     "\001\001"

typedef enum {
    IPP_OK = 0,
    IPP_OK_SUBST = 1,
    IPP_OK_CONFLICT = 2,
    IPP_OK_IGNORED_SUBSCRIPTIONS = 3,
    IPP_OK_IGNORED_NOTIFICATIONS = 4,
    IPP_OK_TOO_MANY_EVENTS = 5,
    IPP_OK_BUT_CANCEL_SUBSCRIPTION = 6,
    IPP_REDIRECTION_OTHER_SITE = 768,
    IPP_BAD_REQUEST = 1024,
    IPP_FORBIDDEN = 1025,
    IPP_NOT_AUTHENTICATED = 1026,
    IPP_NOT_AUTHORIZED = 1027,
    IPP_NOT_POSSIBLE = 1028,
    IPP_TIMEOUT = 1029,
    IPP_NOT_FOUND = 1030,
    IPP_GONE = 1031,
    IPP_REQUEST_ENTITY = 1032,
    IPP_REQUEST_VALUE = 1033,
    IPP_DOCUMENT_FORMAT = 1034,
    IPP_ATTRIBUTES = 1035,
    IPP_URI_SCHEME = 1036,
    IPP_CHARSET = 1037,
    IPP_CONFLICT = 1038,
    IPP_COMPRESSION_NOT_SUPPORTED = 1039,
    IPP_COMPRESSION_ERROR = 1040,
    IPP_DOCUMENT_FORMAT_ERROR = 1041,
    IPP_DOCUMENT_ACCESS_ERROR = 1042,
    IPP_ATTRIBUTES_NOT_SETTABLE = 1043,
    IPP_IGNORED_ALL_SUBSCRIPTIONS = 1044,
    IPP_TOO_MANY_SUBSCRIPTIONS = 1045,
    IPP_IGNORED_ALL_NOTIFICATIONS = 1046,
    IPP_PRINT_SUPPORT_FILE_NOT_FOUND = 1047,
    IPP_INTERNAL_ERROR = 1280,
    IPP_OPERATION_NOT_SUPPORTED = 1281,
    IPP_SERVICE_UNAVAILABLE = 1282,
    IPP_VERSION_NOT_SUPPORTED = 1283,
    IPP_DEVICE_ERROR = 1284,
    IPP_TEMPORARY_ERROR = 1285,
    IPP_NOT_ACCEPTING = 1286,
    IPP_PRINTER_BUSY = 1287,
    IPP_ERROR_JOB_CANCELLED = 1288,
    IPP_MULTIPLE_JOBS_NOT_SUPPORTED = 1289,
    IPP_PRINTER_IS_DEACTIVATED = 1290
} ipp_status_t;
typedef enum ipp_tag_e {
    IPP_TAG_ZERO,
    IPP_TAG_OPERATION,
    IPP_TAG_JOB,
    IPP_TAG_END,
    IPP_TAG_PRINTER,
    IPP_TAG_UNSUPPORTED_GROUP,
    IPP_TAG_SUBSCRIPTION,
    IPP_TAG_EVENT_NOTIFICATION,
    IPP_TAG_UNSUPPORTED_VALUE,
    IPP_TAG_DEFAULT,
    IPP_TAG_UNKNOWN,
    IPP_TAG_NOVALUE,
    IPP_TAG_NOTSETTABLE,
    IPP_TAG_DELETEATTR,
    IPP_TAG_ADMINDEFINE,
    IPP_TAG_INTEGER,
    IPP_TAG_BOOLEAN,
    IPP_TAG_ENUM,
    IPP_TAG_STRING,
    IPP_TAG_DATE,
    IPP_TAG_RESOLUTION,
    IPP_TAG_RANGE,
    IPP_TAG_BEGIN_COLLECTION,
    IPP_TAG_TEXTLANG,
    IPP_TAG_NAMELANG,
    IPP_TAG_END_COLLECTION,
    IPP_TAG_TEXT,
    IPP_TAG_NAME,
    IPP_TAG_KEYWORD,
    IPP_TAG_URI,
    IPP_TAG_URISCHEME,
    IPP_TAG_CHARSET,
    IPP_TAG_LANGUAGE,
    IPP_TAG_MIMETYPE,
    IPP_TAG_MEMBERNAME,
    IPP_TAG_MASK,
    IPP_TAG_COPY
} ipp_tag_t;
typedef enum ipp_res_e {
    IPP_RES_PER_INCH,
    IPP_RES_PER_CM
} ipp_res_t;
typedef enum ipp_finish_e {
    IPP_FINISHINGS_NONE,
    IPP_FINISHINGS_STAPLE,
    IPP_FINISHINGS_PUNCH,
    IPP_FINISHINGS_COVER,
    IPP_FINISHINGS_BIND,
    IPP_FINISHINGS_SADDLE_STITCH,
    IPP_FINISHINGS_EDGE_STITCH,
    IPP_FINISHINGS_FOLD,
    IPP_FINISHINGS_TRIM,
    IPP_FINISHINGS_BALE,
    IPP_FINISHINGS_BOOKLET_MAKER,
    IPP_FINISHINGS_JOB_OFFSET,
    IPP_FINISHINGS_STAPLE_TOP_LEFT,
    IPP_FINISHINGS_STAPLE_BOTTOM_LEFT,
    IPP_FINISHINGS_STAPLE_TOP_RIGHT,
    IPP_FINISHINGS_STAPLE_BOTTOM_RIGHT,
    IPP_FINISHINGS_EDGE_STITCH_LEFT,
    IPP_FINISHINGS_EDGE_STITCH_TOP,
    IPP_FINISHINGS_EDGE_STITCH_RIGHT,
    IPP_FINISHINGS_EDGE_STITCH_BOTTOM,
    IPP_FINISHINGS_STAPLE_DUAL_LEFT,
    IPP_FINISHINGS_STAPLE_DUAL_TOP,
    IPP_FINISHINGS_STAPLE_DUAL_RIGHT,
    IPP_FINISHINGS_STAPLE_DUAL_BOTTOM,
    IPP_FINISHINGS_BIND_LEFT,
    IPP_FINISHINGS_BIND_TOP,
    IPP_FINISHINGS_BIND_RIGHT,
    IPP_FINISHINGS_BIND_BOTTOM
} ipp_finish_t;
typedef enum ipp_orient_e {
    IPP_PORTRAIT,
    IPP_LANDSCAPE,
    IPP_REVERSE_LANDSCAPE,
    IPP_REVERSE_PORTRAIT
} ipp_orient_t;
typedef enum ipp_quality_e {
    IPP_QUALITY_DRAFT,
    IPP_QUALITY_NORMAL,
    IPP_QUALITY_HIGH
} ipp_quality_t;
typedef enum ipp_pstate_e {
    IPP_PRINTER_IDLE,
    IPP_PRINTER_PROCESSING,
    IPP_PRINTER_STOPPED
} ipp_pstate_t;
typedef enum ipp_state_e {
    IPP_ERROR,
    IPP_IDLE,
    IPP_HEADER,
    IPP_ATTRIBUTE,
    IPP_DATA
} ipp_state_t;
typedef enum ipp_op_e {
    IPP_PRINT_JOB,
    IPP_PRINT_URI,
    IPP_VALIDATE_JOB,
    IPP_CREATE_JOB,
    IPP_SEND_DOCUMENT,
    IPP_SEND_URI,
    IPP_CANCEL_JOB,
    IPP_GET_JOB_ATTRIBUTES,
    IPP_GET_JOBS,
    IPP_GET_PRINTER_ATTRIBUTES,
    IPP_HOLD_JOB,
    IPP_RELEASE_JOB,
    IPP_RESTART_JOB,
    IPP_PAUSE_PRINTER,
    IPP_RESUME_PRINTER,
    IPP_PURGE_JOBS,
    IPP_SET_PRINTER_ATTRIBUTES,
    IPP_SET_JOB_ATTRIBUTES,
    IPP_GET_PRINTER_SUPPORTED_VALUES,
    IPP_CREATE_PRINTER_SUBSCRIPTION,
    IPP_CREATE_JOB_SUBSCRIPTION,
    IPP_GET_SUBSCRIPTION_ATTRIBUTES,
    IPP_GET_SUBSCRIPTIONS,
    IPP_RENEW_SUBSCRIPTION,
    IPP_CANCEL_SUBSCRIPTION,
    IPP_GET_NOTIFICATIONS,
    IPP_SEND_NOTIFICATIONS,
    IPP_GET_PRINT_SUPPORT_FILES,
    IPP_ENABLE_PRINTER,
    IPP_DISABLE_PRINTER,
    IPP_PAUSE_PRINTER_AFTER_CURRENT_JOB,
    IPP_HOLD_NEW_JOBS,
    IPP_RELEASE_HELD_NEW_JOBS,
    IPP_DEACTIVATE_PRINTER,
    IPP_ACTIVATE_PRINTER,
    IPP_RESTART_PRINTER,
    IPP_SHUTDOWN_PRINTER,
    IPP_STARTUP_PRINTER,
    IPP_REPROCESS_JOB,
    IPP_CANCEL_CURRENT_JOB,
    IPP_SUSPEND_CURRENT_JOB,
    IPP_RESUME_JOB,
    IPP_PROMOTE_JOB,
    IPP_SCHEDULE_JOB_AFTER,
    IPP_PRIVATE,
    CUPS_GET_DEFAULT,
    CUPS_GET_PRINTERS,
    CUPS_ADD_MODIFY_PRINTER,
    CUPS_DELETE_PRINTER,
    CUPS_GET_CLASSES,
    CUPS_ADD_MODIFY_CLASS,
    CUPS_DELETE_CLASS,
    CUPS_ACCEPT_JOBS,
    CUPS_REJECT_JOBS,
    CUPS_SET_DEFAULT,
    CUPS_GET_DEVICES,
    CUPS_GET_PPDS,
    CUPS_MOVE_JOB,
    CUPS_AUTHENTICATE_JOB,
    CUPS_GET_PPD
} ipp_op_t;
typedef unsigned char ipp_uchar_t;
typedef union ipp_request_u {
    struct {
        ipp_uchar_t version[2];
        int op_status;
        int request_id;
    } any;
    struct {
        ipp_uchar_t version[2];
        ipp_op_t operation_id;
        int request_id;
    } op;
    struct {
        ipp_uchar_t version[2];
        ipp_status_t status_code;
        int request_id;
    } status;
    struct {
        ipp_uchar_t version[2];
        ipp_status_t status_code;
        int request_id;
    } event;
} ipp_request_t;
typedef struct ipp_s {
    ipp_state_t state;
    ipp_request_t request;
    ipp_attribute_t *attrs;
    ipp_attribute_t *last;
    ipp_attribute_t *current;
    ipp_tag_t curtag;
} ipp_t;
typedef union ipp_value_u {
    int integer;
    char boolean;
    ipp_uchar_t date[11];
    struct {
        int xres;
        int yres;
        ipp_res_t units;
    } resolution;
    struct {
        int lower;
        int upper;
    } range;
    struct {
        char *charset;
        char *text;
    } string;
    struct {
        int length;
        void *data;
    } unknown;
    ipp_t *collection;
} ipp_value_t;
typedef struct ipp_attribute_s {
    struct ipp_attribute_s *next;
    ipp_tag_t group_tag;
    ipp_tag_t value_tag;
    char *name;
    int num_values;
    ipp_value_t values[1];
} ipp_attribute_t;
extern  ipp_attribute_t  *ippAddBoolean(ipp_t  *  ipp,  ipp_tag_t
group,
                                     const   char   *name,   char
value);
extern  ipp_attribute_t  *ippAddBooleans(ipp_t  *  ipp,  ipp_tag_t
group,
                                      const   char   *name,   int
num_values,
                                      const char *values);
extern  ipp_attribute_t  *ippAddCollection(ipp_t  *  ipp,  ipp_tag_t
group,
                                        const  char  *name,  ipp_t
* value);
extern  ipp_attribute_t  *ippAddCollections(ipp_t  *  ipp,  ipp_tag_t
group,
                                         const  char  *name,  int
num_values,
                                         const ipp_t * *values);
extern ipp_attribute_t *ippAddDate(ipp_t * ipp, ipp_tag_t group,
                                  const char *name,
                                  const ipp_uchar_t * value);
extern  ipp_attribute_t  *ippAddInteger(ipp_t  *  ipp,  ipp_tag_t
group,
                                     ipp_tag_t  type,  const  char
*name,
                                     int value);
extern  ipp_attribute_t  *ippAddIntegers(ipp_t  *  ipp,  ipp_tag_t
group,
                                      ipp_tag_t type, const char
*name,
                                      int  num_values,  const  int
*values);
extern ipp_attribute_t *ippAddRange(ipp_t * ipp, ipp_tag_t group,
                                   const char *name, int lower,
                                   int upper);
extern  ipp_attribute_t  *ippAddRanges(ipp_t  *  ipp,  ipp_tag_t
group,
                                    const   char   *name,   int
num_values,
                                     const int *lower, const int
*upper);
extern ipp_attribute_t *ippAddResolution(ipp_t * ipp, ipp_tag_t
group,
                                         const char *name,
ipp_res_t units,
                                         int xres, int yres);
extern ipp_attribute_t *ippAddResolutions(ipp_t * ipp, ipp_tag_t
group,
                                          const char *name, int
num_values,
                                          ipp_res_t units, const
int *xres,
                                          const int *yres);
extern ipp_attribute_t *ippAddSeparator(ipp_t * ipp);
extern ipp_attribute_t *ippAddString(ipp_t * ipp, ipp_tag_t
group,
                                     ipp_tag_t type, const char
*name,
                                     const char *charset,
                                     const char *value);
extern ipp_attribute_t *ippAddStrings(ipp_t * ipp, ipp_tag_t
group,
                                      ipp_tag_t type, const char
*name,
                                      int num_values, const char
*charset,
                                      const char *const *values);
extern time_t ippDateToTime(const ipp_uchar_t * date);
extern void ippDelete(ipp_t * ipp);
extern void ippDeleteAttribute(ipp_t * ipp, ipp_attribute_t *
attr);
extern const char *ippErrorString(ipp_status_t error);
extern ipp_attribute_t *ippFindAttribute(ipp_t * ipp, const char
*name,
                                         ipp_tag_t type);
extern ipp_attribute_t *ippFindNextAttribute(ipp_t * ipp, const
char *name,
                                             ipp_tag_t type);
extern size_t ippLength(ipp_t * ipp);
extern ipp_t *ippNew(void);
extern int ippPort(void);
extern ipp_state_t ippRead(http_t * http, ipp_t * ipp);
extern ipp_state_t ippReadFile(int fd, ipp_t * ipp);
extern void ippSetPort(int p);
extern const ipp_uchar_t *ippTimeToDate(time_t t);
extern ipp_state_t ippWrite(http_t * http, ipp_t * ipp);
extern ipp_state_t ippWriteFile(int fd, ipp_t * ipp);
```

7.2.4 cups/ppd.h

```
#define _CUPS_PPD_H_
#define PPD_MAX_LINE    256
#define PPD_VERSION     4.3
#define PPD_MAX_NAME    41
#define PPD_MAX_TEXT    81

typedef enum {
    PPD_CS_CMYK = -4,
    PPD_CS_CMY = -3,
    PPD_CS_GRAY = 1,
    PPD_CS_RGB = 3,
    PPD_CS_RGBK = 4,
    PPD_CS_N = 5
} ppd_cs_t;
```

```
typedef struct {
    char name[41];
    char *start;
    char *stop;
} ppd_emul_t;
typedef enum {
    PPD_UI_BOOLEAN = 0,
    PPD_UI_PICKONE = 1,
    PPD_UI_PICKMANY = 2
} ppd_ui_t;
typedef enum {
    PPD_ORDER_ANY = 0,
    PPD_ORDER_DOCUMENT = 1,
    PPD_ORDER_EXIT = 2,
    PPD_ORDER_JCL = 3,
    PPD_ORDER_PAGE = 4,
    PPD_ORDER_PROLOG = 5
} ppd_section_t;
typedef struct {
    char marked;
    char choice[41];
    char text[81];
    char *code;
    void *option;
} ppd_choice_t;
typedef struct {
    char conflicted;
    char keyword[41];
    char defchoice[41];
    char text[81];
    ppd_ui_t ui;
    ppd_section_t section;
    float order;
    int num_choices;
    ppd_choice_t *choices;
} ppd_option_t;
typedef struct ppd_group_str {
    char text[40];
    char name[41];
    int num_options;
    ppd_option_t *options;
    int num_subgroups;
    struct ppd_group_str *subgroups;
} ppd_group_t;
typedef struct {
    int marked;
    char name[41];
    float width;
    float length;
    float left;
    float bottom;
    float right;
    float top;
} ppd_size_t;
typedef struct {
    char option1[41];
    char choice1[41];
    char option2[41];
    char choice2[41];
} ppd_const_t;
typedef struct {
    char resolution[41];
    char media_type[41];
    float density;
    float gamma;
    float matrix[3][3];
```

```
} ppd_profile_t;
typedef struct {
    char name[41];
    char spec[41];
    char text[81];
    char *value;
} ppd_attr_t;
typedef struct {
    int language_level;
    int color_device;
    int variable_sizes;
    int accurate_screens;
    int contone_only;
    int landscape;
    int model_number;
    int manual_copies;
    int throughput;
    ppd_cs_t colorspace;
    char *patches;
    int num_emulations;
    ppd_emul_t *emulations;
    char *jcl_begin;
    char *jcl_ps;
    char *jcl_end;
    char *lang_encoding;
    char *lang_version;
    char *modelname;
    char *ttrasterizer;
    char *manufacturer;
    char *product;
    char *nickname;
    char *shortnickname;
    int num_groups;
    ppd_group_t *groups;
    int num_sizes;
    ppd_size_t *sizes;
    float custom_min[2];
    float custom_max[2];
    float custom_margins[4];
    int num_consts;
    ppd_const_t *consts;
    int num_fonts;
    char **fonts;
    int num_profiles;
    ppd_profile_t *profiles;
    int num_filters;
    char **filters;
    int flip_duplex;
    char *protocols;
    char *pcfilename;
    int num_attrs;
    int cur_attr;
    ppd_attr_t **attrs;
} ppd_file_t;
typedef enum {
    PPD_OK = 0,
    PPD_FILE_OPEN_ERROR = 1,
    PPD_NULL_FILE = 2,
    PPD_ALLOC_ERROR = 3,
    PPD_MISSING_PPDADOBE4 = 4,
    PPD_MISSING_VALUE = 5,
    PPD_INTERNAL_ERROR = 6,
    PPD_BAD_OPEN_GROUP = 7,
    PPD_NESTED_OPEN_GROUP = 8,
    PPD_BAD_OPEN_UI = 9,
    PPD_NESTED_OPEN_UI = 10,
    PPD_BAD_ORDER_DEPENDENCY = 11,
    PPD_BAD_UI_CONSTRAINTS = 12,
    PPD_MISSING_ASTERISK = 13,
    PPD_LINE_TOO_LONG = 14,
    PPD_ILLEGAL_CHARACTER = 15,
    PPD_ILLEGAL_MAIN_KEYWORD = 16,
    PPD_ILLEGAL_OPTION_KEYWORD = 17,
    PPD_ILLEGAL_TRANSLATION = 18,
    PPD_ILLEGAL_WHITESPACE = 19
} ppd_status_t;
typedef enum {
    PPD_CONFORM_RELAXED = 0,
    PPD_CONFORM_STRICT = 1
} ppd_conform_t;
extern void ppdClose(ppd_file_t * ppd);
extern int ppdCollect(ppd_file_t * ppd, ppd_section_t section,
                      ppd_choice_t * **choices);
extern int ppdConflicts(ppd_file_t * ppd);
extern  int  ppdEmit(ppd_file_t  *  ppd,  FILE  *  fp,  ppd_section_t
section);
extern  int  ppdEmitFd(ppd_file_t  *  ppd,  int  fd,  ppd_section_t
section);
extern int ppdEmitJCL(ppd_file_t * ppd, FILE * fp, int job_id,
                      const char *user, const char *title);
extern const char *ppdErrorString(ppd_status_t status);
extern  ppd_attr_t  *ppdFindAttr(ppd_file_t  *  ppd,  const  char
*name,
                               const char *spec);
extern  ppd_choice_t  *ppdFindChoice(ppd_option_t  *  o,  const  char
*option);
extern ppd_choice_t *ppdFindMarkedChoice(ppd_file_t * ppd,
                                         const char *keyword);
extern  ppd_attr_t  *ppdFindNextAttr(ppd_file_t  *  ppd,  const  char
*name,
                                   const char *spec);
extern  ppd_option_t  *ppdFindOption(ppd_file_t  *  ppd,  const  char
*keyword);
extern int ppdIsMarked(ppd_file_t * ppd, const char *keyword,
                       const char *option);
extern ppd_status_t ppdLastError(int *line);
extern void ppdMarkDefaults(ppd_file_t * ppd);
extern int ppdMarkOption(ppd_file_t * ppd, const char *keyword,
                         const char *option);
extern ppd_file_t *ppdOpen(FILE * fp);
extern ppd_file_t *ppdOpenFd(int fd);
extern ppd_file_t *ppdOpenFile(const char *filename);
extern float ppdPageLength(ppd_file_t * ppd, const char *name);
extern  ppd_size_t  *ppdPageSize(ppd_file_t  *  ppd,  const  char
*name);
extern float ppdPageWidth(ppd_file_t * ppd, const char *name);
extern void ppdSetConformance(ppd_conform_t c);
```

7.3 Interface Definitions for libcups

The interfaces defined on the following pages are included in libcups and are defined by this specification. Unless otherwise noted, these interfaces shall be included in the source standard.

Other interfaces listed in Section 7.1 shall behave as described in the referenced base document.

cupsAddDest

Name

`cupsAddDest` — Add a destination to the list of destinations.

Synopsis

```
#include <cups/cups.h>
int cupsAddDest(const char * name, const char * instance, int
num_dests, cups_dest_t ** dests);
```

Description

Add a destination to the list of destinations.

This function cannot be used to add a new class or printer queue, it only adds a new container of saved options for the named destination or instance.

If the named destination already exists, the destination list is returned unchanged. Adding a new instance of a destination creates a copy of that destination's options.

Use the cupsSaveDests() function to save the updated list of destinations to the user's lpoptions file.

Return Value

New number of destinations

cupsAddOption

Name

`cupsAddOption` — Add an option to an option array.

Synopsis

```
#include <cups/cups.h>
int cupsAddOption(const char * name, const char * value, int
num_options, cups_option_t ** options);
```

Description

Add an option to an option array.

Return Value

Number of options

cupsCancelJob

Name

`cupsCancelJob` — Cancel a print job on the default server.

Synopsis

```
#include <cups/cups.h>
int cupsCancelJob(const char * name, int job);
```

Description

Cancel a print job on the default server.

Use the cupsLastError() and cupsLastErrorString() functions to get the cause of any failure.

Return Value

1 on success, 0 on failure

cupsEncryption

Name

`cupsEncryption` — Get the default encryption settings.

Synopsis

```
#include <cups/cups.h>
http_encryption_t cupsEncryption(void);
```

Description

Get the default encryption settings.

The default encryption setting comes from the CUPS_ENCRYPTION environment variable, then the ~/.cupsrc file, and finally the /etc/cups/client.conf file. If not set, the default is HTTP_ENCRYPT_IF_REQUESTED.

Return Value

Encryption settings

cupsFreeDests

Name

`cupsFreeDests` — Free the memory used by the list of destinations.

Synopsis

```
#include <cups/cups.h>
void cupsFreeDests(int num_dests, cups_dest_t * dests);
```

Description

Free the memory used by the list of destinations.

Return Value

This function does not return a value.

cupsFreeJobs

Name

`cupsFreeJobs` — Free memory used by job data.

Synopsis

```
#include <cups/cups.h>
void cupsFreeJobs(int num_jobs, cups_job_t * jobs);
```

Description

Free memory used by job data.

Return Value

This function does not return a value.

cupsFreeOptions

Name

`cupsFreeOptions` — Free all memory used by options.

Synopsis

```
#include <cups/cups.h>
void cupsFreeOptions(int num_options, cups_option_t * options);
```

Description

Free all memory used by options.

Return Value

This function does not return a value.

cupsGetDefault

Name

`cupsGetDefault` — Get the default printer or class for the default server.

Synopsis

```
#include <cups/cups.h>
const char * cupsGetDefault(void);
```

Description

Get the default printer or class for the default server.

This function returns the default printer or class as defined by the LPDEST or PRINTER environment variables. If these environment variables are not set, the server default destination is returned. Applications should use the cupsGetDests() and cupsGetDest() functions to get the user-defined default printer, as this function does not support the lpoptions-defined default printer.

Return Value

Default printer or NULL

cupsGetDest

Name

`cupsGetDest` — Get the named destination from the list.

Synopsis

```
#include <cups/cups.h>
cups_dest_t * cupsGetDest(const char * name, const char * instance,
int num_dests, cups_dest_t * dests);
```

Description

Get the named destination from the list.

Use the cupsGetDests() or cupsGetDests2() functions to get a list of supported destinations for the current user.

Return Value

Destination pointer or NULL

cupsGetDests

Name

`cupsGetDests` — Get the list of destinations from the default server.

Synopsis

```
#include <cups/cups.h>
int cupsGetDests(cups_dest_t ** dests);
```

Description

Get the list of destinations from the default server.

Starting with CUPS 1.2, the returned list of destinations include the printer-info, printer-is-accepting-jobs, printer-is-shared, printer-make-and-model, printer-state, printer-state-change-time, printer-state-reasons, and printer-type attributes as options.

Use the cupsFreeDests() function to free the destination list and the cupsGetDest() function to find a particular destination.

Return Value

Number of destinations

cupsGetJobs

Name

`cupsGetJobs` — Get the jobs from the default server.

Synopsis

```
#include <cups/cups.h>
int cupsGetJobs(cups_job_t ** jobs, const char * mydest, int myjobs,
int completed);
```

Description

Get the jobs from the default server.

Return Value

Number of jobs

cupsGetOption

Name

`cupsGetOption` — Get an option value.

Synopsis

```
#include <cups/cups.h>
const  char  *  cupsGetOption(const  char  *  name,  int  num_options,
cups_option_t * options);
```

Description

Get an option value.

Return Value

Option value or NULL

cupsGetPPD

Name

`cupsGetPPD` — Get the PPD file for a printer on the default server.

Synopsis

```
#include <cups/cups.h>
const char * cupsGetPPD(const char * name);
```

Description

Get the PPD file for a printer on the default server.

For classes, cupsGetPPD() returns the PPD file for the first printer in the class.

Return Value

Filename for PPD file

cupsGetPassword

Name

`cupsGetPassword` — Get a password from the user.

Synopsis

```
#include <cups/cups.h>
const char * cupsGetPassword(const char * prompt);
```

Description

Get a password from the user.

Uses the current password callback function. Returns NULL if the user does not provide a password.

Return Value

Password

cupsLangEncoding

Name

`cupsLangEncoding` — Return the character encoding (us-ascii, etc.)

Synopsis

```
#include <cups/cups.h>
const char * cupsLangEncoding(cups_lang_t * lang);
```

Description

Return the character encoding (us-ascii, etc.) for the given language.

Return Value

Character encoding

cupsLangFlush

Name

`cupsLangFlush` — Flush all language data out of the cache.

Synopsis

```
#include <cups/cups.h>
void cupsLangFlush(void);
```

Description

Flush all language data out of the cache.

Return Value

This function does not return a value.

cupsLangFree

Name

`cupsLangFree` — Free language data.

Synopsis

```
#include <cups/cups.h>
void cupsLangFree(cups_lang_t * lang);
```

Description

Free language data.

This does not actually free anything; use cupsLangFlush() for that.

Return Value

This function does not return a value.

cupsLangGet

Name

`cupsLangGet` — Get a language.

Synopsis

```
#include <cups/cups.h>
cups_lang_t * cupsLangGet(const char * language);
```

Description

Get a language.

Return Value

Language data

cupsLastError

Name

`cupsLastError` — Return the last IPP status code.

Synopsis

```
#include <cups/cups.h>
ipp_status_t cupsLastError(void);
```

Description

Return the last IPP status code.

Return Value

IPP status code from last request

cupsMarkOptions

Name

`cupsMarkOptions` — Mark command-line options in a PPD file.

Synopsis

```
#include <cups/cups.h>
int cupsMarkOptions(ppd_file_t * ppd, int num_options, cups_option_t
* options);
```

Description

Mark command-line options in a PPD file.

Return Value

1 if conflicting

cupsParseOptions

Name

`cupsParseOptions` — Parse options from a command-line argument.

Synopsis

```
#include <cups/cups.h>
int cupsParseOptions(const char * arg, int num_options, cups_option_t
** options);
```

Description

Parse options from a command-line argument.

This function converts space-delimited name/value pairs according to the PAPI text option ABNF specification. Collection values ("name={a=... b=... c=...}") are stored with the curley brackets intact - use cupsParseOptions() on the value to extract the collection attributes.

Return Value

Number of options found

cupsPrintFile

Name

`cupsPrintFile` — Print a file to a printer or class on the default server.

Synopsis

```
#include <cups/cups.h>
int cupsPrintFile(const char * name, const char * filename, const
char * title, int num_options, cups_option_t * options);
```

Description

Print a file to a printer or class on the default server.

Return Value

Job ID

cupsPrintFiles

Name

`cupsPrintFiles` — Print one or more files to a printer or class on the

Synopsis

```
#include <cups/cups.h>
int cupsPrintFiles(const char * name, int num_files, const char **
files, const char * title, int num_options, cups_option_t *
options);
```

Description

Print one or more files to a printer or class on the default server.

Return Value

Job ID

cupsServer

Name

`cupsServer` — Return the hostname/address of the default server.

Synopsis

```
#include <cups/cups.h>
const char * cupsServer(void);
```

Description

Return the hostname/address of the default server.

The returned value can be a fully-qualified hostname, a numeric IPv4 or IPv6 address, or a domain socket pathname.

Return Value

Server name

cupsSetDests

Name

`cupsSetDests` — Save the list of destinations for the default server.

Synopsis

```
#include <cups/cups.h>
void cupsSetDests(int num_dests, cups_dest_t * dests);
```

Description

Save the list of destinations for the default server.

This function saves the destinations to /etc/cups/lpoptions when run as root and ~/.cups/lpoptions when run as a normal user.

Return Value

This function does not return a value.

cupsSetEncryption

Name

`cupsSetEncryption` — Set the encryption preference.

Synopsis

```
#include <cups/cups.h>
void cupsSetEncryption(http_encryption_t e);
```

Description

Set the encryption preference.

Return Value

This function does not return a value.

cupsSetPasswordCB

Name

`cupsSetPasswordCB` — Set the password callback for CUPS.

Synopsis

```
#include <cups/cups.h>
void cupsSetPasswordCB(cups_password_cb_t cb);
```

Description

Set the password callback for CUPS.

Pass NULL to restore the default (console) password callback.

Return Value

This function does not return a value.

cupsSetServer

Name

`cupsSetServer` — Set the default server name.

Synopsis

```
#include <cups/cups.h>
void cupsSetServer(const char * server);
```

Description

Set the default server name.

The "server" string can be a fully-qualified hostname, a numeric IPv4 or IPv6 address, or a domain socket pathname. Pass NULL to restore the default server name.

Return Value

This function does not return a value.

cupsSetUser

Name

`cupsSetUser` — Set the default user name.

Synopsis

```
#include <cups/cups.h>
void cupsSetUser(const char * user);
```

Description

Set the default user name.

Pass NULL to restore the default user name.

Return Value

This function does not return a value.

cupsTempFd

Name

`cupsTempFd` — Creates a temporary file.

Synopsis

```
#include <cups/cups.h>
int cupsTempFd(char * filename, int len);
```

Description

Creates a temporary file.

The temporary filename is returned in the filename buffer. The temporary file is opened for reading and writing.

Return Value

New file descriptor or -1 on error

cupsUser

Name

`cupsUser` — Return the current user's name.

Synopsis

```
#include <cups/cups.h>
const char * cupsUser(void);
```

Description

Return the current user's name.

Return Value

User name

ppdClose

Name

`ppdClose` — Free all memory used by the PPD file.

Synopsis

```
#include <cups/cups.h>
void ppdClose(ppd_file_t * ppd);
```

Description

Free all memory used by the PPD file.

Return Value

This function does not return a value.

ppdCollect

Name

`ppdCollect` — Collect all marked options that reside in the specified

Synopsis

```
#include <cups/cups.h>
int   ppdCollect(ppd_file_t   *   ppd,   ppd_section_t   section,
ppd_choice_t *** choices);
```

Description

Collect all marked options that reside in the specified section.

Return Value

Number of options marked

ppdConflicts

Name

`ppdConflicts` — Check to see if there are any conflicts.

Synopsis

```
#include <cups/cups.h>
int ppdConflicts(ppd_file_t * ppd);
```

Description

Check to see if there are any conflicts.

Return Value

Number of conflicts found

ppdEmit

Name

`ppdEmit` — Emit code for marked options to a file.

Synopsis

```
#include <cups/cups.h>
int ppdEmit(ppd_file_t * ppd, FILE * fp, ppd_section_t section);
```

Description

Emit code for marked options to a file.

Return Value

0 on success, -1 on failure

ppdEmitFd

Name

`ppdEmitFd` — Emit code for marked options to a file.

Synopsis

```
#include <cups/cups.h>
int ppdEmitFd(ppd_file_t * ppd, int fd, ppd_section_t section);
```

Description

Emit code for marked options to a file.

Return Value

0 on success, -1 on failure

ppdEmitJCL

Name

`ppdEmitJCL` — Emit code for JCL options to a file.

Synopsis

```
#include <cups/cups.h>
int ppdEmitJCL(ppd_file_t * ppd, FILE * fp, int job_id, const char *
user, const char * title);
```

Description

Emit code for JCL options to a file.

Return Value

0 on success, -1 on failure

ppdErrorString

Name

`ppdErrorString` — Returns the text assocated with a status.

Synopsis

```
#include <cups/cups.h>
const char * ppdErrorString(ppd_status_t status);
```

Description

Returns the text assocated with a status.

Return Value

Status string

ppdFindAttr

Name

`ppdFindAttr` — Find the first matching attribute...

Synopsis

```
#include <cups/cups.h>
ppd_attr_t * ppdFindAttr(ppd_file_t * ppd, const char * name, const
char * spec);
```

Description

Find the first matching attribute...

Return Value

Attribute or NULL if not found

ppdFindChoice

Name

`ppdFindChoice` — Return a pointer to an option choice.

Synopsis

```
#include <cups/cups.h>
ppd_choice_t * ppdFindChoice(ppd_option_t * o, const char *
choice);
```

Description

Return a pointer to an option choice.

Return Value

Choice pointer or NULL

ppdFindMarkedChoice

Name

`ppdFindMarkedChoice` — Return the marked choice for the specified option.

Synopsis

```
#include <cups/cups.h>
ppd_choice_t * ppdFindMarkedChoice(ppd_file_t * ppd, const char *
option);
```

Description

Return the marked choice for the specified option.

Return Value

Pointer to choice or NULL

ppdFindNextAttr

Name

`ppdFindNextAttr` — Find the next matching attribute...

Synopsis

```
#include <cups/cups.h>
ppd_attr_t * ppdFindNextAttr(ppd_file_t * ppd, const char * name,
const char * spec);
```

Description

Find the next matching attribute...

Return Value

Attribute or NULL if not found

ppdFindOption

Name

`ppdFindOption` — Return a pointer to the specified option.

Synopsis

```
#include <cups/cups.h>
ppd_option_t * ppdFindOption(ppd_file_t * ppd, const char * option);
```

Description

Return a pointer to the specified option.

Return Value

Pointer to option or NULL

ppdIsMarked

Name

`ppdIsMarked` — Check to see if an option is marked...

Synopsis

```
#include <cups/cups.h>
int ppdIsMarked(ppd_file_t * ppd, const char * option, const char *
choice);
```

Description

Check to see if an option is marked...

Return Value

Non-zero if option is marked

ppdLastError

Name

`ppdLastError` — Return the status from the last ppdOpen*().

Synopsis

```
#include <cups/cups.h>
ppd_status_t ppdLastError(int * line);
```

Description

Return the status from the last ppdOpen*().

Return Value

Status code

ppdMarkDefaults

Name

`ppdMarkDefaults` — Mark all default options in the PPD file.

Synopsis

```
#include <cups/cups.h>
void ppdMarkDefaults(ppd_file_t * ppd);
```

Description

Mark all default options in the PPD file.

Return Value

This function does not return a value.

ppdMarkOption

Name

`ppdMarkOption` — Mark an option in a PPD file.

Synopsis

```
#include <cups/cups.h>
int ppdMarkOption(ppd_file_t * ppd, const char * option, const char
* choice);
```

Description

Mark an option in a PPD file.

Notes:

-1 is returned if the given option would conflict with any currently selected option.

Return Value

Number of conflicts

ppdOpen

Name

`ppdOpen` — Read a PPD file into memory.

Synopsis

```
#include <cups/cups.h>
ppd_file_t * ppdOpen(FILE * fp);
```

Description

Read a PPD file into memory.

Return Value

PPD file record

ppdOpenFd

Name

`ppdOpenFd` — Read a PPD file into memory.

Synopsis

```
#include <cups/cups.h>
ppd_file_t * ppdOpenFd(int fd);
```

Description

Read a PPD file into memory.

Return Value

PPD file record

ppdOpenFile

Name

`ppdOpenFile` — Read a PPD file into memory.

Synopsis

```
#include <cups/cups.h>
ppd_file_t * ppdOpenFile(const char * filename);
```

Description

Read a PPD file into memory.

Return Value

PPD file record

ppdPageLength

Name

`ppdPageLength` — Get the page length for the given size.

Synopsis

```
#include <cups/cups.h>
float ppdPageLength(ppd_file_t * ppd, const char * name);
```

Description

Get the page length for the given size.

Return Value

Length of page in points or 0.0

ppdPageSize

Name

`ppdPageSize` — Get the page size record for the given size.

Synopsis

```
#include <cups/cups.h>
ppd_size_t * ppdPageSize(ppd_file_t * ppd, const char * name);
```

Description

Get the page size record for the given size.

Return Value

Size record for page or NULL

ppdPageWidth

Name

`ppdPageWidth` — Get the page width for the given size.

Synopsis

```
#include <cups/cups.h>
float ppdPageWidth(ppd_file_t * ppd, const char * name);
```

Description

Get the page width for the given size.

Return Value

Width of page in points or 0.0

ppdSetConformance

Name

`ppdSetConformance` — Set the conformance level for PPD files.

Synopsis

```
#include <cups/cups.h>
void ppdSetConformance(ppd_conform_t c);
```

Description

Set the conformance level for PPD files.

Return Value

This function does not return a value.

7.4 Interfaces for libcupsimage

Table 7-3 defines the library name and shared object name for the libcupsimage library

Table 7-3 libcupsimage Definition

Library:	libcupsimage
SONAME:	libcupsimage.so.2

The behavior of the interfaces in this library is specified by the following specifications:

[LSB] This Specification

7.4.1 CUPS Raster ABI

7.4.1.1 Interfaces for CUPS Raster ABI

An LSB conforming implementation shall provide the generic functions for CUPS Raster ABI specified in Table 7-4, with the full mandatory functionality as described in the referenced underlying specification.

Table 7-4 libcupsimage - CUPS Raster ABI Function Interfaces

cupsRasterClose [LSB]	cupsRasterOpen [LSB]	cupsRasterReadHeader [LSB]	cupsRasterReadPixels [LSB]
cupsRasterWriteHeader [LSB]	cupsRasterWritePixels [LSB]		

7.5 Data Definitions for libcupsimage

This section defines global identifiers and their values that are associated with interfaces contained in libcupsimage. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and application developers should use this ABI to supplement - not to replace - source interface definition specifications.

This specification uses the ISO C (1999) C Language as the reference programming language, and data definitions are specified in ISO C format. The C language is used here as a convenient notation. Using a C language description of these data objects does not preclude their use by other programming languages.

7.5.1 cups/raster.h

```
#define _CUPS_RASTER_H_
#define CUPS_RASTER_SYNC        0x52615374
#define CUPS_RASTER_REVSYNC     0x74536152
#define CUPS_RASTER_HAVE_COLORIMETRIC   1

typedef enum {
    CUPS_RASTER_READ = 0,
    CUPS_RASTER_WRITE = 1
} cups_mode_t;
typedef struct _cups_raster_s cups_raster_t;
typedef enum {
    CUPS_ADVANCE_NONE = 0,
    CUPS_ADVANCE_FILE = 1,
    CUPS_ADVANCE_JOB = 2,
    CUPS_ADVANCE_SET = 3,
    CUPS_ADVANCE_PAGE = 4
} cups_adv_t;
typedef enum {
    CUPS_FALSE = 0,
    CUPS_TRUE = 1
} cups_bool_t;
typedef enum {
    CUPS_CUT_NONE = 0,
    CUPS_CUT_FILE = 1,
    CUPS_CUT_JOB = 2,
    CUPS_CUT_SET = 3,
    CUPS_CUT_PAGE = 4
} cups_cut_t;
typedef enum {
    CUPS_JOG_NONE = 0,
    CUPS_JOG_FILE = 1,
    CUPS_JOG_JOB = 2,
    CUPS_JOG_SET = 3
} cups_jog_t;
typedef enum {
    CUPS_EDGE_TOP = 0,
    CUPS_EDGE_RIGHT = 1,
    CUPS_EDGE_BOTTOM = 2,
    CUPS_EDGE_LEFT = 3
} cups_edge_t;
typedef enum {
    CUPS_ORIENT_0 = 0,
    CUPS_ORIENT_90 = 1,
    CUPS_ORIENT_180 = 2,
    CUPS_ORIENT_270 = 3
} cups_orient_t;
typedef enum {
    CUPS_ORDER_CHUNKED = 0,
    CUPS_ORDER_BANDED = 1,
    CUPS_ORDER_PLANAR = 2
```

```
} cups_order_t;
typedef enum {
    CUPS_CSPACE_W = 0,
    CUPS_CSPACE_RGB = 1,
    CUPS_CSPACE_RGBA = 2,
    CUPS_CSPACE_K = 3,
    CUPS_CSPACE_CMY = 4,
    CUPS_CSPACE_YMC = 5,
    CUPS_CSPACE_CMYK = 6,
    CUPS_CSPACE_YMCK = 7,
    CUPS_CSPACE_KCMY = 8,
    CUPS_CSPACE_KCMYcm = 9,
    CUPS_CSPACE_GMCK = 10,
    CUPS_CSPACE_GMCS = 11,
    CUPS_CSPACE_WHITE = 12,
    CUPS_CSPACE_GOLD = 13,
    CUPS_CSPACE_SILVER = 14,
    CUPS_CSPACE_CIEXYZ = 15,
    CUPS_CSPACE_CIELab = 16,
    CUPS_CSPACE_ICC1 = 32,
    CUPS_CSPACE_ICC2 = 33,
    CUPS_CSPACE_ICC3 = 34,
    CUPS_CSPACE_ICC4 = 35,
    CUPS_CSPACE_ICC5 = 36,
    CUPS_CSPACE_ICC6 = 37,
    CUPS_CSPACE_ICC7 = 38,
    CUPS_CSPACE_ICC8 = 39,
    CUPS_CSPACE_ICC9 = 40,
    CUPS_CSPACE_ICCA = 41,
    CUPS_CSPACE_ICCB = 42,
    CUPS_CSPACE_ICCC = 43,
    CUPS_CSPACE_ICCD = 44,
    CUPS_CSPACE_ICCE = 45,
    CUPS_CSPACE_ICCF = 46
} cups_cspace_t;
typedef struct {
    char MediaClass[64];
    char MediaColor[64];
    char MediaType[64];
    char OutputType[64];
    unsigned int AdvanceDistance;
    cups_adv_t AdvanceMedia;
    cups_bool_t Collate;
    cups_cut_t CutMedia;
    cups_bool_t Duplex;
    unsigned int HWResolution[2];
    unsigned int ImagingBoundingBox[4];
    cups_bool_t InsertSheet;
    cups_jog_t Jog;
    cups_edge_t LeadingEdge;
    unsigned int Margins[2];
    cups_bool_t ManualFeed;
    unsigned int MediaPosition;
    unsigned int MediaWeight;
    cups_bool_t MirrorPrint;
    cups_bool_t NegativePrint;
    unsigned int NumCopies;
    cups_orient_t Orientation;
    cups_bool_t OutputFaceUp;
    unsigned int PageSize[2];
    cups_bool_t Separations;
    cups_bool_t TraySwitch;
    cups_bool_t Tumble;
    unsigned int cupsWidth;
    unsigned int cupsHeight;
    unsigned int cupsMediaType;
    unsigned int cupsBitsPerColor;
    unsigned int cupsBitsPerPixel;
    unsigned int cupsBytesPerLine;
    cups_order_t cupsColorOrder;
    cups_cspace_t cupsColorSpace;
    unsigned int cupsCompression;
    unsigned int cupsRowCount;
    unsigned int cupsRowFeed;
    unsigned int cupsRowStep;
} cups_page_header_t;
extern void cupsRasterClose(cups_raster_t * r);
extern cups_raster_t *cupsRasterOpen(int fd, cups_mode_t mode);
extern unsigned int cupsRasterReadHeader(cups_raster_t * r,
                                         cups_page_header_t * h);
extern unsigned int cupsRasterReadPixels(cups_raster_t * r,
                                         unsigned char *p,
                                         unsigned int len);
extern unsigned int cupsRasterWriteHeader(cups_raster_t * r,
                                          cups_page_header_t     *
h);
extern unsigned int cupsRasterWritePixels(cups_raster_t * r,
                                          unsigned char *p,
                                          unsigned int len);
```

7.6 Interface Definitions for libcupsimage

The interfaces defined on the following pages are included in libcupsimage and are defined by this specification. Unless otherwise noted, these interfaces shall be included in the source standard.

Other interfaces listed in Section 7.4 shall behave as described in the referenced base document.

cupsRasterClose

Name

`cupsRasterClose` — Close a raster stream.

Synopsis

```
#include <cups/raster.h>
void cupsRasterClose(cups_raster_t * r);
```

Description

Close a raster stream.

Return Value

This function does not return a value.

cupsRasterOpen

Name

`cupsRasterOpen` — Open a raster stream.

Synopsis

```
#include <cups/raster.h>
cups_raster_t * cupsRasterOpen(int fd, cups_mode_t mode);
```

Description

Open a raster stream.

Return Value

New stream

cupsRasterReadHeader

Name

`cupsRasterReadHeader` — Read a raster page header and store it in a

Synopsis

```
#include <cups/raster.h>
unsigned cupsRasterReadHeader(cups_raster_t * r, cups_page_header_t
* h);
```

Description

Read a raster page header and store it in a V1 page header structure.

Return Value

1 on success, 0 on fail

cupsRasterReadPixels

Name

`cupsRasterReadPixels` — Read raster pixels.

Synopsis

```
#include <cups/raster.h>
unsigned cupsRasterReadPixels(cups_raster_t * r, unsigned char * p,
unsigned len);
```

Description

Read raster pixels.

Return Value

Number of bytes read

cupsRasterWriteHeader

Name

`cupsRasterWriteHeader` — Write a raster page header from a V1 page

Synopsis

```
#include <cups/raster.h>
unsigned cupsRasterWriteHeader(cups_raster_t * r, cups_page_header_t
* h);
```

Description

Write a raster page header from a V1 page header structure.

Return Value

1 on success, 0 on failure

cupsRasterWritePixels

Name

`cupsRasterWritePixels` — Write raster pixels.

Synopsis

```
#include <cups/raster.h>
unsigned cupsRasterWritePixels(cups_raster_t * r, unsigned char * p,
unsigned len);
```

Description

Write raster pixels.

Return Value

Number of bytes written

III Printing Commands

8 Printing Commands

8.1 Printing Commands

An LSB conforming implementation shall provide the commands and utilities as described in Table 8-1, with at least the behavior described as mandatory in the referenced underlying specification, with the following exceptions:

1. If any operand (except one which follows `--`) starts with a hyphen, the behavior is unspecified.

Rationale (Informative): Applications should place options before operands, or use `--`, as needed. This text is needed because, by default, GNU option parsing differs from POSIX, unless the environment variable POSIXLY_CORRECT is set. For example, **ls . -a** in GNU **ls** means to list the current directory, showing all files (that is, `"."` is an operand and `-a` is an option). In POSIX, `"."` and `-a` are both operands, and the command means to list the current directory, and also the file named `-a`. Suggesting that applications rely on the setting of the POSIXLY_CORRECT environment variable, or try to set it, seems worse than just asking the applications to invoke commands in ways which work with either the POSIX or GNU behaviors.

Table 8-1 Commands And Utilities

foomatic-rip [1]	gs [1]			

Referenced Specification(s)

[1]. This Specification

8.2 Command Behavior

This section contains descriptions for commands and utilities whose specified behavior in the LSB contradicts or extends the standards referenced. It also contains commands and utilities only required by the LSB and not specified by other standards.

FOOMATIC-RIP

2002-11-26

Name

`foomatic-rip` — Universal print filter/RIP wrapper

SYNOPSIS

Standalone Mode

foomatic-rip [-v] [-q] [-d] [--ppd ppdfile] [-J jobtitle] [-o option=value [...]] [files]

CUPS Mode

foomatic-rip jobid user jobtitle numcopies options [file]

DESCRIPTION

foomatic-rip is a universal print filter which works with every known free software printer spooler.

This page describes the facilities of foomatic-rip when used as a CUPS filter and when used outside of a print system. While implementations of foomatic-rip may support other print systems, such use is not documented here.

When run as a CUPS filter, foomatic-rip reads the job from the specified file, or from standard input if no file is specified. It renders the file into a printer-specific format, and writes the result to standard output.

When run standalone, foomatic-rip will read the job from the specified files, or from standard input if no files are given. The files are rendered into a printer-specific format, which is then output according to the PPD option "FoomaticRIPPostPipe", documented in the LSB.

Printer capabilities are described to foomatic-rip via PPD files, as described (with extensions used by foomatic-rip) in the LSB. The method foomatic-rip uses to determine the proper PPD file for the printer in question is defined by the implementation of both the spooler and foomatic-rip.

CUPS OPTIONS

Unless otherwise noted, all parameters are required when running foomatic-rip as a CUPS filter.

`jobid`

The internal Job ID from CUPS.

`username`

The username of the user who submitted the job.

`jobtitle`

The job's title, as submitted by the user.

`numcopies`

The number of copies of the job requested.

`options`

A series of printer options, separated by spaces, each of which take the form `name` or `name=value`. The specific list of options supported is dependent on the printer and spooler, and is usually documented in the PPD file for the printer.

An option may be preceded by a page specification, describing the pages to which the option should apply. A page specification consists of one or more items, separated by commas, and separated from the option name by a colon. Valid items include the words "even" and "odd", a single page number, and a page range. Page ranges are described with a starting page, a dash ("-"), and an ending page. If omitted, the starting and ending pages are the first and last page, respectively, but only one of the ends of the range may be omitted.

`file`

The full path to the file containing the job. This parameter is optional; if it is not supplied, the job is read from standard input.

SPOOLER-LESS OPTIONS

`-v`

Verbose mode. Intended for debugging and testing purposes only.

`-q`

Quiet mode - minimal information output.

`-d`

Identical to the 'opts' option, but option information is left in text format. The PPD file will need to be specified using the --ppd option.

`--ppd` *`ppdfile`*

The PPD file `ppdfile` should be applied for processing this job.

`-J` *`jobtitle`*

Print the given job title in the header of every page of a plain text job.

`-o` *`option=value`*

Set an option setting for this job.

EXIT STATUS

foomatic-rip returns 0 unless something unexpected happens.

AUTHOR

Till Kamppeter <*till.kamppeter@gmail.com*> with parts of Manfred Wassmanns's <*manolo@NCC-1701.B.Shuttle.de*> man pages for the Foomatic 2.0.x filters.

Jeff Licquia <*licquia@linux-foundation.org*> adapted the original man page for the LSB.

GS

2007-11-29

Name

`gs` — GhostScript (PostScript and PDF language interpreter)

SYNOPSIS

gs -h | --help

gs [options] ps-file [[options] ps-file2] ...

DESCRIPTION

The gs command invokes Ghostscript, an interpreter of Adobe Systems' PostScript(tm) and Portable Document Format (PDF) languages. gs reads the files named by ps-file in sequence and executes them as Ghostscript programs. After doing this, it reads further input from the standard input stream (normally the keyboard), interpreting each line separately. The interpreter exits gracefully when it encounters the "quit" command (either in a file or from the keyboard), at end-of-file, or at an interrupt signal (such as Control-C at the keyboard).

Some of GhostScript's options are set via command-line options; others are set as processing parameters, each of which consists of a name and a value.

OPTIONS

`-h --help`

Show GhostScript's help, as well as lists of the supported input formats, supported devices, and the search path for gs components.

`-q`

Suppress normal startup messages, and also set the processing parameter QUIET.

`-c`

Begin interpreting arguments as PostScript code. All following arguments are sent to the interpreter up to the next argument beginning with "-" followed by a non-digit, or with "@". This code is interpolated with the file list, so files specified before *`-c`* are interpreted beforehand, and files after *`-c`* are interpreted afterwards.

`-f`

Specifies a PostScript file to run as its argument. This is equivalent to the ps-file arguments, but is useful for terminating PostScript code as passed via *`-c`*.

`-d -D`

Set a processing parameter. The "name=value" pair follows immediately after the option, as in "-Dfoo=bar". The values here must be integers or the values "true" or "false". The equals sign and value may be omitted; this is assumed to set the name to "true".

`-s -S`

Set a processing parameter to a string value. The "name=value" pair follows immediately after the option, as in "-Sfoo=bar".

`-u`

Unset a processing parameter. The name to be unset follows immediately after the option, as in "-ufoo".

`-o`

Write rendered output to the named file, and also inhibit pauses and the interactive shell. This is equivalent to setting the processing parameters BATCH and NOPAUSE to true, and OutputFile to the parameter after -o.

`-r`

Set the device resolution. The resolution is specified as two numbers separated with an "x", as in "300x150", corresponding to the X and Y axis resolutions, respectively. If a single number is given without an "x", it is treated as the value for both resolutions.

This is equivalent to setting DEVICEXRESOLUTION and DEVICEYRESOLUTION in systemdict.

`-g`

Set the device size, in pixels. The size is specified as two numbers separated with an "x", as in "640x480", corresponding to the width and height, respectively.

This is equivalent to setting DEVICEWIDTH and DEVICEHEIGHT in systemdict.

RECOGNIZED PROCESSING PARAMETERS

Processing parameters may have arbitrary names; no limits are placed on the settings that may be made. However, certain settings have meaning to the gs interpreter, and drivers may use other settings. Below is a list of recognized settings that the gs interpreter must respect.

`BATCH`

If set to true, do not enter an interactive shell after processing all command-line files.

`DEVICE`

Contains the name of the device used to render the page, as a string.

The list of available devices can be discovered with the -h parameter, as described above. At least the following devices must be present: cups (CUPS Raster), ijs, pxlmono, pxlcolor, and opvp (OpenPrinting Vector).

`DEVICEHEIGHT`

Contains the height, in pixels, of the output device.

The effect of this setting when the current driver is a vector driver is undefined.

`DEVICEHEIGHTPOINTS`

Sets the initial page height, in units of 1/72 of an inch.

`DEVICEWIDTH`

Contains the width, in pixels, of the output device.

The effect of this setting when the current driver is a vector driver is undefined.

DEVICEWIDTHPOINTS

Sets the initial page width, in units of 1/72 of an inch.

DEVICEXRESOLUTION

Contains the resolution, in pixels per inch, of the X dimension (horizontal) of the output device.

DEVICEYRESOLUTION

Contains the resolution, in pixels per inch, of the Y dimension (vertical) of the output device.

NOPAUSE

If set to true, disable the prompt and pause normally displayed after rendering a page.

OutputFile

Contains the path to the file to which gs should write its output, as a string. This parameter may be set to '-', in which case gs's output is sent to standard output.

PAPERSIZE

Contains the string representation of the paper size. The ISO paper sizes a0-a10 (plus a4small), isob0-isob6, and c0-c6 are recognized, as are jisb0-jisb6 (JIS standard sizes) and the US paper sizes 11x17, ledger, legal, letter, lettersmall, and archA-archE.

QUIET

If set to true, suppress routine information comments on standard output.

SAFER

If set to true, disable several unsafe PostScript features: the deletefile and renamefile operators, piped commands, reading or writing to general files, and changing of certain system settings.

STRICT

If set to true, disable as many extensions to the Adobe PostScript specification as possible.

EXIT STATUS

gs returns 0 on successful execution. Any other return value indicates an error.

AUTHOR

Jeff Licquia (licquia@linux-foundation.org) wrote this man page for the LSB specification.

Portions of this page were taken from the GhostScript documentation. The maintainer and rights holder for GhostScript is Artifex Software, Inc.

IV Execution Environment

9 File System Hierarchy

In addition to the requirements for `/usr/share` in the Filesystem Hierarchy Standard, an LSB conforming system shall also provide the following directories or symbolic links to directories:

`/usr/share/ppd`

ppd files