CUPS Programming Manual

Michael R Sweet

Contents

Introduction
- Guidelines
- Terms Used in This Document
- Compiling Programs That Use the CUPS API

Working with Destinations
- Finding Available Destinations
- Basic Destination Information
- Detailed Destination Information
- Submitting a Print Job

Sending IPP Requests
- Connecting to the Scheduler or Printer
- Creating an IPP Request
- Sending the IPP Request
- Processing the IPP Response
- Authentication

Functions
- cupsAddDest
- cupsAddDestMediaOptions
- cupsAddIntegerOption
- cupsAddOption
- cupsCancelDestJob
- cupsCheckDestSupported
- cupsCloseDestJob
- cupsConnectDest
- cupsCopyDest
- cupsCopyDestConflicts
- cupsCopyDestInfo
- cupsCreateDestJob
- cupsDoAuthentication
- cupsEncodeOption
- cupsEncodeOptions
- cupsEncodeOptions2
- cupsEncryption
- cupsEnumDests
- cupsFindDestDefault
- cupsFindDestReady
- cupsFindDestSupported
- cupsFinishDestDocument
- cupsFreeDestInfo
- cupsFreeDests
- cupsFreeJobs
- cupsFreeOptions

cupsGetDest
cupsGetDestMediaByIndex
cupsGetDestMediaByName
cupsGetDestMediaBySize
cupsGetDestMediaCount
cupsGetDestMediaDefault
cupsGetDestWithURI
cupsGetDests2
cupsGetIntegerOption
cupsGetJobs2
cupsGetNamedDest
cupsGetOption
cupsGetPassword2
cupsLocalizeDestMedia
cupsLocalizeDestOption
cupsLocalizeDestValue
cupsMakeServerCredentials
cupsParseOptions
cupsRemoveDest
cupsRemoveOption
cupsServer
cupsSetClientCertCB
cupsSetCredentials
cupsSetDefaultDest
cupsSetDests2
cupsSetEncryption
cupsSetPasswordCB2
cupsSetServer
cupsSetServerCertCB
cupsSetServerCredentials
cupsSetUser
cupsSetUserAgent
cupsStartDestDocument
cupsUser
cupsUserAgent
httpAcceptConnection
httpAddCredential
httpAddrAny
httpAddrClose
httpAddrConnect2
httpAddrCopyList
httpAddrEqual
httpAddrFamily
httpAddrFreeList
httpAddrGetList
httpAddrLength
httpAddrListen
httpAddrLocalhost
httpAddrLookup
httpAddrPort
httpAddrString
httpAssembleURI
httpAssembleURIf
httpAssembleUUID
httpBlocking
httpCheck
httpClearCookie

httpClearFields
httpClose
httpCompareCredentials
httpConnect2
httpCopyCredentials
httpCredentialsAreValidForName
httpCredentialsGetExpiration
httpCredentialsGetTrust
httpCredentialsString
httpDecode64_2
httpDelete
httpEncode64_2
httpEncryption
httpError
httpFieldValue
httpFlush
httpFlushWrite
httpFreeCredentials
httpGet
httpGetActivity
httpGetAddress
httpGetAuthString
httpGetBlocking
httpGetContentEncoding
httpGetCookie
httpGetDateString2
httpGetDateTime
httpGetEncryption
httpGetExpect
httpGetFd
httpGetField
httpGetHostname
httpGetKeepAlive
httpGetLength2
httpGetPending
httpGetReady
httpGetRemaining
httpGetState
httpGetStatus
httpGetSubField2
httpGetVersion
httpGets
httpHead
httpInitialize
httpIsChunked
httpIsEncrypted
httpLoadCredentials
httpOptions
httpPeek
httpPost
httpPut
httpRead2
httpReadRequest
httpReconnect2
httpResolveHostname
httpSaveCredentials
httpSeparateURI

httpSetAuthString
httpSetCookie
httpSetCredentials
httpSetDefaultField
httpSetExpect
httpSetField
httpSetKeepAlive
httpSetLength
httpSetTimeout
httpShutdown
httpStateString
httpStatus
httpURIStatusString
httpUpdate
httpWait
httpWrite2
httpWriteResponse
ippAddBoolean
ippAddBooleans
ippAddCollection
ippAddCollections
ippAddDate
ippAddInteger
ippAddIntegers
ippAddOctetString
ippAddOutOfBand
ippAddRange
ippAddRanges
ippAddResolution
ippAddResolutions
ippAddSeparator
ippAddString
ippAddStringf
ippAddStringfv
ippAddStrings
ippAttributeString
ippContainsInteger
ippContainsString
ippCopyAttribute
ippCopyAttributes
ippCreateRequestedArray
ippDateToTime
ippDelete
ippDeleteAttribute
ippDeleteValues
ippEnumString
ippEnumValue
ippErrorString
ippErrorValue
ippFindAttribute
ippFindNextAttribute
ippFirstAttribute
ippGetBoolean
ippGetCollection
ippGetCount
ippGetDate
ippGetGroupTag

ippGetInteger
ippGetName
ippGetOctetString
ippGetOperation
ippGetRange
ippGetRequestId
ippGetResolution
ippGetState
ippGetStatusCode
ippGetString
ippGetValueTag
ippGetVersion
ippLength
ippNew
ippNewRequest
ippNewResponse
ippNextAttribute
ippOpString
ippOpValue
ippPort
ippRead
ippReadFile
ippReadIO
ippSetBoolean
ippSetCollection
ippSetDate
ippSetGroupTag
ippSetInteger
ippSetName
ippSetOctetString
ippSetOperation
ippSetPort
ippSetRange
ippSetRequestId
ippSetResolution
ippSetState
ippSetStatusCode
ippSetString
ippSetStringf
ippSetStringfv
ippSetValueTag
ippSetVersion
ippStateString
ippTagString
ippTagValue
ippTimeToDate
ippValidateAttribute
ippValidateAttributes
ippWrite
ippWriteFile
ippWriteIO

Data Types
cups_client_cert_cb_t
cups_dest_cb_t
cups_dest_t
cups_dinfo_t

- cups_job_t
- cups_option_t
- cups_password_cb2_t
- cups_ptype_t
- cups_server_cert_cb_t
- cups_size_t
- http_addr_t
- http_encoding_t
- http_encryption_t
- http_field_t
- http_keepalive_t
- http_state_t
- http_t
- http_timeout_cb_t
- http_trust_t
- http_uri_coding_t
- http_uri_status_t
- ipp_attribute_t
- ipp_copycb_t
- ipp_iocb_t
- ipp_orient_t
- ipp_pstate_t
- ipp_quality_t
- ipp_res_t
- ipp_rstate_t
- ipp_sstate_t
- ipp_state_t
- ipp_t

Structures

- cups_dest_s
- cups_job_s
- cups_option_s
- cups_size_s

Enumerations

- cups_ptype_e
- http_encoding_e
- http_encryption_e
- http_field_e
- http_keepalive_e
- http_state_e
- http_status_e
- http_trust_e
- http_uri_coding_e
- http_uri_status_e
- ipp_finishings_e
- ipp_jstate_e
- ipp_op_e
- ipp_orient_e
- ipp_pstate_e
- ipp_quality_e
- ipp_res_e
- ipp_rstate_e
- ipp_sstate_e
- ipp_state_e
- ipp_status_e

ipp_tag_e

Please file issues on Github to provide feedback on this document.

Introduction

CUPS provides the "cups" library to talk to the different parts of CUPS and with Internet Printing Protocol (IPP) printers. The "cups" library functions are accessed by including the `<cups/cups.h>` header.

CUPS is based on the Internet Printing Protocol ("IPP"), which allows clients (applications) to communicate with a server (the scheduler, printers, etc.) to get a list of destinations, send print jobs, and so forth. You identify which server you want to communicate with using a pointer to the opaque structure `http_t`. The `CUPS_HTTP_DEFAULT` constant can be used when you want to talk to the CUPS scheduler.

Guidelines

When writing software (other than printer drivers) that uses the "cups" library:

- Do not use undocumented or deprecated APIs,
- Do not rely on pre-configured printers,
- Do not assume that printers support specific features or formats, and
- Do not rely on implementation details (PPDs, etc.)

CUPS is designed to insulate users and developers from the implementation details of printers and file formats. The goal is to allow an application to supply a print file in a standard format with the user intent ("print four copies, two-sided on A4 media, and staple each copy") and have the printing system manage the printer communication and format conversion needed.

Similarly, printer and job management applications can use standard query operations to obtain the status information in a common, generic form and use standard management operations to control the state of those printers and jobs.

> **Note:**
>
> CUPS printer drivers necessarily depend on specific file formats and certain implementation details of the CUPS software. Please consult the Postscript and raster printer driver developer documentation on CUPS.org for more information.

Terms Used in This Document

A *Destination* is a printer or print queue that accepts print jobs. A *Print Job* is a collection of one or more documents that are processed by a destination using options supplied when creating the job. A *Document* is a file (JPEG image, PDF file, etc.) suitable for printing. An *Option* controls some aspect of printing, such as the media used. *Media* is the sheets or roll that is printed on. An *Attribute* is an option encoded for an Internet Printing Protocol (IPP) request.

Compiling Programs That Use the CUPS API

The CUPS libraries can be used from any C, C++, or Objective C program. The method of compiling against the libraries varies depending on the operating system and installation of CUPS. The following sections show how to compile a simple program (shown below) in two common environments.

The following simple program lists the available destinations:

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

int print_dest(void *user_data, unsigned flags, cups_dest_t *dest)
{
  if (dest->instance)
    printf("%s/%s\n", dest->name, dest->instance);
  else
    puts(dest->name);

  return (1);
}

int main(void)
{
  cupsEnumDests(CUPS_DEST_FLAGS_NONE, 1000, NULL, 0, 0, print_dest, NULL);

  return (0);
}
```

Compiling with Xcode

In Xcode, choose *New Project...* from the *File* menu (or press SHIFT+CMD+N), then select the *Command Line Tool* under the macOS Application project type. Click *Next* and enter a name

for the project, for example "firstcups". Click *Next* and choose a project directory. The click *Next* to create the project.

In the project window, click on the *Build Phases* group and expand the *Link Binary with Libraries* section. Click +, type "libcups" to show the library, and then double-click on `libcups.tbd`.

Finally, click on the `main.c` file in the sidebar and copy the example program to the file. Build and run (CMD+R) to see the list of destinations.

Compiling with GCC

From the command-line, create a file called `simple.c` using your favorite editor, copy the example to this file, and save. Then run the following command to compile it with GCC and run it:

```
gcc -o simple `cups-config --cflags` simple.c `cups-config --libs`
./simple
```

The `cups-config` command provides the compiler flags (`cups-config --cflags`) and libraries (`cups-config --libs`) needed for the local system.

Working with Destinations

Destinations, which in CUPS represent individual printers or classes (collections or pools) of printers, are represented by the `cups_dest_t` structure which includes the name (`name`), instance (`instance`, saved options/settings), whether the destination is the default for the user (`is_default`), and the options and basic information associated with that destination (`num_options` and `options`).

Historically destinations have been manually maintained by the administrator of a system or network, but CUPS also supports dynamic discovery of destinations on the current network.

Finding Available Destinations

The `cupsEnumDests` function finds all of the available destinations:

```
int
cupsEnumDests(unsigned flags, int msec, int *cancel,
              cups_ptype_t type, cups_ptype_t mask,
              cups_dest_cb_t cb, void *user_data)
```

The `flags` argument specifies enumeration options, which at present must be

`CUPS_DEST_FLAGS_NONE`.

The `msec` argument specifies the maximum amount of time that should be used for enumeration in milliseconds - interactive applications should keep this value to 5000 or less when run on the main thread.

The `cancel` argument points to an integer variable that, when set to a non-zero value, will cause enumeration to stop as soon as possible. It can be `NULL` if not needed.

The `type` and `mask` arguments are bitfields that allow the caller to filter the destinations based on categories and/or capabilities. The destination's "printer-type" value is masked by the `mask` value and compared to the `type` value when filtering. For example, to only enumerate destinations that are hosted on the local system, pass `CUPS_PRINTER_LOCAL` for the `type` argument and `CUPS_PRINTER_DISCOVERED` for the `mask` argument. The following constants can be used for filtering:

- `CUPS_PRINTER_CLASS`: A collection of destinations.
- `CUPS_PRINTER_FAX`: A facsimile device.
- `CUPS_PRINTER_LOCAL`: A local printer or class. This constant has the value 0 (no bits set) and is only used for the `type` argument and is paired with the `CUPS_PRINTER_REMOTE` or `CUPS_PRINTER_DISCOVERED` constant passed in the `mask` argument.
- `CUPS_PRINTER_REMOTE`: A remote (shared) printer or class.
- `CUPS_PRINTER_DISCOVERED`: An available network printer or class.
- `CUPS_PRINTER_BW`: Can do B&W printing.
- `CUPS_PRINTER_COLOR`: Can do color printing.
- `CUPS_PRINTER_DUPLEX`: Can do two-sided printing.
- `CUPS_PRINTER_STAPLE`: Can staple output.
- `CUPS_PRINTER_COLLATE`: Can quickly collate copies.
- `CUPS_PRINTER_PUNCH`: Can punch output.
- `CUPS_PRINTER_COVER`: Can cover output.
- `CUPS_PRINTER_BIND`: Can bind output.
- `CUPS_PRINTER_SORT`: Can sort output (mailboxes, etc.)
- `CUPS_PRINTER_SMALL`: Can print on Letter/Legal/A4-size media.
- `CUPS_PRINTER_MEDIUM`: Can print on Tabloid/B/C/A3/A2-size media.
- `CUPS_PRINTER_LARGE`: Can print on D/E/A1/A0-size media.
- `CUPS_PRINTER_VARIABLE`: Can print on rolls and custom-size media.

The `cb` argument specifies a function to call for every destination that is found:

```
typedef int (*cups_dest_cb_t)(void *user_data,
                              unsigned flags,
                              cups_dest_t *dest);
```

The callback function receives a copy of the `user_data` argument along with a bitfield (`flags`) and the destination that was found. The `flags` argument can have any of the following constant (bit) values set:

- `CUPS_DEST_FLAGS_MORE`: There are more destinations coming.

- `CUPS_DEST_FLAGS_REMOVED`: The destination has gone away and should be removed from the list of destinations a user can select.
- `CUPS_DEST_FLAGS_ERROR`: An error occurred. The reason for the error can be found by calling the `cupsLastError` and/or `cupsLastErrorString` functions.

The callback function returns 0 to stop enumeration or 1 to continue.

> **Note:**
>
> The callback function will likely be called multiple times for the same destination, so it is up to the caller to suppress any duplicate destinations.

The following example shows how to use `cupsEnumDests` to get a filtered array of destinations:

```
typedef struct
{
  int num_dests;
  cups_dest_t *dests;
} my_user_data_t;

int
my_dest_cb(my_user_data_t *user_data, unsigned flags,
           cups_dest_t *dest)
{
  if (flags & CUPS_DEST_FLAGS_REMOVED)
  {
   /*
    * Remove destination from array...
    */

    user_data->num_dests =
        cupsRemoveDest(dest->name, dest->instance,
                       user_data->num_dests,
                       &(user_data->dests));
  }
  else
  {
   /*
    * Add destination to array...
    */

    user_data->num_dests =
        cupsCopyDest(dest, user_data->num_dests,
                     &(user_data->dests));
  }

  return (1);
}

int
my_get_dests(cups_ptype_t type, cups_ptype_t mask,
             cups_dest_t **dests)
{
  my_user_data_t user_data = { 0, NULL };

  if (!cupsEnumDests(CUPS_DEST_FLAGS_NONE, 1000, NULL, type,
                     mask, (cups_dest_cb_t)my_dest_cb,
                     &user_data))
  {
   /*
    * An error occurred, free all of the destinations and
    * return...
    */

    cupsFreeDests(user_data.num_dests, user_dasta.dests);

    *dests = NULL;

    return (0);
  }

 /*
  * Return the destination array...
  */

  *dests = user_data.dests;

  return (user_data.num_dests);
}
```

Basic Destination Information

The `num_options` and `options` members of the `cups_dest_t` structure provide basic attributes about the destination in addition to the user default options and values for that destination. The following names are predefined for various destination attributes:

- "auth-info-required": The type of authentication required for printing to this destination: "none", "username,password", "domain,username,password", or "negotiate" (Kerberos).
- "printer-info": The human-readable description of the destination such as "My Laser Printer".
- "printer-is-accepting-jobs": "true" if the destination is accepting new jobs, "false" otherwise.
- "printer-is-shared": "true" if the destination is being shared with other computers, "false" otherwise.
- "printer-location": The human-readable location of the destination such as "Lab 4".
- "printer-make-and-model": The human-readable make and model of the destination such as "ExampleCorp LaserPrinter 4000 Series".
- "printer-state": "3" if the destination is idle, "4" if the destination is printing a job, and "5" if the destination is stopped.
- "printer-state-change-time": The UNIX time when the destination entered the current state.
- "printer-state-reasons": Additional comma-delimited state keywords for the destination such as "media-tray-empty-error" and "toner-low-warning".
- "printer-type": The `cups_ptype_t` value associated with the destination.
- "printer-uri-supported": The URI associated with the destination; if not set, this destination was discovered but is not yet setup as a local printer.

Use the `cupsGetOption` function to retrieve the value. For example, the following code gets the make and model of a destination:

```
const char *model = cupsGetOption("printer-make-and-model",
                                  dest->num_options,
                                  dest->options);
```

Detailed Destination Information

Once a destination has been chosen, the `cupsCopyDestInfo` function can be used to gather detailed information about the destination:

```
cups_dinfo_t *
cupsCopyDestInfo(http_t *http, cups_dest_t *dest);
```

The `http` argument specifies a connection to the CUPS scheduler and is typically the constant `CUPS_HTTP_DEFAULT`. The `dest` argument specifies the destination to query.

The `cups_dinfo_t` structure that is returned contains a snapshot of the supported options and their supported, ready, and default values. It also can report constraints between different options and values, and recommend changes to resolve those constraints.

Getting Supported Options and Values

The `cupsCheckDestSupported` function can be used to test whether a particular option or option and value is supported:

```
int
cupsCheckDestSupported(http_t *http, cups_dest_t *dest,
                       cups_dinfo_t *info,
                       const char *option,
                       const char *value);
```

The `option` argument specifies the name of the option to check. The following constants can be used to check the various standard options:

- `CUPS_COPIES`: Controls the number of copies that are produced.
- `CUPS_FINISHINGS`: A comma-delimited list of integer constants that control the finishing processes that are applied to the job, including stapling, punching, and folding.
- `CUPS_MEDIA`: Controls the media size that is used, typically one of the following: `CUPS_MEDIA_3X5`, `CUPS_MEDIA_4X6`, `CUPS_MEDIA_5X7`, `CUPS_MEDIA_8X10`, `CUPS_MEDIA_A3`, `CUPS_MEDIA_A4`, `CUPS_MEDIA_A5`, `CUPS_MEDIA_A6`, `CUPS_MEDIA_ENV10`, `CUPS_MEDIA_ENVDL`, `CUPS_MEDIA_LEGAL`, `CUPS_MEDIA_LETTER`, `CUPS_MEDIA_PHOTO_L`, `CUPS_MEDIA_SUPERBA3`, or `CUPS_MEDIA_TABLOID`.
- `CUPS_MEDIA_SOURCE`: Controls where the media is pulled from, typically either `CUPS_MEDIA_SOURCE_AUTO` or `CUPS_MEDIA_SOURCE_MANUAL`.
- `CUPS_MEDIA_TYPE`: Controls the type of media that is used, typically one of the following: `CUPS_MEDIA_TYPE_AUTO`, `CUPS_MEDIA_TYPE_ENVELOPE`, `CUPS_MEDIA_TYPE_LABELS`, `CUPS_MEDIA_TYPE_LETTERHEAD`, `CUPS_MEDIA_TYPE_PHOTO`, `CUPS_MEDIA_TYPE_PHOTO_GLOSSY`, `CUPS_MEDIA_TYPE_PHOTO_MATTE`, `CUPS_MEDIA_TYPE_PLAIN`, or `CUPS_MEDIA_TYPE_TRANSPARENCY`.
- `CUPS_NUMBER_UP`: Controls the number of document pages that are placed on each media side.
- `CUPS_ORIENTATION`: Controls the orientation of document pages placed on the media: `CUPS_ORIENTATION_PORTRAIT` or `CUPS_ORIENTATION_LANDSCAPE`.
- `CUPS_PRINT_COLOR_MODE`: Controls whether the output is in color (`CUPS_PRINT_COLOR_MODE_COLOR`), grayscale (`CUPS_PRINT_COLOR_MODE_MONOCHROME`), or either (`CUPS_PRINT_COLOR_MODE_AUTO`).
- `CUPS_PRINT_QUALITY`: Controls the generate quality of the output: `CUPS_PRINT_QUALITY_DRAFT`, `CUPS_PRINT_QUALITY_NORMAL`, or `CUPS_PRINT_QUALITY_HIGH`.
- `CUPS_SIDES`: Controls whether prints are placed on one or both sides of the media: `CUPS_SIDES_ONE_SIDED`, `CUPS_SIDES_TWO_SIDED_PORTRAIT`, or `CUPS_SIDES_TWO_SIDED_LANDSCAPE`.

If the `value` argument is `NULL`, the `cupsCheckDestSupported` function returns whether the option is supported by the destination. Otherwise, the function returns whether the specified value of the option is supported.

The `cupsFindDestSupported` function returns the IPP attribute containing the supported values for a given option:

```
ipp_attribute_t *
cupsFindDestSupported(http_t *http, cups_dest_t *dest,
                      cups_dinfo_t *dinfo,
                      const char *option);
```

For example, the following code prints the supported finishing processes for a destination, if any, to the standard output:

```
cups_dinfo_t *info = cupsCopyDestInfo(CUPS_HTTP_DEFAULT,
                                      dest);

if (cupsCheckDestSupported(CUPS_HTTP_DEFAULT, dest, info,
                           CUPS_FINISHINGS, NULL))
{
  ipp_attribute_t *finishings =
      cupsFindDestSupported(CUPS_HTTP_DEFAULT, dest, info,
                            CUPS_FINISHINGS);
  int i, count = ippGetCount(finishings);

  puts("finishings supported:");
  for (i = 0; i < count; i ++)
    printf("  %d\n", ippGetInteger(finishings, i));
}
else
  puts("finishings not supported.");
```

The "job-creation-attributes" option can be queried to get a list of supported options. For example, the following code prints the list of supported options to the standard output:

```
ipp_attribute_t *attrs =
    cupsFindDestSupported(CUPS_HTTP_DEFAULT, dest, info,
                          "job-creation-attributes");
int i, count = ippGetCount(attrs);

for (i = 0; i < count; i ++)
  puts(ippGetString(attrs, i, NULL));
```

Getting Default Values

There are two sets of default values - user defaults that are available via the `num_options` and `options` members of the `cups_dest_t` structure, and destination defaults that available via the `cups_dinfo_t` structure and the `cupsFindDestDefault` function which returns the IPP attribute containing the default value(s) for a given option:

```
ipp_attribute_t *
cupsFindDestDefault(http_t *http, cups_dest_t *dest,
                    cups_dinfo_t *dinfo,
                    const char *option);
```

The user defaults from `cupsGetOption` should always take preference over the destination defaults. For example, the following code prints the default finishings value(s) to the standard output:

```
const char *def_value =
    cupsGetOption(CUPS_FINISHINGS, dest->num_options,
                  dest->options);
ipp_attribute_t *def_attr =
    cupsFindDestDefault(CUPS_HTTP_DEFAULT, dest, info,
                        CUPS_FINISHINGS);

if (def_value != NULL)
{
  printf("Default finishings: %s\n", def_value);
}
else
{
  int i, count = ippGetCount(def_attr);

  printf("Default finishings: %d",
         ippGetInteger(def_attr, 0));
  for (i = 1; i < count; i ++)
    printf(",%d", ippGetInteger(def_attr, i));
  putchar('\n');
}
```

Getting Ready (Loaded) Values

The finishings and media options also support queries for the ready, or loaded, values. For example, a printer may have punch and staple finishers installed but be out of staples - the supported values will list both punch and staple finishing processes but the ready values will only list the punch processes. Similarly, a printer may support hundreds of different sizes of media but only have a single size loaded at any given time - the ready values are limited to the media that is actually in the printer.

The `cupsFindDestReady` function finds the IPP attribute containing the ready values for a given option:

```
ipp_attribute_t *
cupsFindDestReady(http_t *http, cups_dest_t *dest,
                  cups_dinfo_t *dinfo, const char *option);
```

For example, the following code lists the ready finishing processes:

```
ipp_attribute_t *ready_finishings =
    cupsFindDestReady(CUPS_HTTP_DEFAULT, dest, info,
                      CUPS_FINISHINGS);

if (ready_finishings != NULL)
{
  int i, count = ippGetCount(ready_finishings);

  puts("finishings ready:");
  for (i = 0; i < count; i ++)
    printf("  %d\n", ippGetInteger(ready_finishings, i));
}
else
  puts("no finishings are ready.");
```

Media Size Options

CUPS provides functions for querying the dimensions and margins for each of the supported media size options. The `cups_size_t` structure is used to describe a media size:

```
typedef struct cups_size_s
{
  char media[128];
  int width, length;
  int bottom, left, right, top;
} cups_size_t;
```

The `width` and `length` members specify the dimensions of the media in hundredths of millimeters (1/2540th of an inch). The `bottom`, `left`, `right`, and `top` members specify the margins of the printable area, also in hundredths of millimeters.

The `cupsGetDestMediaByName` and `cupsGetDestMediaBySize` functions lookup the media size information using a standard media size name or dimensions in hundredths of millimeters:

```
int
cupsGetDestMediaByName(http_t *http, cups_dest_t *dest,
                       cups_dinfo_t *dinfo,
                       const char *media,
                       unsigned flags, cups_size_t *size);

int
cupsGetDestMediaBySize(http_t *http, cups_dest_t *dest,
                       cups_dinfo_t *dinfo,
                       int width, int length,
                       unsigned flags, cups_size_t *size);
```

The `media`, `width`, and `length` arguments specify the size to lookup. The `flags` argument specifies a bitfield controlling various lookup options:

- `CUPS_MEDIA_FLAGS_DEFAULT`: Find the closest size supported by the printer.
- `CUPS_MEDIA_FLAGS_BORDERLESS`: Find a borderless size.
- `CUPS_MEDIA_FLAGS_DUPLEX`: Find a size compatible with two-sided printing.
- `CUPS_MEDIA_FLAGS_EXACT`: Find an exact match for the size.
- `CUPS_MEDIA_FLAGS_READY`: If the printer supports media sensing or configuration of the media in each tray/source, find the size amongst the "ready" media.

If a matching size is found for the destination, the size information is stored in the structure pointed to by the `size` argument and 1 is returned. Otherwise 0 is returned.

For example, the following code prints the margins for two-sided printing on US Letter media:

```
cups_size_t size;

if (cupsGetDestMediaByName(CUPS_HTTP_DEFAULT, dest, info,
                           CUPS_MEDIA_LETTER,
                           CUPS_MEDIA_FLAGS_DUPLEX, &size))
{
  puts("Margins for duplex US Letter:");
  printf("  Bottom: %.2fin\n", size.bottom / 2540.0);
  printf("    Left: %.2fin\n", size.left / 2540.0);
  printf("   Right: %.2fin\n", size.right / 2540.0);
  printf("     Top: %.2fin\n", size.top / 2540.0);
}
else
  puts("Margins for duplex US Letter are not available.");
```

You can also enumerate all of the sizes that match a given `flags` value using the `cupsGetDestMediaByIndex` and `cupsGetDestMediaCount` functions:

```
int
cupsGetDestMediaByIndex(http_t *http, cups_dest_t *dest,
                        cups_dinfo_t *dinfo, int n,
                        unsigned flags, cups_size_t *size);

int
cupsGetDestMediaCount(http_t *http, cups_dest_t *dest,
                      cups_dinfo_t *dinfo, unsigned flags);
```

For example, the following code prints the list of ready media and corresponding margins:

```
cups_size_t size;
int i;
int count = cupsGetDestMediaCount(CUPS_HTTP_DEFAULT,
                                  dest, info,
                                  CUPS_MEDIA_FLAGS_READY);

for (i = 0; i < count; i ++)
{
  if (cupsGetDestMediaByIndex(CUPS_HTTP_DEFAULT, dest, info,
                              i, CUPS_MEDIA_FLAGS_READY,
                              &size))
  {
    printf("%s:\n", size.name);
    printf("   Width: %.2fin\n", size.width / 2540.0);
    printf("  Length: %.2fin\n", size.length / 2540.0);
    printf("  Bottom: %.2fin\n", size.bottom / 2540.0);
    printf("    Left: %.2fin\n", size.left / 2540.0);
    printf("   Right: %.2fin\n", size.right / 2540.0);
    printf("     Top: %.2fin\n", size.top / 2540.0);
  }
}
```

Finally, the `cupsGetDestMediaDefault` function returns the default media size:

```
int
cupsGetDestMediaDefault(http_t *http, cups_dest_t *dest,
                        cups_dinfo_t *dinfo, unsigned flags,
                        cups_size_t *size);
```

Localizing Options and Values

CUPS provides three functions to get localized, human-readable strings in the user's current locale for options and values: `cupsLocalizeDestMedia`, `cupsLocalizeDestOption`, and `cupsLocalizeDestValue`:

```
const char *
cupsLocalizeDestMedia(http_t *http, cups_dest_t *dest,
                      cups_dinfo_t *info, unsigned flags,
                      cups_size_t *size);

const char *
cupsLocalizeDestOption(http_t *http, cups_dest_t *dest,
                       cups_dinfo_t *info,
                       const char *option);

const char *
cupsLocalizeDestValue(http_t *http, cups_dest_t *dest,
                      cups_dinfo_t *info,
                      const char *option, const char *value);
```

Submitting a Print Job

Once you are ready to submit a print job, you create a job using the `cupsCreateDestJob` function:

```
ipp_status_t
cupsCreateDestJob(http_t *http, cups_dest_t *dest,
                  cups_dinfo_t *info, int *job_id,
                  const char *title, int num_options,
                  cups_option_t *options);
```

The `title` argument specifies a name for the print job such as "My Document". The `num_options` and `options` arguments specify the options for the print job which are allocated using the `cupsAddOption` function.

When successful, the job's numeric identifier is stored in the integer pointed to by the `job_id` argument and `IPP_STATUS_OK` is returned. Otherwise, an IPP error status is returned.

For example, the following code creates a new job that will print 42 copies of a two-sided US Letter document:

```
int job_id = 0;
int num_options = 0;
cups_option_t *options = NULL;

num_options = cupsAddOption(CUPS_COPIES, "42",
                            num_options, &options);
num_options = cupsAddOption(CUPS_MEDIA, CUPS_MEDIA_LETTER,
                            num_options, &options);
num_options = cupsAddOption(CUPS_SIDES,
                            CUPS_SIDES_TWO_SIDED_PORTRAIT,
                            num_options, &options);

if (cupsCreateDestJob(CUPS_HTTP_DEFAULT, dest, info,
                      &job_id, "My Document", num_options,
                      options) == IPP_STATUS_OK)
  printf("Created job: %d\n", job_id);
else
  printf("Unable to create job: %s\n",
         cupsLastErrorString());
```

Once the job is created, you submit documents for the job using the `cupsStartDestDocument`, `cupsWriteRequestData`, and `cupsFinishDestDocument` functions:

```
http_status_t
cupsStartDestDocument(http_t *http, cups_dest_t *dest,
                      cups_dinfo_t *info, int job_id,
                      const char *docname,
                      const char *format,
                      int num_options,
                      cups_option_t *options,
                      int last_document);

http_status_t
cupsWriteRequestData(http_t *http, const char *buffer,
                     size_t length);

ipp_status_t
cupsFinishDestDocument(http_t *http, cups_dest_t *dest,
                       cups_dinfo_t *info);
```

The `docname` argument specifies the name of the document, typically the original filename. The `format` argument specifies the MIME media type of the document, including the following constants:

- `CUPS_FORMAT_JPEG`: "image/jpeg"
- `CUPS_FORMAT_PDF`: "application/pdf"
- `CUPS_FORMAT_POSTSCRIPT`: "application/postscript"
- `CUPS_FORMAT_TEXT`: "text/plain"

The `num_options` and `options` arguments specify per-document print options, which at present must be 0 and `NULL`. The `last_document` argument specifies whether this is the last document in the job.

For example, the following code submits a PDF file to the job that was just created:

```
FILE *fp = fopen("filename.pdf", "rb");
size_t bytes;
char buffer[65536];

if (cupsStartDestDocument(CUPS_HTTP_DEFAULT, dest, info,
                          job_id, "filename.pdf", 0, NULL,
                          1) == HTTP_STATUS_CONTINUE)
{
  while ((bytes = fread(buffer, 1, sizeof(buffer), fp)) > 0)
    if (cupsWriteRequestData(CUPS_HTTP_DEFAULT, buffer,
                             bytes) != HTTP_STATUS_CONTINUE)
      break;

  if (cupsFinishDestDocument(CUPS_HTTP_DEFAULT, dest,
                             info) == IPP_STATUS_OK)
    puts("Document send succeeded.");
  else
    printf("Document send failed: %s\n",
           cupsLastErrorString());
}

fclose(fp);
```

Sending IPP Requests

CUPS provides a rich API for sending IPP requests to the scheduler or printers, typically from management or utility applications whose primary purpose is not to send print jobs.

Connecting to the Scheduler or Printer

The connection to the scheduler or printer is represented by the HTTP connection type `http_t`. The `cupsConnectDest` function connects to the scheduler or printer associated with the destination:

```
http_t *
cupsConnectDest(cups_dest_t *dest, unsigned flags, int msec,
                int *cancel, char *resource,
                size_t resourcesize, cups_dest_cb_t cb,
                void *user_data);
```

The `dest` argument specifies the destination to connect to.

The `flags` argument specifies whether you want to connect to the scheduler (`CUPS_DEST_FLAGS_NONE`) or device/printer (`CUPS_DEST_FLAGS_DEVICE`) associated with the destination.

The `msec` argument specifies how long you are willing to wait for the connection to be established in milliseconds. Specify a value of `-1` to wait indefinitely.

The `cancel` argument specifies the address of an integer variable that can be set to a non-

zero value to cancel the connection. Specify a value of `NULL` to not provide a cancel variable.

The `resource` and `resourcesize` arguments specify the address and size of a character string array to hold the path to use when sending an IPP request.

The `cb` and `user_data` arguments specify a destination callback function that returns 1 to continue connecting or 0 to stop. The destination callback work the same way as the one used for the `cupsEnumDests` function.

On success, a HTTP connection is returned that can be used to send IPP requests and get IPP responses.

For example, the following code connects to the printer associated with a destination with a 30 second timeout:

```
char resource[256];
http_t *http = cupsConnectDest(dest, CUPS_DEST_FLAGS_DEVICE,
                               30000, NULL, resource,
                               sizeof(resource), NULL, NULL);
```

Creating an IPP Request

IPP requests are represented by the IPP message type `ipp_t` and each IPP attribute in the request is representing using the type `ipp_attribute_t`. Each IPP request includes an operation code (`IPP_OP_CREATE_JOB`, `IPP_OP_GET_PRINTER_ATTRIBUTES`, etc.) and a 32-bit integer identifier.

The `ippNewRequest` function creates a new IPP request:

```
ipp_t *
ippNewRequest(ipp_op_t op);
```

The `op` argument specifies the IPP operation code for the request. For example, the following code creates an IPP Get-Printer-Attributes request:

```
ipp_t *request = ippNewRequest(IPP_OP_GET_PRINTER_ATTRIBUTES);
```

The request identifier is automatically set to a unique value for the current process.

Each IPP request starts with two IPP attributes, "attributes-charset" and "attributes-natural-language", followed by IPP attribute(s) that specify the target of the operation. The `ippNewRequest` automatically adds the correct "attributes-charset" and "attributes-natural-language" attributes, but you must add the target attribute(s). For example, the following code adds the "printer-uri" attribute to the IPP Get-Printer-Attributes request to specify which printer is being queried:

```
const char *printer_uri = cupsGetOption("device-uri",
                                        dest->num_options,
                                        dest->options);

ippAddString(request, IPP_TAG_OPERATION, IPP_TAG_URI,
             "printer-uri", NULL, printer_uri);
```

> **Note:**
>
> If we wanted to query the scheduler instead of the device, we would look up the "printer-uri-supported" option instead of the "device-uri" value.

The `ippAddString` function adds the "printer-uri" attribute the the IPP request. The `IPP_TAG_OPERATION` argument specifies that the attribute is part of the operation. The `IPP_TAG_URI` argument specifies that the value is a Universal Resource Identifier (URI) string. The `NULL` argument specifies there is no language (English, French, Japanese, etc.) associated with the string, and the `printer_uri` argument specifies the string value.

The IPP Get-Printer-Attributes request also supports an IPP attribute called "requested-attributes" that lists the attributes and values you are interested in. For example, the following code requests the printer state attributes:

```
static const char * const requested_attributes[] =
{
  "printer-state",
  "printer-state-message",
  "printer-state-reasons"
};

ippAddStrings(request, IPP_TAG_OPERATION, IPP_TAG_KEYWORD,
              "requested-attributes", 3, NULL,
              requested_attributes);
```

The `ippAddStrings` function adds an attribute with one or more strings, in this case three. The `IPP_TAG_KEYWORD` argument specifies that the strings are keyword values, which are used for attribute names. All strings use the same language (`NULL`), and the attribute will contain the three strings in the array `requested_attributes`.

CUPS provides many functions to adding attributes of different types:

- `ippAddBoolean` adds a boolean (`IPP_TAG_BOOLEAN`) attribute with one value.
- `ippAddInteger` adds an enum (`IPP_TAG_ENUM`) or integer (`IPP_TAG_INTEGER`) attribute with one value.
- `ippAddIntegers` adds an enum or integer attribute with one or more values.
- `ippAddOctetString` adds an octetString attribute with one value.
- `ippAddOutOfBand` adds a admin-defined (`IPP_TAG_ADMINDEFINE`), default (`IPP_TAG_DEFAULT`), delete-attribute (`IPP_TAG_DELETEATTR`), no-value (`IPP_TAG_NOVALUE`), not-settable (`IPP_TAG_NOTSETTABLE`), unknown (`IPP_TAG_UNKNOWN`), or unsupported (`IPP_TAG_UNSUPPORTED_VALUE`) out-of-band attribute.
- `ippAddRange` adds a rangeOfInteger attribute with one range.

- `ippAddRanges` adds a rangeOfInteger attribute with one or more ranges.
- `ippAddResolution` adds a resolution attribute with one resolution.
- `ippAddResolutions` adds a resolution attribute with one or more resolutions.
- `ippAddString` adds a charset (`IPP_TAG_CHARSET`), keyword (`IPP_TAG_KEYWORD`), mimeMediaType (`IPP_TAG_MIMETYPE`), name (`IPP_TAG_NAME` and `IPP_TAG_NAMELANG`), naturalLanguage (`IPP_TAG_NATURAL_LANGUAGE`), text (`IPP_TAG_TEXT` and `IPP_TAG_TEXTLANG`), uri (`IPP_TAG_URI`), or uriScheme (`IPP_TAG_URISCHEME`) attribute with one value.
- `ippAddStrings` adds a charset, keyword, mimeMediaType, name, naturalLanguage, text, uri, or uriScheme attribute with one or more values.

Sending the IPP Request

Once you have created the IPP request, you can send it using the `cupsDoRequest` function. For example, the following code sends the IPP Get-Printer-Attributes request to the destination and saves the response:

```
ipp_t *response = cupsDoRequest(http, request, resource);
```

For requests like Send-Document that include a file, the `cupsDoFileRequest` function should be used:

```
ipp_t *response = cupsDoFileRequest(http, request, resource,
                                    filename);
```

Both `cupsDoRequest` and `cupsDoFileRequest` free the IPP request. If a valid IPP response is received, it is stored in a new IPP message (`ipp_t`) and returned to the caller. Otherwise `NULL` is returned.

The status from the most recent request can be queried using the `cupsLastError` function, for example:

```
if (cupsLastError() >= IPP_STATUS_ERROR_BAD_REQUEST)
{
  /* request failed */
}
```

A human-readable error message is also available using the `cupsLastErrorString` function:

```
if (cupsLastError() >= IPP_STATUS_ERROR_BAD_REQUEST)
{
  /* request failed */
  printf("Request failed: %s\n", cupsLastErrorString());
}
```

Processing the IPP Response

Each response to an IPP request is also an IPP message (`ipp_t`) with its own IPP attributes

(`ipp_attribute_t`) that includes a status code (`IPP_STATUS_OK`, `IPP_STATUS_ERROR_BAD_REQUEST`, etc.) and the corresponding 32-bit integer identifier from the request.

For example, the following code finds the printer state attributes and prints their values:

```
ipp_attribute_t *attr;

if ((attr = ippFindAttribute(response, "printer-state",
                             IPP_TAG_ENUM)) != NULL)
{
  printf("printer-state=%s\n",
         ippEnumString("printer-state", ippGetInteger(attr, 0)));
}
else
  puts("printer-state=unknown");

if ((attr = ippFindAttribute(response, "printer-state-message",
                             IPP_TAG_TEXT)) != NULL)
{
  printf("printer-state-message=\"%s\"\n",
         ippGetString(attr, 0, NULL)));
}

if ((attr = ippFindAttribute(response, "printer-state-reasons",
                             IPP_TAG_KEYWORD)) != NULL)
{
  int i, count = ippGetCount(attr);

  puts("printer-state-reasons=");
  for (i = 0; i < count; i ++)
    printf("    %s\n", ippGetString(attr, i, NULL)));
}
```

The `ippGetCount` function returns the number of values in an attribute.

The `ippGetInteger` and `ippGetString` functions return a single integer or string value from an attribute.

The `ippEnumString` function converts a enum value to its keyword (string) equivalent.

Once you are done using the IPP response message, free it using the `ippDelete` function:

```
ippDelete(response);
```

Authentication

CUPS normally handles authentication through the console. GUI applications should set a password callback using the `cupsSetPasswordCB2` function:

```
void
cupsSetPasswordCB2(cups_password_cb2_t cb, void *user_data);
```

The password callback will be called when needed and is responsible for setting the current user name using `cupsSetUser` and returning a string:

```
const char *
cups_password_cb2(const char *prompt, http_t *http,
                  const char *method, const char *resource,
                  void *user_data);
```

The `prompt` argument is a string from CUPS that should be displayed to the user.

The `http` argument is the connection hosting the request that is being authenticated. The password callback can call the `httpGetField` and `httpGetSubField` functions to look for additional details concerning the authentication challenge.

The `method` argument specifies the HTTP method used for the request and is typically "POST".

The `resource` argument specifies the path used for the request.

The `user_data` argument provides the user data pointer from the `cupsSetPasswordCB2` call.

Functions

cupsAddDest

Add a destination to the list of destinations.

```
int cupsAddDest(const char *name, const char *instance, int num_dests, cups_dest_t **dests);
```

Parameters

name	Destination name
instance	Instance name or `NULL` for none/primary
num_dests	Number of destinations
dests	Destinations

Return Value

New number of destinations

Discussion

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.

cupsAddDestMediaOptions

CUPS 2.3/macOS 10.14

Add the option corresponding to the specified media size.

```
int cupsAddDestMediaOptions(http_t *http, cups_dest_t *dest, cups_dinfo_t *dinfo, unsigned flags,
cups_size_t *size, int num_options, cups_option_t **options);
```

Parameters

http	Connection to destination
dest	Destination
dinfo	Destination information
flags	Media matching flags
size	Media size
num_options	Current number of options
options	Options

Return Value

New number of options

cupsAddIntegerOption

CUPS 2.2.4/macOS 10.13

Add an integer option to an option array.

```
int cupsAddIntegerOption(const char *name, int value, int num_options, cups_option_t **options);
```

Parameters

name	Name of option
value	Value of option
num_options	Number of options
options	Pointer to options

Return Value

Number of options

Discussion

New option arrays can be initialized simply by passing 0 for the "num_options" parameter.

cupsAddOption

Add an option to an option array.

```
int cupsAddOption(const char *name, const char *value, int num_options, cups_option_t **options);
```

Parameters

name	Name of option
value	Value of option
num_options	Number of options
options	Pointer to options

Return Value

Number of options

Discussion

New option arrays can be initialized simply by passing 0 for the "num_options" parameter.

cupsCancelDestJob

CUPS 1.6/macOS 10.8

Cancel a job on a destination.

```
ipp_status_t cupsCancelDestJob(http_t *http, cups_dest_t *dest, int job_id);
```

Parameters

http	Connection to destination
dest	Destination
job_id	Job ID

Return Value

Status of cancel operation

Discussion

The "job_id" is the number returned by cupsCreateDestJob.

Returns `IPP_STATUS_OK` on success and `IPP_STATUS_ERROR_NOT_AUTHORIZED` or `IPP_STATUS_ERROR_FORBIDDEN` on failure.

cupsCheckDestSupported

CUPS 1.6/macOS 10.8

Check that the option and value are supported by the destination.

```
int cupsCheckDestSupported(http_t *http, cups_dest_t *dest, cups_dinfo_t *dinfo, const char *option,
const char *value);
```

Parameters

http	Connection to destination
dest	Destination
dinfo	Destination information
option	Option
value	Value or `NULL`

Return Value

1 if supported, 0 otherwise

Discussion

Returns 1 if supported, 0 otherwise.

cupsCloseDestJob

CUPS 1.6/macOS 10.8

Close a job and start printing.

```
ipp_status_t cupsCloseDestJob(http_t *http, cups_dest_t *dest, cups_dinfo_t *info, int job_id);
```

Parameters

http	Connection to destination
dest	Destination

info	Destination information
job_id	Job ID

Return Value

IPP status code

Discussion

Use when the last call to cupsStartDocument passed 0 for "last_document". "job_id" is the job ID returned by cupsCreateDestJob. Returns `IPP_STATUS_OK` on success.

cupsConnectDest

CUPS 1.6/macOS 10.8

Open a connection to the destination.

```
http_t *cupsConnectDest(cups_dest_t *dest, unsigned flags, int msec, int *cancel, char *resource,
size_t resourcesize, cups_dest_cb_t cb, void *user_data);
```

Parameters

dest	Destination
flags	Connection flags
msec	Timeout in milliseconds
cancel	Pointer to "cancel" variable
resource	Resource buffer
resourcesize	Size of resource buffer
cb	Callback function
user_data	User data pointer

Return Value

Connection to destination or `NULL`

Discussion

Connect to the destination, returning a new `http_t` connection object and optionally the resource path to use for the destination. These calls will block until a connection is made, the timeout expires, the integer pointed to by "cancel" is non-zero, or the callback function (or block) returns 0. The caller is responsible for calling `httpClose` on the returned connection.

Starting with CUPS 2.2.4, the caller can pass `CUPS_DEST_FLAGS_DEVICE` for the "flags" argument to connect directly to the device associated with the destination. Otherwise, the connection is made to the CUPS scheduler associated with the destination.

cupsCopyDest

CUPS 1.6/macOS 10.8

Copy a destination.

```
int cupsCopyDest(cups_dest_t *dest, int num_dests, cups_dest_t **dests);
```

Parameters

dest	Destination to copy
num_dests	Number of destinations
dests	Destination array

Return Value

New number of destinations

Discussion

Make a copy of the destination to an array of destinations (or just a single copy) - for use with the cupsEnumDests* functions. The caller is responsible for calling cupsFreeDests() on the returned object(s).

cupsCopyDestConflicts

CUPS 1.6/macOS 10.8

Get conflicts and resolutions for a new option/value pair.

```
int cupsCopyDestConflicts(http_t *http, cups_dest_t *dest, cups_dinfo_t *dinfo, int num_options,
cups_option_t *options, const char *new_option, const char *new_value, int *num_conflicts,
cups_option_t **conflicts, int *num_resolved, cups_option_t **resolved);
```

Parameters

http	Connection to destination
dest	Destination
dinfo	Destination information
num_options	Number of current options
options	Current options

new_option	New option
new_value	New value
num_conflicts	Number of conflicting options
conflicts	Conflicting options
num_resolved	Number of options to resolve
resolved	Resolved options

Return Value

1 if there is a conflict, 0 if none, -1 on error

Discussion

"num_options" and "options" represent the currently selected options by the user. "new_option" and "new_value" are the setting the user has just changed.

Returns 1 if there is a conflict, 0 if there are no conflicts, and -1 if there was an unrecoverable error such as a resolver loop.

If "num_conflicts" and "conflicts" are not `NULL`, they are set to contain the list of conflicting option/value pairs. Similarly, if "num_resolved" and "resolved" are not `NULL` they will be set to the list of changes needed to resolve the conflict.

If cupsCopyDestConflicts returns 1 but "num_resolved" and "resolved" are set to 0 and `NULL`, respectively, then the conflict cannot be resolved.

cupsCopyDestInfo

CUPS 1.6/macOS 10.8

Get the supported values/capabilities for the destination.

```
cups_dinfo_t *cupsCopyDestInfo(http_t *http, cups_dest_t *dest);
```

Parameters

http	Connection to destination
dest	Destination

Return Value

Destination information

Discussion

The caller is responsible for calling `cupsFreeDestInfo` on the return value. `NULL` is returned on error.

cupsCreateDestJob

CUPS 1.6/macOS 10.8

Create a job on a destination.

```
ipp_status_t cupsCreateDestJob(http_t *http, cups_dest_t *dest, cups_dinfo_t *info, int *job_id, const
char *title, int num_options, cups_option_t *options);
```

Parameters

http	Connection to destination
dest	Destination
info	Destination information
job_id	Job ID or 0 on error
title	Job name
num_options	Number of job options
options	Job options

Return Value

IPP status code

Discussion

Returns `IPP_STATUS_OK` or `IPP_STATUS_OK_SUBST` on success, saving the job ID in the variable pointed to by "job_id".

cupsDoAuthentication

CUPS 1.1.20/macOS 10.4

Authenticate a request.

```
int cupsDoAuthentication(http_t *http, const char *method, const char *resource);
```

Parameters

http	Connection to server or `CUPS_HTTP_DEFAULT`
method	Request method ("GET", "POST", "PUT")
resource	Resource path

Return Value

0 on success, -1 on error

Discussion

This function should be called in response to a `HTTP_STATUS_UNAUTHORIZED` status, prior to resubmitting your request.

cupsEncodeOption

CUPS 2.3/macOS 10.14

Encode a single option into an IPP attribute.

```
ipp_attribute_t *cupsEncodeOption(ipp_t *ipp, ipp_tag_t group_tag, const char *name, const char
*value);
```

Parameters

ipp	IPP request/response
group_tag	Attribute group
name	Option name
value	Option string value

Return Value

New attribute or `NULL` on error

cupsEncodeOptions

Encode printer options into IPP attributes.

```
void cupsEncodeOptions(ipp_t *ipp, int num_options, cups_option_t *options);
```

Parameters

ipp	IPP request/response
num_options	Number of options
options	Options

Discussion

This function adds operation, job, and then subscription attributes, in that order. Use the

`cupsEncodeOptions2` function to add attributes for a single group.

cupsEncodeOptions2

CUPS 1.2/macOS 10.5

Encode printer options into IPP attributes for a group.

```
void cupsEncodeOptions2(ipp_t *ipp, int num_options, cups_option_t *options, ipp_tag_t group_tag);
```

Parameters

ipp	IPP request/response
num_options	Number of options
options	Options
group_tag	Group to encode

Discussion

This function only adds attributes for a single group. Call this function multiple times for each group, or use `cupsEncodeOptions` to add the standard groups.

cupsEncryption

Get the current encryption settings.

```
http_encryption_t cupsEncryption(void);
```

Return Value

Encryption settings

Discussion

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

Note: The current encryption setting is tracked separately for each thread in a program. Multi-threaded programs that override the setting via the `cupsSetEncryption` function need to do so in each thread for the same setting to be used.

cupsEnumDests

CUPS 1.6/macOS 10.8

Enumerate available destinations with a callback function.

```
int cupsEnumDests(unsigned flags, int msec, int *cancel, cups_ptype_t type, cups_ptype_t mask,
cups_dest_cb_t cb, void *user_data);
```

Parameters

flags	Enumeration flags
msec	Timeout in milliseconds, -1 for indefinite
cancel	Pointer to "cancel" variable
type	Printer type bits
mask	Mask for printer type bits
cb	Callback function
user_data	User data

Return Value

1 on success, 0 on failure

Discussion

Destinations are enumerated from one or more sources. The callback function receives the `user_data` pointer and the destination pointer which can be used as input to the `cupsCopyDest` function. The function must return 1 to continue enumeration or 0 to stop.

The `type` and `mask` arguments allow the caller to filter the destinations that are enumerated. Passing 0 for both will enumerate all printers. The constant `CUPS_PRINTER_DISCOVERED` is used to filter on destinations that are available but have not yet been added locally.

Enumeration happens on the current thread and does not return until all destinations have been enumerated or the callback function returns 0.

Note: The callback function will likely receive multiple updates for the same destinations - it is up to the caller to suppress any duplicate destinations.

cupsFindDestDefault

CUPS 1.7/macOS 10.9

Find the default value(s) for the given option.

```
ipp_attribute_t *cupsFindDestDefault(http_t *http, cups_dest_t *dest, cups_dinfo_t *dinfo, const char
*option);
```

Parameters

http	Connection to destination
dest	Destination
dinfo	Destination information
option	Option/attribute name

Return Value

Default attribute or `NULL` for none

Discussion

The returned value is an IPP attribute. Use the `ippGetBoolean`, `ippGetCollection`, `ippGetCount`, `ippGetDate`, `ippGetInteger`, `ippGetOctetString`, `ippGetRange`, `ippGetResolution`, `ippGetString`, and `ippGetValueTag` functions to inspect the default value(s) as needed.

cupsFindDestReady

CUPS 1.7/macOS 10.9

Find the default value(s) for the given option.

```
ipp_attribute_t *cupsFindDestReady(http_t *http, cups_dest_t *dest, cups_dinfo_t *dinfo, const char
*option);
```

Parameters

http	Connection to destination
dest	Destination
dinfo	Destination information
option	Option/attribute name

Return Value

Default attribute or `NULL` for none

Discussion

The returned value is an IPP attribute. Use the `ippGetBoolean`, `ippGetCollection`, `ippGetCount`, `ippGetDate`, `ippGetInteger`, `ippGetOctetString`, `ippGetRange`, `ippGetResolution`, `ippGetString`, and `ippGetValueTag` functions to inspect the default value(s) as needed.

cupsFindDestSupported

CUPS 1.7/macOS 10.9

Find the default value(s) for the given option.

```
ipp_attribute_t *cupsFindDestSupported(http_t *http, cups_dest_t *dest, cups_dinfo_t *dinfo, const
char *option);
```

Parameters

http	Connection to destination
dest	Destination
dinfo	Destination information
option	Option/attribute name

Return Value

Default attribute or `NULL` for none

Discussion

The returned value is an IPP attribute. Use the `ippGetBoolean`, `ippGetCollection`, `ippGetCount`, `ippGetDate`, `ippGetInteger`, `ippGetOctetString`, `ippGetRange`, `ippGetResolution`, `ippGetString`, and `ippGetValueTag` functions to inspect the default value(s) as needed.

cupsFinishDestDocument

CUPS 1.6/macOS 10.8

Finish the current document.

```
ipp_status_t cupsFinishDestDocument(http_t *http, cups_dest_t *dest, cups_dinfo_t *info);
```

Parameters

http	Connection to destination
dest	Destination
info	Destination information

Return Value

Status of document submission

Discussion

Returns `IPP_STATUS_OK` or `IPP_STATUS_OK_SUBST` on success.

cupsFreeDestInfo

CUPS 1.6/macOS 10.8

Free destination information obtained using `cupsCopyDestInfo`.

```
void cupsFreeDestInfo(cups_dinfo_t *dinfo);
```

Parameters

dinfo	Destination information

cupsFreeDests

Free the memory used by the list of destinations.

```
void cupsFreeDests(int num_dests, cups_dest_t *dests);
```

Parameters

num_dests	Number of destinations
dests	Destinations

cupsFreeJobs

Free memory used by job data.

```
void cupsFreeJobs(int num_jobs, cups_job_t *jobs);
```

Parameters

num_jobs	Number of jobs
jobs	Jobs

cupsFreeOptions

Free all memory used by options.

```
void cupsFreeOptions(int num_options, cups_option_t *options);
```

Parameters

num_options	Number of options

options	Pointer to options

cupsGetDest

Get the named destination from the list.

```
cups_dest_t *cupsGetDest(const char *name, const char *instance, int num_dests, cups_dest_t *dests);
```

Parameters

name	Destination name or `NULL` for the default destination
instance	Instance name or `NULL`
num_dests	Number of destinations
dests	Destinations

Return Value

Destination pointer or `NULL`

Discussion

Use the `cupsEnumDests` or `cupsGetDests2` functions to get a list of supported destinations for the current user.

cupsGetDestMediaByIndex *CUPS 1.7/macOS 10.9*

Get a media name, dimension, and margins for a specific size.

```
int cupsGetDestMediaByIndex(http_t *http, cups_dest_t *dest, cups_dinfo_t *dinfo, int n, unsigned
flags, cups_size_t *size);
```

Parameters

http	Connection to destination
dest	Destination
dinfo	Destination information
n	Media size number (0-based)
flags	Media flags
size	Media size information

Return Value

1 on success, 0 on failure

Discussion

The `flags` parameter determines which set of media are indexed. For example, passing `CUPS_MEDIA_FLAGS_BORDERLESS` will get the Nth borderless size supported by the printer.

cupsGetDestMediaByName

CUPS 1.6/macOS 10.8

Get media names, dimensions, and margins.

```
int cupsGetDestMediaByName(http_t *http, cups_dest_t *dest, cups_dinfo_t *dinfo, const char *media,
unsigned flags, cups_size_t *size);
```

Parameters

http	Connection to destination
dest	Destination
dinfo	Destination information
media	Media name
flags	Media matching flags
size	Media size information

Return Value

1 on match, 0 on failure

Discussion

The "media" string is a PWG media name. "Flags" provides some matching guidance (multiple flags can be combined):

CUPS_MEDIA_FLAGS_DEFAULT = find the closest size supported by the printer, CUPS_MEDIA_FLAGS_BORDERLESS = find a borderless size, CUPS_MEDIA_FLAGS_DUPLEX = find a size compatible with 2-sided printing, CUPS_MEDIA_FLAGS_EXACT = find an exact match for the size, and CUPS_MEDIA_FLAGS_READY = if the printer supports media sensing, find the size amongst the "ready" media.

The matching result (if any) is returned in the "cups_size_t" structure.

Returns 1 when there is a match and 0 if there is not a match.

cupsGetDestMediaBySize

CUPS 1.6/macOS 10.8

Get media names, dimensions, and margins.

```
int cupsGetDestMediaBySize(http_t *http, cups_dest_t *dest, cups_dinfo_t *dinfo, int width, int
length, unsigned flags, cups_size_t *size);
```

Parameters

http	Connection to destination
dest	Destination
dinfo	Destination information
width	Media width in hundredths of of millimeters
length	Media length in hundredths of of millimeters
flags	Media matching flags
size	Media size information

Return Value

1 on match, 0 on failure

Discussion

"Width" and "length" are the dimensions in hundredths of millimeters. "Flags" provides some matching guidance (multiple flags can be combined):

CUPS_MEDIA_FLAGS_DEFAULT = find the closest size supported by the printer, CUPS_MEDIA_FLAGS_BORDERLESS = find a borderless size, CUPS_MEDIA_FLAGS_DUPLEX = find a size compatible with 2-sided printing, CUPS_MEDIA_FLAGS_EXACT = find an exact match for the size, and CUPS_MEDIA_FLAGS_READY = if the printer supports media sensing, find the size amongst the "ready" media.

The matching result (if any) is returned in the "cups_size_t" structure.

Returns 1 when there is a match and 0 if there is not a match.

cupsGetDestMediaCount

CUPS 1.7/macOS 10.9

Get the number of sizes supported by a destination.

```
int cupsGetDestMediaCount(http_t *http, cups_dest_t *dest, cups_dinfo_t *dinfo, unsigned flags);
```

Parameters

http	Connection to destination
dest	Destination
dinfo	Destination information
flags	Media flags

Return Value

Number of sizes

Discussion

The `flags` parameter determines the set of media sizes that are counted. For example, passing `CUPS_MEDIA_FLAGS_BORDERLESS` will return the number of borderless sizes.

cupsGetDestMediaDefault

CUPS 1.7/macOS 10.9

Get the default size for a destination.

```
int cupsGetDestMediaDefault(http_t *http, cups_dest_t *dest, cups_dinfo_t *dinfo, unsigned flags,
cups_size_t *size);
```

Parameters

http	Connection to destination
dest	Destination
dinfo	Destination information
flags	Media flags
size	Media size information

Return Value

1 on success, 0 on failure

Discussion

The `flags` parameter determines which default size is returned. For example, passing `CUPS_MEDIA_FLAGS_BORDERLESS` will return the default borderless size, typically US Letter or A4, but sometimes 4x6 photo media.

cupsGetDestWithURI

CUPS 2.0/macOS 10.10

Get a destination associated with a URI.

```
cups_dest_t *cupsGetDestWithURI(const char *name, const char *uri);
```

Parameters

name	Desired printer name or `NULL`
uri	URI for the printer

Return Value

Destination or `NULL`

Discussion

"name" is the desired name for the printer. If `NULL`, a name will be created using the URI.

"uri" is the "ipp" or "ipps" URI for the printer.

cupsGetDests2

CUPS 1.1.21/macOS 10.4

Get the list of destinations from the specified server.

```
int cupsGetDests2(http_t *http, cups_dest_t **dests);
```

Parameters

http	Connection to server or `CUPS_HTTP_DEFAULT`
dests	Destinations

Return Value

Number of destinations

Discussion

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", "printer-type", and "printer-uri-supported" attributes as options.

CUPS 1.4 adds the "marker-change-time", "marker-colors", "marker-high-levels", "marker-

levels", "marker-low-levels", "marker-message", "marker-names", "marker-types", and "printer-commands" attributes as options.

CUPS 2.2 adds accessible IPP printers to the list of destinations that can be used. The "printer-uri-supported" option will be present for those IPP printers that have been recently used.

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

cupsGetIntegerOption

CUPS 2.2.4/macOS 10.13

Get an integer option value.

```
int cupsGetIntegerOption(const char *name, int num_options, cups_option_t *options);
```

Parameters

name	Name of option
num_options	Number of options
options	Options

Return Value

Option value or `INT_MIN`

Discussion

INT_MIN is returned when the option does not exist, is not an integer, or exceeds the range of values for the "int" type.

cupsGetJobs2

CUPS 1.1.21/macOS 10.4

Get the jobs from the specified server.

```
int cupsGetJobs2(http_t *http, cups_job_t **jobs, const char *name, int myjobs, int whichjobs);
```

Parameters

http	Connection to server or `CUPS_HTTP_DEFAULT`
jobs	Job data
name	`NULL` = all destinations, otherwise show jobs for named destination

myjobs	0 = all users, 1 = mine
whichjobs	`CUPS_WHICHJOBS_ALL`, `CUPS_WHICHJOBS_ACTIVE`, or `CUPS_WHICHJOBS_COMPLETED`

Return Value

Number of jobs

Discussion

A "whichjobs" value of `CUPS_WHICHJOBS_ALL` returns all jobs regardless of state, while `CUPS_WHICHJOBS_ACTIVE` returns jobs that are pending, processing, or held and `CUPS_WHICHJOBS_COMPLETED` returns jobs that are stopped, canceled, aborted, or completed.

cupsGetNamedDest

CUPS 1.4/macOS 10.6

Get options for the named destination.

```
cups_dest_t *cupsGetNamedDest(http_t *http, const char *name, const char *instance);
```

Parameters

http	Connection to server or `CUPS_HTTP_DEFAULT`
name	Destination name or `NULL` for the default destination
instance	Instance name or `NULL`

Return Value

Destination or `NULL`

Discussion

This function is optimized for retrieving a single destination and should be used instead of `cupsGetDests2` and `cupsGetDest` when you either know the name of the destination or want to print to the default destination. If `NULL` is returned, the destination does not exist or there is no default destination.

If "http" is `CUPS_HTTP_DEFAULT`, the connection to the default print server will be used.

If "name" is `NULL`, the default printer for the current user will be returned.

The returned destination must be freed using `cupsFreeDests` with a "num_dests" value of 1.

cupsGetOption

Get an option value.

```
const char *cupsGetOption(const char *name, int num_options, cups_option_t *options);
```

Parameters

name	Name of option
num_options	Number of options
options	Options

Return Value

Option value or `NULL`

cupsGetPassword2

CUPS 1.4/macOS 10.6

Get a password from the user using the current password callback.

```
const char *cupsGetPassword2(const char *prompt, http_t *http, const char *method, const char
*resource);
```

Parameters

prompt	Prompt string
http	Connection to server or `CUPS_HTTP_DEFAULT`
method	Request method ("GET", "POST", "PUT")
resource	Resource path

Return Value

Password

Discussion

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

Note: The current password callback function is tracked separately for each thread in a program. Multi-threaded programs that override the setting via the `cupsSetPasswordCB2` function need to do so in each thread for the same function to be used.

cupsLocalizeDestMedia

CUPS 2.0/macOS 10.10

Get the localized string for a destination media size.

```
const char *cupsLocalizeDestMedia(http_t *http, cups_dest_t *dest, cups_dinfo_t *dinfo, unsigned
flags, cups_size_t *size);
```

Parameters

http	Connection to destination
dest	Destination
dinfo	Destination information
flags	Media flags
size	Media size

Return Value

Localized string

Discussion

The returned string is stored in the destination information and will become invalid if the destination information is deleted.

cupsLocalizeDestOption

CUPS 1.6/macOS 10.8

Get the localized string for a destination option.

```
const char *cupsLocalizeDestOption(http_t *http, cups_dest_t *dest, cups_dinfo_t *dinfo, const char
*option);
```

Parameters

http	Connection to destination
dest	Destination
dinfo	Destination information
option	Option to localize

Return Value

Localized string

Discussion

The returned string is stored in the destination information and will become invalid if the destination information is deleted.

cupsLocalizeDestValue

CUPS 1.6/macOS 10.8

Get the localized string for a destination option+value pair.

```
const char *cupsLocalizeDestValue(http_t *http, cups_dest_t *dest, cups_dinfo_t *dinfo, const char
*option, const char *value);
```

Parameters

http	Connection to destination
dest	Destination
dinfo	Destination information
option	Option to localize
value	Value to localize

Return Value

Localized string

Discussion

The returned string is stored in the destination information and will become invalid if the destination information is deleted.

cupsMakeServerCredentials

CUPS 2.0/OS 10.10

Make a self-signed certificate and private key pair.

```
int cupsMakeServerCredentials(const char *path, const char *common_name, int num_alt_names, const char
**alt_names, time_t expiration_date);
```

Parameters

path	Keychain path or `NULL` for default
common_name	Common name
num_alt_names	Number of subject alternate names
alt_names	Subject Alternate Names

expiration_date	Expiration date

Return Value

1 on success, 0 on failure

cupsParseOptions

Parse options from a command-line argument.

```
int cupsParseOptions(const char *arg, int num_options, cups_option_t **options);
```

Parameters

arg	Argument to parse
num_options	Number of options
options	Options found

Return Value

Number of options found

Discussion

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.

cupsRemoveDest

CUPS 1.3/macOS 10.5

Remove a destination from the destination list.

```
int cupsRemoveDest(const char *name, const char *instance, int num_dests, cups_dest_t **dests);
```

Parameters

name	Destination name
instance	Instance name or `NULL`
num_dests	Number of destinations
dests	Destinations

Return Value

New number of destinations

Discussion

Removing a destination/instance does not delete the class or printer queue, merely the lpoptions for that destination/instance. Use the `cupsSetDests` or `cupsSetDests2` functions to save the new options for the user.

cupsRemoveOption

CUPS 1.2/macOS 10.5

Remove an option from an option array.

```
int cupsRemoveOption(const char *name, int num_options, cups_option_t **options);
```

Parameters

name	Option name
num_options	Current number of options
options	Options

Return Value

New number of options

cupsServer

Return the hostname/address of the current server.

```
const char *cupsServer(void);
```

Return Value

Server name

Discussion

The default server comes from the CUPS_SERVER environment variable, then the ~/.cups/client.conf file, and finally the /etc/cups/client.conf file. If not set, the default is the local system - either "localhost" or a domain socket path.

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

Note: The current server is tracked separately for each thread in a program. Multi-threaded programs that override the server via the `cupsSetServer` function need to do so in each thread for the same server to be used.

cupsSetClientCertCB

CUPS 1.5/macOS 10.7

Set the client certificate callback.

```
void cupsSetClientCertCB(cups_client_cert_cb_t cb, void *user_data);
```

Parameters

cb	Callback function
user_data	User data pointer

Discussion

Pass `NULL` to restore the default callback.

Note: The current certificate callback is tracked separately for each thread in a program. Multi-threaded programs that override the callback need to do so in each thread for the same callback to be used.

cupsSetCredentials

CUPS 1.5/macOS 10.7

Set the default credentials to be used for SSL/TLS connections.

```
int cupsSetCredentials(cups_array_t *credentials);
```

Parameters

credentials	Array of credentials

Return Value

Status of call (0 = success)

Discussion

Note: The default credentials are tracked separately for each thread in a program. Multi-threaded programs that override the setting need to do so in each thread for the same setting to be used.

cupsSetDefaultDest

CUPS 1.3/macOS 10.5

Set the default destination.

```
void cupsSetDefaultDest(const char *name, const char *instance, int num_dests, cups_dest_t *dests);
```

Parameters

name	Destination name
instance	Instance name or `NULL`
num_dests	Number of destinations
dests	Destinations

cupsSetDests2

CUPS 1.1.21/macOS 10.4

Save the list of destinations for the specified server.

```
int cupsSetDests2(http_t *http, int num_dests, cups_dest_t *dests);
```

Parameters

http	Connection to server or `CUPS_HTTP_DEFAULT`
num_dests	Number of destinations
dests	Destinations

Return Value

0 on success, -1 on error

Discussion

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

cupsSetEncryption

Set the encryption preference.

```
void cupsSetEncryption(http_encryption_t e);
```

Parameters

e	New encryption preference

Discussion

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

Note: The current encryption setting is tracked separately for each thread in a program. Multi-threaded programs that override the setting need to do so in each thread for the same setting to be used.

cupsSetPasswordCB2

CUPS 1.4/macOS 10.6

Set the advanced password callback for CUPS.

```
void cupsSetPasswordCB2(cups_password_cb2_t cb, void *user_data);
```

Parameters

cb	Callback function
user_data	User data pointer

Discussion

Pass `NULL` to restore the default (console) password callback, which reads the password from the console. Programs should call either this function or `cupsSetPasswordCB2`, as only one callback can be registered by a program per thread.

Note: The current password callback is tracked separately for each thread in a program. Multi-threaded programs that override the callback need to do so in each thread for the same callback to be used.

cupsSetServer

Set the default server name and port.

```
void cupsSetServer(const char *server);
```

Parameters

server	Server name

Discussion

The "server" string can be a fully-qualified hostname, a numeric IPv4 or IPv6 address, or a domain socket pathname. Hostnames and numeric IP addresses can be optionally followed by a colon and port number to override the default port 631, e.g. "hostname:8631". Pass `NULL` to restore the default server name and port.

Note: The current server is tracked separately for each thread in a program. Multi-threaded programs that override the server need to do so in each thread for the same server to be used.

cupsSetServerCertCB

CUPS 1.5/macOS 10.7

Set the server certificate callback.

```
void cupsSetServerCertCB(cups_server_cert_cb_t cb, void *user_data);
```

Parameters

cb	Callback function
user_data	User data pointer

Discussion

Pass `NULL` to restore the default callback.

Note: The current credentials callback is tracked separately for each thread in a program. Multi-threaded programs that override the callback need to do so in each thread for the same callback to be used.

cupsSetServerCredentials

CUPS 2.0/macOS 10.10

Set the default server credentials.

```
int cupsSetServerCredentials(const char *path, const char *common_name, int auto_create);
```

Parameters

path	Keychain path or `NULL` for default
common_name	Default common name for server
auto_create	1 = automatically create self-signed certificates

Return Value

1 on success, 0 on failure

Discussion

Note: The server credentials are used by all threads in the running process. This function is threadsafe.

cupsSetUser

Set the default user name.

```
void cupsSetUser(const char *user);
```

Parameters

user	User name

Discussion

Pass `NULL` to restore the default user name.

Note: The current user name is tracked separately for each thread in a program. Multi-threaded programs that override the user name need to do so in each thread for the same user name to be used.

cupsSetUserAgent *CUPS 1.7/macOS 10.9*

Set the default HTTP User-Agent string.

```
void cupsSetUserAgent(const char *user_agent);
```

Parameters

user_agent	User-Agent string or `NULL`

Discussion

Setting the string to NULL forces the default value containing the CUPS version, IPP version, and operating system version and architecture.

cupsStartDestDocument *CUPS 1.6/macOS 10.8*

Start a new document.

```
http_status_t cupsStartDestDocument(http_t *http, cups_dest_t *dest, cups_dinfo_t *info, int job_id,
const char *docname, const char *format, int num_options, cups_option_t *options, int last_document);
```

Parameters

http	Connection to destination
dest	Destination
info	Destination information
job_id	Job ID
docname	Document name
format	Document format
num_options	Number of document options
options	Document options
last_document	1 if this is the last document

Return Value

Status of document creation

Discussion

"job_id" is the job ID returned by cupsCreateDestJob. "docname" is the name of the document/file being printed, "format" is the MIME media type for the document (see CUPS_FORMAT_xxx constants), and "num_options" and "options" are the options do be applied to the document. "last_document" should be 1 if this is the last document to be submitted in the job. Returns `HTTP_CONTINUE` on success.

cupsUser

Return the current user's name.

```
const char *cupsUser(void);
```

Return Value

User name

Discussion

Note: The current user name is tracked separately for each thread in a program. Multi-

threaded programs that override the user name with the `cupsSetUser` function need to do so in each thread for the same user name to be used.

cupsUserAgent

CUPS 1.7/macOS 10.9

Return the default HTTP User-Agent string.

```
const char *cupsUserAgent(void);
```

Return Value

User-Agent string

httpAcceptConnection

CUPS 1.7/macOS 10.9

Accept a new HTTP client connection from the specified listening socket.

```
http_t *httpAcceptConnection(int fd, int blocking);
```

Parameters

fd	Listen socket file descriptor
blocking	1 if the connection should be blocking, 0 otherwise

Return Value

HTTP connection or `NULL`

httpAddCredential

CUPS 1.5/macOS 10.7

Allocates and adds a single credential to an array.

```
int httpAddCredential(cups_array_t *credentials, const void *data, size_t datalen);
```

Parameters

credentials	Credentials array
data	PEM-encoded X.509 data
datalen	Length of data

Return Value

0 on success, -1 on error

Discussion

Use `cupsArrayNew(NULL, NULL)` to create a credentials array.

httpAddrAny

CUPS 1.2/macOS 10.5

Check for the "any" address.

```
int httpAddrAny(const http_addr_t *addr);
```

Parameters

addr	Address to check

Return Value

1 if "any", 0 otherwise

httpAddrClose

CUPS 2.0/OS 10.10

Close a socket created by `httpAddrConnect` or `httpAddrListen`.

```
int httpAddrClose(http_addr_t *addr, int fd);
```

Parameters

addr	Listen address or `NULL`
fd	Socket file descriptor

Return Value

0 on success, -1 on failure

Discussion

Pass `NULL` for sockets created with `httpAddrConnect2` and the listen address for sockets created with `httpAddrListen`. This function ensures that domain sockets are removed when closed.

httpAddrConnect2

CUPS 1.7/macOS 10.9

Connect to any of the addresses in the list with a timeout and optional cancel.

```
http_addrlist_t *httpAddrConnect2(http_addrlist_t *addrlist, int *sock, int msec, int *cancel);
```

Parameters

addrlist	List of potential addresses
sock	Socket
msec	Timeout in milliseconds
cancel	Pointer to "cancel" variable

Return Value

Connected address or NULL on failure

httpAddrCopyList

CUPS 1.7/macOS 10.9

Copy an address list.

```
http_addrlist_t *httpAddrCopyList(http_addrlist_t *src);
```

Parameters

src	Source address list

Return Value

New address list or `NULL` on error

httpAddrEqual

CUPS 1.2/macOS 10.5

Compare two addresses.

```
int httpAddrEqual(const http_addr_t *addr1, const http_addr_t *addr2);
```

Parameters

addr1	First address
addr2	Second address

Return Value

1 if equal, 0 if not

httpAddrFamily

Get the address family of an address.

```
int httpAddrFamily(http_addr_t *addr);
```

Parameters

addr	Address

Return Value

Address family

httpAddrFreeList

CUPS 1.2/macOS 10.5

Free an address list.

```
void httpAddrFreeList(http_addrlist_t *addrlist);
```

Parameters

addrlist	Address list to free

httpAddrGetList

CUPS 1.2/macOS 10.5

Get a list of addresses for a hostname.

```
http_addrlist_t *httpAddrGetList(const char *hostname, int family, const char *service);
```

Parameters

hostname	Hostname, IP address, or NULL for passive listen address
family	Address family or AF_UNSPEC
service	Service name or port number

Return Value

List of addresses or NULL

httpAddrLength

CUPS 1.2/macOS 10.5

Return the length of the address in bytes.

```
int httpAddrLength(const http_addr_t *addr);
```

Parameters

addr	Address

Return Value

Length in bytes

httpAddrListen

CUPS 1.7/macOS 10.9

Create a listening socket bound to the specified address and port.

```
int httpAddrListen(http_addr_t *addr, int port);
```

Parameters

addr	Address to bind to
port	Port number to bind to

Return Value

Socket or -1 on error

httpAddrLocalhost

CUPS 1.2/macOS 10.5

Check for the local loopback address.

```
int httpAddrLocalhost(const http_addr_t *addr);
```

Parameters

addr	Address to check

Return Value

1 if local host, 0 otherwise

httpAddrLookup

CUPS 1.2/macOS 10.5

Lookup the hostname associated with the address.

```
char *httpAddrLookup(const http_addr_t *addr, char *name, int namelen);
```

Parameters

addr	Address to lookup
name	Host name buffer
namelen	Size of name buffer

Return Value

Host name

httpAddrPort

CUPS 1.7/macOS 10.9

Get the port number associated with an address.

```
int httpAddrPort(http_addr_t *addr);
```

Parameters

addr	Address

Return Value

Port number

httpAddrString

CUPS 1.2/macOS 10.5

Convert an address to a numeric string.

```
char *httpAddrString(const http_addr_t *addr, char *s, int slen);
```

Parameters

addr	Address to convert
s	String buffer
slen	Length of string

Return Value

Numeric address string

httpAssembleURI

CUPS 1.2/macOS 10.5

Assemble a uniform resource identifier from its components.

```
http_uri_status_t httpAssembleURI(http_uri_coding_t encoding, char *uri, int urilen, const char
*scheme, const char *username, const char *host, int port, const char *resource);
```

Parameters

encoding	Encoding flags
uri	URI buffer
urilen	Size of URI buffer
scheme	Scheme name
username	Username
host	Hostname or address
port	Port number
resource	Resource

Return Value

URI status

Discussion

This function escapes reserved characters in the URI depending on the value of the "encoding" argument. You should use this function in place of traditional string functions whenever you need to create a URI string.

httpAssembleURIf

CUPS 1.2/macOS 10.5

Assemble a uniform resource identifier from its components with a formatted resource.

```
http_uri_status_t httpAssembleURIf(http_uri_coding_t encoding, char *uri, int urilen, const char
*scheme, const char *username, const char *host, int port, const char *resourcef, ...);
```

Parameters

encoding	Encoding flags

uri	URI buffer
urilen	Size of URI buffer
scheme	Scheme name
username	Username
host	Hostname or address
port	Port number
resourcef	Printf-style resource
...	Additional arguments as needed

Return Value

URI status

Discussion

This function creates a formatted version of the resource string argument "resourcef" and escapes reserved characters in the URI depending on the value of the "encoding" argument. You should use this function in place of traditional string functions whenever you need to create a URI string.

httpAssembleUUID

CUPS 1.7/macOS 10.9

Assemble a name-based UUID URN conforming to RFC 4122.

```
char *httpAssembleUUID(const char *server, int port, const char *name, int number, char *buffer,
size_t bufsize);
```

Parameters

server	Server name
port	Port number
name	Object name or NULL
number	Object number or 0
buffer	String buffer
bufsize	Size of buffer

Return Value

UUID string

Discussion

This function creates a unique 128-bit identifying number using the server name, port number, random data, and optionally an object name and/or object number. The result is formatted as a UUID URN as defined in RFC 4122.

The buffer needs to be at least 46 bytes in size.

httpBlocking

Set blocking/non-blocking behavior on a connection.

```
void httpBlocking(http_t *http, int b);
```

Parameters

http	HTTP connection
b	1 = blocking, 0 = non-blocking

httpCheck

Check to see if there is a pending response from the server.

```
int httpCheck(http_t *http);
```

Parameters

http	HTTP connection

Return Value

0 = no data, 1 = data available

httpClearCookie *CUPS 1.1.19/macOS 10.3*

Clear the cookie value(s).

```
void httpClearCookie(http_t *http);
```

Parameters

http	HTTP connection

httpClearFields

Clear HTTP request fields.

```
void httpClearFields(http_t *http);
```

Parameters

http	HTTP connection

httpClose

Close an HTTP connection.

```
void httpClose(http_t *http);
```

Parameters

http	HTTP connection

httpCompareCredentials

CUPS 2.0/OS 10.10

Compare two sets of X.509 credentials.

```
int httpCompareCredentials(cups_array_t *cred1, cups_array_t *cred2);
```

Parameters

cred1	First set of X.509 credentials
cred2	Second set of X.509 credentials

Return Value

1 if they match, 0 if they do not

httpConnect2

CUPS 1.7/macOS 10.9

Connect to a HTTP server.

```
http_t *httpConnect2(const char *host, int port, http_addrlist_t *addrlist, int family,
http_encryption_t encryption, int blocking, int msec, int *cancel);
```

Parameters

host	Host to connect to
port	Port number
addrlist	List of addresses or `NULL` to lookup
family	Address family to use or `AF_UNSPEC` for any
encryption	Type of encryption to use
blocking	1 for blocking connection, 0 for non-blocking
msec	Connection timeout in milliseconds, 0 means don't connect
cancel	Pointer to "cancel" variable

Return Value

New HTTP connection

httpCopyCredentials

CUPS 1.5/macOS 10.7

Copy the credentials associated with the peer in an encrypted connection.

```
int httpCopyCredentials(http_t *http, cups_array_t **credentials);
```

Parameters

http	Connection to server
credentials	Array of credentials

Return Value

Status of call (0 = success)

httpCredentialsAreValidForName

CUPS 2.0/macOS 10.10

Return whether the credentials are valid for the given name.

```
int httpCredentialsAreValidForName(cups_array_t *credentials, const char *common_name);
```

Parameters

credentials	Credentials
common_name	Name to check

Return Value

1 if valid, 0 otherwise

httpCredentialsGetExpiration CUPS 2.0/macOS 10.10

Return the expiration date of the credentials.

```
time_t httpCredentialsGetExpiration(cups_array_t *credentials);
```

Parameters

credentials	Credentials

Return Value

Expiration date of credentials

httpCredentialsGetTrust CUPS 2.0/macOS 10.10

Return the trust of credentials.

```
http_trust_t httpCredentialsGetTrust(cups_array_t *credentials, const char *common_name);
```

Parameters

credentials	Credentials
common_name	Common name for trust lookup

Return Value

Level of trust

httpCredentialsString CUPS 2.0/macOS 10.10

Return a string representing the credentials.

```
size_t httpCredentialsString(cups_array_t *credentials, char *buffer, size_t bufsize);
```

Parameters

credentials	Credentials
buffer	Buffer or `NULL`

bufsize	Size of buffer

Return Value

Total size of credentials string

httpDecode64_2

CUPS 1.1.21/macOS 10.4

Base64-decode a string.

```
char *httpDecode64_2(char *out, int *outlen, const char *in);
```

Parameters

out	String to write to
outlen	Size of output string
in	String to read from

Return Value

Decoded string

Discussion

The caller must initialize "outlen" to the maximum size of the decoded string before calling `httpDecode64_2`. On return "outlen" contains the decoded length of the string.

httpDelete

Send a DELETE request to the server.

```
int httpDelete(http_t *http, const char *uri);
```

Parameters

http	HTTP connection
uri	URI to delete

Return Value

Status of call (0 = success)

httpEncode64_2

CUPS 1.1.21/macOS 10.4

Base64-encode a string.

```
char *httpEncode64_2(char *out, int outlen, const char *in, int inlen);
```

Parameters

out	String to write to
outlen	Maximum size of output string
in	String to read from
inlen	Size of input string

Return Value

Encoded string

httpEncryption

Set the required encryption on the link.

```
int httpEncryption(http_t *http, http_encryption_t e);
```

Parameters

http	HTTP connection
e	New encryption preference

Return Value

-1 on error, 0 on success

httpError

Get the last error on a connection.

```
int httpError(http_t *http);
```

Parameters

http	HTTP connection

Return Value

Error code (errno) value

httpFieldValue

Return the HTTP field enumeration value for a field name.

```
http_field_t httpFieldValue(const char *name);
```

Parameters

name	String name

Return Value

Field index

httpFlush

Flush data read from a HTTP connection.

```
void httpFlush(http_t *http);
```

Parameters

http	HTTP connection

httpFlushWrite

CUPS 1.2/macOS 10.5

Flush data written to a HTTP connection.

```
int httpFlushWrite(http_t *http);
```

Parameters

http	HTTP connection

Return Value

Bytes written or -1 on error

httpFreeCredentials

Free an array of credentials.

```
void httpFreeCredentials(cups_array_t *credentials);
```

Parameters

credentials	Array of credentials

httpGet

Send a GET request to the server.

```
int httpGet(http_t *http, const char *uri);
```

Parameters

http	HTTP connection
uri	URI to get

Return Value

Status of call (0 = success)

httpGetActivity

CUPS 2.0/OS 10.10

Get the most recent activity for a connection.

```
time_t httpGetActivity(http_t *http);
```

Parameters

http	HTTP connection

Return Value

Time of last read or write

Discussion

The return value is the time in seconds of the last read or write.

httpGetAddress

CUPS 2.0/OS 10.10

Get the address of the connected peer of a connection.

```
http_addr_t *httpGetAddress(http_t *http);
```

Parameters

http	HTTP connection

Return Value

Connected address or `NULL`

Discussion

For connections created with `httpConnect2`, the address is for the server. For connections created with `httpAccept`, the address is for the client.

Returns `NULL` if the socket is currently unconnected.

httpGetAuthString

CUPS 1.3/macOS 10.5

Get the current authorization string.

```
char *httpGetAuthString(http_t *http);
```

Parameters

http	HTTP connection

Return Value

Authorization string

Discussion

The authorization string is set by `cupsDoAuthentication` and `httpSetAuthString`. Use `httpGetAuthString` to retrieve the string to use with `httpSetField` for the `HTTP_FIELD_AUTHORIZATION` value.

httpGetBlocking

CUPS 1.2/macOS 10.5

Get the blocking/non-block state of a connection.

```
int httpGetBlocking(http_t *http);
```

Parameters

http	HTTP connection

Return Value

1 if blocking, 0 if non-blocking

httpGetContentEncoding

CUPS 1.7/macOS 10.9

Get a common content encoding, if any, between the client and server.

```
const char *httpGetContentEncoding(http_t *http);
```

Parameters

http	HTTP connection

Return Value

Content-Coding value or `NULL` for the identity coding.

Discussion

This function uses the value of the Accepts-Encoding HTTP header and must be called after receiving a response from the server or a request from the client. The value returned can be use in subsequent requests (for clients) or in the response (for servers) in order to compress the content stream.

httpGetCookie

CUPS 1.1.19/macOS 10.3

Get any cookie data from the response.

```
const char *httpGetCookie(http_t *http);
```

Parameters

http	HTTP connection

Return Value

Cookie data or `NULL`

httpGetDateString2

CUPS 1.2/macOS 10.5

Get a formatted date/time string from a time value.

```
const char *httpGetDateString2(time_t t, char *s, int slen);
```

Parameters

t	Time in seconds
s	String buffer
slen	Size of string buffer

Return Value

Date/time string

httpGetDateTime

Get a time value from a formatted date/time string.

```
time_t httpGetDateTime(const char *s);
```

Parameters

s	Date/time string

Return Value

Time in seconds

httpGetEncryption

CUPS 2.0/OS 10.10

Get the current encryption mode of a connection.

```
http_encryption_t httpGetEncryption(http_t *http);
```

Parameters

http	HTTP connection

Return Value

Current encryption mode

Discussion

This function returns the encryption mode for the connection. Use the `httpIsEncrypted` function to determine whether a TLS session has been established.

httpGetExpect

CUPS 1.7/macOS 10.9

Get the value of the Expect header, if any.

```
http_status_t httpGetExpect(http_t *http);
```

Parameters

http	HTTP connection

Return Value

Expect: status, if any

Discussion

Returns `HTTP_STATUS_NONE` if there is no Expect header, otherwise returns the expected HTTP status code, typically `HTTP_STATUS_CONTINUE`.

httpGetFd

CUPS 1.2/macOS 10.5

Get the file descriptor associated with a connection.

```
int httpGetFd(http_t *http);
```

Parameters

http	HTTP connection

Return Value

File descriptor or -1 if none

httpGetField

Get a field value from a request/response.

```
const char *httpGetField(http_t *http, http_field_t field);
```

Parameters

http	HTTP connection
field	Field to get

Return Value

Field value

httpGetHostname

CUPS 1.2/macOS 10.5

Get the FQDN for the connection or local system.

```
const char *httpGetHostname(http_t *http, char *s, int slen);
```

Parameters

http	HTTP connection or NULL
s	String buffer for name
slen	Size of buffer

Return Value

FQDN for connection or system

Discussion

When "http" points to a connected socket, return the hostname or address that was used in the call to httpConnect() or httpConnectEncrypt(), or the address of the client for the connection from httpAcceptConnection(). Otherwise, return the FQDN for the local system using both gethostname() and gethostbyname() to get the local hostname with domain.

httpGetKeepAlive

CUPS 2.0/OS 10.10

Get the current Keep-Alive state of the connection.

```
http_keepalive_t httpGetKeepAlive(http_t *http);
```

Parameters

http	HTTP connection

Return Value

Keep-Alive state

httpGetLength2

CUPS 1.2/macOS 10.5

Get the amount of data remaining from the content-length or transfer-encoding fields.

```
off_t httpGetLength2(http_t *http);
```

Parameters

http	HTTP connection

Return Value

Content length

Discussion

This function returns the complete content length, even for content larger than 2^31 - 1.

httpGetPending

CUPS 2.0/OS 10.10

Get the number of bytes that are buffered for writing.

```
size_t httpGetPending(http_t *http);
```

Parameters

http	HTTP connection

Return Value

Number of bytes buffered

httpGetReady

CUPS 2.0/OS 10.10

Get the number of bytes that can be read without blocking.

```
size_t httpGetReady(http_t *http);
```

Parameters

http	HTTP connection

Return Value

Number of bytes available

httpGetRemaining

CUPS 2.0/OS 10.10

Get the number of remaining bytes in the message body or current chunk.

```
size_t httpGetRemaining(http_t *http);
```

Parameters

http	HTTP connection

Return Value

Remaining bytes

Discussion

The `httpIsChunked` function can be used to determine whether the message body is chunked or fixed-length.

httpGetState

Get the current state of the HTTP request.

```
http_state_t httpGetState(http_t *http);
```

Parameters

http	HTTP connection

Return Value

HTTP state

httpGetStatus

CUPS 1.2/macOS 10.5

Get the status of the last HTTP request.

```
http_status_t httpGetStatus(http_t *http);
```

Parameters

http	HTTP connection

Return Value

HTTP status

httpGetSubField2

CUPS 1.2/macOS 10.5

Get a sub-field value.

```
char *httpGetSubField2(http_t *http, http_field_t field, const char *name, char *value, int valuelen);
```

Parameters

http	HTTP connection
field	Field index
name	Name of sub-field
value	Value string
valuelen	Size of value buffer

Return Value

Value or `NULL`

httpGetVersion

Get the HTTP version at the other end.

```
http_version_t httpGetVersion(http_t *http);
```

Parameters

http	HTTP connection

Return Value

Version number

httpGets

Get a line of text from a HTTP connection.

```
char *httpGets(char *line, int length, http_t *http);
```

Parameters

line	Line to read into
length	Max length of buffer
http	HTTP connection

Return Value

Line or `NULL`

httpHead

Send a HEAD request to the server.

```
int httpHead(http_t *http, const char *uri);
```

Parameters

http	HTTP connection
uri	URI for head

Return Value

Status of call (0 = success)

httpInitialize

Initialize the HTTP interface library and set the default HTTP proxy (if any).

```
void httpInitialize(void);
```

httpIsChunked *CUPS 2.0/OS 10.10*

Report whether a message body is chunked.

```
int httpIsChunked(http_t *http);
```

Parameters

http	HTTP connection

Return Value

1 if chunked, 0 if not

Discussion

This function returns non-zero if the message body is composed of variable-length chunks.

httpIsEncrypted

CUPS 2.0/OS 10.10

Report whether a connection is encrypted.

```
int httpIsEncrypted(http_t *http);
```

Parameters

http	HTTP connection

Return Value

1 if encrypted, 0 if not

Discussion

This function returns non-zero if the connection is currently encrypted.

httpLoadCredentials

CUPS 2.0/OS 10.10

Load X.509 credentials from a keychain file.

```
int httpLoadCredentials(const char *path, cups_array_t **credentials, const char *common_name);
```

Parameters

path	Keychain path or `NULL` for default
credentials	Credentials
common_name	Common name for credentials

Return Value

0 on success, -1 on error

httpOptions

Send an OPTIONS request to the server.

```
int httpOptions(http_t *http, const char *uri);
```

Parameters

http	HTTP connection
uri	URI for options

Return Value

Status of call (0 = success)

httpPeek

CUPS 1.7/macOS 10.9

Peek at data from a HTTP connection.

```
ssize_t httpPeek(http_t *http, char *buffer, size_t length);
```

Parameters

http	HTTP connection
buffer	Buffer for data
length	Maximum number of bytes

Return Value

Number of bytes copied

Discussion

This function copies available data from the given HTTP connection, reading a buffer as needed. The data is still available for reading using `httpRead2`.

For non-blocking connections the usual timeouts apply.

httpPost

Send a POST request to the server.

```
int httpPost(http_t *http, const char *uri);
```

Parameters

http	HTTP connection
uri	URI for post

Return Value

Status of call (0 = success)

httpPut

Send a PUT request to the server.

```
int httpPut(http_t *http, const char *uri);
```

Parameters

http	HTTP connection
uri	URI to put

Return Value

Status of call (0 = success)

httpRead2 *CUPS 1.2/macOS 10.5*

Read data from a HTTP connection.

```
ssize_t httpRead2(http_t *http, char *buffer, size_t length);
```

Parameters

http	HTTP connection
buffer	Buffer for data
length	Maximum number of bytes

Return Value

Number of bytes read

httpReadRequest

CUPS 1.7/macOS 10.9

Read a HTTP request from a connection.

```
http_state_t httpReadRequest(http_t *http, char *uri, size_t urilen);
```

Parameters

http	HTTP connection
uri	URI buffer
urilen	Size of URI buffer

Return Value

New state of connection

httpReconnect2

Reconnect to a HTTP server with timeout and optional cancel.

```
int httpReconnect2(http_t *http, int msec, int *cancel);
```

Parameters

http	HTTP connection
msec	Timeout in milliseconds
cancel	Pointer to "cancel" variable

Return Value

0 on success, non-zero on failure

httpResolveHostname

CUPS 2.0/OS 10.10

Resolve the hostname of the HTTP connection address.

```
const char *httpResolveHostname(http_t *http, char *buffer, size_t bufsize);
```

Parameters

http	HTTP connection
buffer	Hostname buffer
bufsize	Size of buffer

Return Value

Resolved hostname or `NULL`

httpSaveCredentials

CUPS 2.0/OS 10.10

Save X.509 credentials to a keychain file.

```
int httpSaveCredentials(const char *path, cups_array_t *credentials, const char *common_name);
```

Parameters

path	Keychain path or `NULL` for default
credentials	Credentials
common_name	Common name for credentials

Return Value

-1 on error, 0 on success

httpSeparateURI

CUPS 1.2/macOS 10.5

Separate a Universal Resource Identifier into its components.

```
http_uri_status_t httpSeparateURI(http_uri_coding_t decoding, const char *uri, char *scheme, int
schemelen, char *username, int usernamelen, char *host, int hostlen, int *port, char *resource, int
resourcelen);
```

Parameters

decoding	Decoding flags
uri	Universal Resource Identifier
scheme	Scheme (http, https, etc.)
schemelen	Size of scheme buffer
username	Username
usernamelen	Size of username buffer

host	Hostname
hostlen	Size of hostname buffer
port	Port number to use
resource	Resource/filename
resourcelen	Size of resource buffer

Return Value

Result of separation

httpSetAuthString

CUPS 1.3/macOS 10.5

Set the current authorization string.

```
void httpSetAuthString(http_t *http, const char *scheme, const char *data);
```

Parameters

http	HTTP connection
scheme	Auth scheme (NULL to clear it)
data	Auth data (NULL for none)

Discussion

This function just stores a copy of the current authorization string in the HTTP connection object. You must still call `httpSetField` to set `HTTP_FIELD_AUTHORIZATION` prior to issuing a HTTP request using `httpGet`, `httpHead`, `httpOptions`, `httpPost`, or `httpPut`.

httpSetCookie

CUPS 1.1.19/macOS 10.3

Set the cookie value(s).

```
void httpSetCookie(http_t *http, const char *cookie);
```

Parameters

http	Connection
cookie	Cookie string

httpSetCredentials

CUPS 1.5/macOS 10.7

Set the credentials associated with an encrypted connection.

```
int httpSetCredentials(http_t *http, cups_array_t *credentials);
```

Parameters

http	HTTP connection
credentials	Array of credentials

Return Value

Status of call (0 = success)

httpSetDefaultField

CUPS 1.7/macOS 10.9

Set the default value of an HTTP header.

```
void httpSetDefaultField(http_t *http, http_field_t field, const char *value);
```

Parameters

http	HTTP connection
field	Field index
value	Value

Discussion

Currently only `HTTP_FIELD_ACCEPT_ENCODING`, `HTTP_FIELD_SERVER`, and `HTTP_FIELD_USER_AGENT` can be set.

httpSetExpect

CUPS 1.2/macOS 10.5

Set the Expect: header in a request.

```
void httpSetExpect(http_t *http, http_status_t expect);
```

Parameters

http	HTTP connection

expect	HTTP status to expect (`HTTP_STATUS_CONTINUE`)

Discussion

Currently only `HTTP_STATUS_CONTINUE` is supported for the "expect" argument.

httpSetField

Set the value of an HTTP header.

```
void httpSetField(http_t *http, http_field_t field, const char *value);
```

Parameters

http	HTTP connection
field	Field index
value	Value

httpSetKeepAlive

CUPS 2.0/OS 10.10

Set the current Keep-Alive state of a connection.

```
void httpSetKeepAlive(http_t *http, http_keepalive_t keep_alive);
```

Parameters

http	HTTP connection
keep_alive	New Keep-Alive value

httpSetLength

CUPS 1.2/macOS 10.5

Set the content-length and content-encoding.

```
void httpSetLength(http_t *http, size_t length);
```

Parameters

http	HTTP connection
length	Length (0 for chunked)

httpSetTimeout

CUPS 1.5/macOS 10.7

Set read/write timeouts and an optional callback.

```
void httpSetTimeout(http_t *http, double timeout, http_timeout_cb_t cb, void *user_data);
```

Parameters

http	HTTP connection
timeout	Number of seconds for timeout, must be greater than 0
cb	Callback function or `NULL`
user_data	User data pointer

Discussion

The optional timeout callback receives both the HTTP connection and a user data pointer and must return 1 to continue or 0 to error (time) out.

httpShutdown

CUPS 2.0/OS 10.10

Shutdown one side of an HTTP connection.

```
void httpShutdown(http_t *http);
```

Parameters

http	HTTP connection

httpStateString

CUPS 2.0/OS 10.10

Return the string describing a HTTP state value.

```
const char *httpStateString(http_state_t state);
```

Parameters

state	HTTP state value

Return Value

State string

httpStatus

Return a short string describing a HTTP status code.

```
const char *httpStatus(http_status_t status);
```

Parameters

status	HTTP status code

Return Value

Localized status string

Discussion

The returned string is localized to the current POSIX locale and is based on the status strings defined in RFC 7231.

httpURIStatusString

CUPS 2.0/OS 10.10

Return a string describing a URI status code.

```
const char *httpURIStatusString(http_uri_status_t status);
```

Parameters

status	URI status code

Return Value

Localized status string

httpUpdate

Update the current HTTP state for incoming data.

```
http_status_t httpUpdate(http_t *http);
```

Parameters

http	HTTP connection

Return Value

HTTP status

httpWait

CUPS 1.1.19/macOS 10.3

Wait for data available on a connection.

```
int httpWait(http_t *http, int msec);
```

Parameters

http	HTTP connection
msec	Milliseconds to wait

Return Value

1 if data is available, 0 otherwise

httpWrite2

CUPS 1.2/macOS 10.5

Write data to a HTTP connection.

```
ssize_t httpWrite2(http_t *http, const char *buffer, size_t length);
```

Parameters

http	HTTP connection
buffer	Buffer for data
length	Number of bytes to write

Return Value

Number of bytes written

httpWriteResponse

CUPS 1.7/macOS 10.9

Write a HTTP response to a client connection.

```
int httpWriteResponse(http_t *http, http_status_t status);
```

Parameters

http	HTTP connection
status	Status code

Return Value

0 on success, -1 on error

ippAddBoolean

Add a boolean attribute to an IPP message.

```
ipp_attribute_t *ippAddBoolean(ipp_t *ipp, ipp_tag_t group, const char *name, char value);
```

Parameters

ipp	IPP message
group	IPP group
name	Name of attribute
value	Value of attribute

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO`, for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

ippAddBooleans

Add an array of boolean values.

```
ipp_attribute_t *ippAddBooleans(ipp_t *ipp, ipp_tag_t group, const char *name, int num_values, const
char *values);
```

Parameters

ipp	IPP message
group	IPP group
name	Name of attribute
num_values	Number of values
values	Values

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO`, for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

ippAddCollection

CUPS 1.1.19/macOS 10.3

Add a collection value.

```
ipp_attribute_t *ippAddCollection(ipp_t *ipp, ipp_tag_t group, const char *name, ipp_t *value);
```

Parameters

ipp	IPP message
group	IPP group
name	Name of attribute
value	Value

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO` , for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

ippAddCollections

CUPS 1.1.19/macOS 10.3

Add an array of collection values.

```
ipp_attribute_t *ippAddCollections(ipp_t *ipp, ipp_tag_t group, const char *name, int num_values,
const ipp_t **values);
```

Parameters

ipp	IPP message
group	IPP group
name	Name of attribute
num_values	Number of values
values	Values

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew` , `ippNewRequest` , or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO` , for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

ippAddDate

Add a dateTime attribute to an IPP message.

```
ipp_attribute_t *ippAddDate(ipp_t *ipp, ipp_tag_t group, const char *name, const ipp_uchar_t *value);
```

Parameters

ipp	IPP message
group	IPP group
name	Name of attribute
value	Value

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO`, for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

ippAddInteger

Add a integer attribute to an IPP message.

```
ipp_attribute_t *ippAddInteger(ipp_t *ipp, ipp_tag_t group, ipp_tag_t value_tag, const char *name, int
value);
```

Parameters

ipp	IPP message
group	IPP group
value_tag	Type of attribute
name	Name of attribute
value	Value of attribute

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO` , for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

Supported values include enum (`IPP_TAG_ENUM`) and integer (`IPP_TAG_INTEGER`).

ippAddIntegers

Add an array of integer values.

```
ipp_attribute_t *ippAddIntegers(ipp_t *ipp, ipp_tag_t group, ipp_tag_t value_tag, const char *name,
int num_values, const int *values);
```

Parameters

ipp	IPP message
group	IPP group
value_tag	Type of attribute
name	Name of attribute
num_values	Number of values
values	Values

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew` , `ippNewRequest` , or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO` , for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

Supported values include enum (`IPP_TAG_ENUM`) and integer (`IPP_TAG_INTEGER`).

ippAddOctetString

CUPS 1.2/macOS 10.5

Add an octetString value to an IPP message.

```
ipp_attribute_t *ippAddOctetString(ipp_t *ipp, ipp_tag_t group, const char *name, const void *data,
int datalen);
```

Parameters

ipp	IPP message
group	IPP group
name	Name of attribute
data	octetString data
datalen	Length of data in bytes

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO`, for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

ippAddOutOfBand

CUPS 1.6/macOS 10.8

Add an out-of-band value to an IPP message.

```
ipp_attribute_t *ippAddOutOfBand(ipp_t *ipp, ipp_tag_t group, ipp_tag_t value_tag, const char *name);
```

Parameters

ipp	IPP message
group	IPP group
value_tag	Type of attribute
name	Name of attribute

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO`, for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

Supported out-of-band values include unsupported-value (`IPP_TAG_UNSUPPORTED_VALUE`), default (`IPP_TAG_DEFAULT`), unknown (`IPP_TAG_UNKNOWN`), no-value (`IPP_TAG_NOVALUE`), not-settable (`IPP_TAG_NOTSETTABLE`), delete-attribute (`IPP_TAG_DELETEATTR`), and admin-define (`IPP_TAG_ADMINDEFINE`).

ippAddRange

Add a range of values to an IPP message.

```
ipp_attribute_t *ippAddRange(ipp_t *ipp, ipp_tag_t group, const char *name, int lower, int upper);
```

Parameters

ipp	IPP message
group	IPP group
name	Name of attribute
lower	Lower value
upper	Upper value

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO`, for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or

unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

The `lower` parameter must be less than or equal to the `upper` parameter.

ippAddRanges

Add ranges of values to an IPP message.

```
ipp_attribute_t *ippAddRanges(ipp_t *ipp, ipp_tag_t group, const char *name, int num_values, const int
*lower, const int *upper);
```

Parameters

ipp	IPP message
group	IPP group
name	Name of attribute
num_values	Number of values
lower	Lower values
upper	Upper values

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO` , for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

ippAddResolution

Add a resolution value to an IPP message.

```
ipp_attribute_t *ippAddResolution(ipp_t *ipp, ipp_tag_t group, const char *name, ipp_res_t units, int
xres, int yres);
```

Parameters

ipp	IPP message
group	IPP group
name	Name of attribute
units	Units for resolution
xres	X resolution
yres	Y resolution

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO`, for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

ippAddResolutions

Add resolution values to an IPP message.

```
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);
```

Parameters

ipp	IPP message
group	IPP group
name	Name of attribute
num_values	Number of values
units	Units for resolution
xres	X resolutions
yres	Y resolutions

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO`, for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

ippAddSeparator

Add a group separator to an IPP message.

```
ipp_attribute_t *ippAddSeparator(ipp_t *ipp);
```

Parameters

ipp	IPP message

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

ippAddString

Add a language-encoded string to an IPP message.

```
ipp_attribute_t *ippAddString(ipp_t *ipp, ipp_tag_t group, ipp_tag_t value_tag, const char *name,
const char *language, const char *value);
```

Parameters

ipp	IPP message
group	IPP group
value_tag	Type of attribute

name	Name of attribute
language	Language code
value	Value

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO` , for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

Supported string values include charset (`IPP_TAG_CHARSET`), keyword (`IPP_TAG_KEYWORD`), language (`IPP_TAG_LANGUAGE`), mimeMediaType (`IPP_TAG_MIMETYPE`), name (`IPP_TAG_NAME`), nameWithLanguage (`IPP_TAG_NAMELANG), text (` code IPP_TAG_TEXT@), textWithLanguage (`IPP_TAG_TEXTLANG`), uri (`IPP_TAG_URI`), and uriScheme (`IPP_TAG_URISCHEME`).

The `language` parameter must be non- `NULL` for nameWithLanguage and textWithLanguage string values and must be `NULL` for all other string values.

ippAddStringf

CUPS 1.7/macOS 10.9

Add a formatted string to an IPP message.

```
ipp_attribute_t *ippAddStringf(ipp_t *ipp, ipp_tag_t group, ipp_tag_t value_tag, const char *name,
const char *language, const char *format, ...);
```

Parameters

ipp	IPP message
group	IPP group
value_tag	Type of attribute
name	Name of attribute
language	Language code (`NULL` for default)
format	Printf-style format string

...	Additional arguments as needed

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO`, for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

Supported string values include charset (`IPP_TAG_CHARSET`), keyword (`IPP_TAG_KEYWORD`), language (`IPP_TAG_LANGUAGE`), mimeMediaType (`IPP_TAG_MIMETYPE`), name (`IPP_TAG_NAME`), nameWithLanguage (`IPP_TAG_NAMELANG), text (` code IPP_TAG_TEXT@), textWithLanguage (`IPP_TAG_TEXTLANG`), uri (`IPP_TAG_URI`), and uriScheme (`IPP_TAG_URISCHEME`).

The `language` parameter must be non-`NULL` for nameWithLanguage and textWithLanguage string values and must be `NULL` for all other string values.

The `format` parameter uses formatting characters compatible with the printf family of standard functions. Additional arguments follow it as needed. The formatted string is truncated as needed to the maximum length of the corresponding value type.

ippAddStringfv

CUPS 1.7/macOS 10.9

Add a formatted string to an IPP message.

```
ipp_attribute_t *ippAddStringfv(ipp_t *ipp, ipp_tag_t group, ipp_tag_t value_tag, const char *name,
const char *language, const char *format, va_list ap);
```

Parameters

ipp	IPP message
group	IPP group
value_tag	Type of attribute
name	Name of attribute
language	Language code (`NULL` for default)
format	Printf-style format string

ap	Additional arguments

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO`, for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

Supported string values include charset (`IPP_TAG_CHARSET`), keyword (`IPP_TAG_KEYWORD`), language (`IPP_TAG_LANGUAGE`), mimeMediaType (`IPP_TAG_MIMETYPE`), name (`IPP_TAG_NAME`), nameWithLanguage (`IPP_TAG_NAMELANG), text (` code IPP_TAG_TEXT@), textWithLanguage (`IPP_TAG_TEXTLANG`), uri (`IPP_TAG_URI`), and uriScheme (`IPP_TAG_URISCHEME`).

The `language` parameter must be non-`NULL` for nameWithLanguage and textWithLanguage string values and must be `NULL` for all other string values.

The `format` parameter uses formatting characters compatible with the printf family of standard functions. Additional arguments are passed in the stdarg pointer `ap`. The formatted string is truncated as needed to the maximum length of the corresponding value type.

ippAddStrings

Add language-encoded strings to an IPP message.

```
ipp_attribute_t *ippAddStrings(ipp_t *ipp, ipp_tag_t group, ipp_tag_t value_tag, const char *name, int
num_values, const char *language, const char *const *values);
```

Parameters

ipp	IPP message
group	IPP group
value_tag	Type of attribute
name	Name of attribute
num_values	Number of values

language	Language code (`NULL` for default)
values	Values

Return Value

New attribute

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO`, for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

Supported string values include charset (`IPP_TAG_CHARSET`), keyword (`IPP_TAG_KEYWORD`), language (`IPP_TAG_LANGUAGE`), mimeMediaType (`IPP_TAG_MIMETYPE`), name (`IPP_TAG_NAME`), nameWithLanguage (`IPP_TAG_NAMELANG), text (` code IPP_TAG_TEXT@), textWithLanguage (`IPP_TAG_TEXTLANG`), uri (`IPP_TAG_URI`), and uriScheme (`IPP_TAG_URISCHEME`).

The `language` parameter must be non-`NULL` for nameWithLanguage and textWithLanguage string values and must be `NULL` for all other string values.

ippAttributeString

CUPS 1.6/macOS 10.8

Convert the attribute's value to a string.

```
size_t ippAttributeString(ipp_attribute_t *attr, char *buffer, size_t bufsize);
```

Parameters

attr	Attribute
buffer	String buffer or NULL
bufsize	Size of string buffer

Return Value

Number of bytes less nul

Discussion

Returns the number of bytes that would be written, not including the trailing nul. The

buffer pointer can be NULL to get the required length, just like (v)snprintf.

ippContainsInteger

CUPS 1.7/macOS 10.9

Determine whether an attribute contains the specified value or is within the list of ranges.

```
int ippContainsInteger(ipp_attribute_t *attr, int value);
```

Parameters

attr	Attribute
value	Integer/enum value

Return Value

1 on a match, 0 on no match

Discussion

Returns non-zero when the attribute contains either a matching integer or enum value, or the value falls within one of the rangeOfInteger values for the attribute.

ippContainsString

CUPS 1.7/macOS 10.9

Determine whether an attribute contains the specified string value.

```
int ippContainsString(ipp_attribute_t *attr, const char *value);
```

Parameters

attr	Attribute
value	String value

Return Value

1 on a match, 0 on no match

Discussion

Returns non-zero when the attribute contains a matching charset, keyword, naturalLanguage, mimeMediaType, name, text, uri, or uriScheme value.

ippCopyAttribute

CUPS 1.6/macOS 10.8

Copy an attribute.

```
ipp_attribute_t *ippCopyAttribute(ipp_t *dst, ipp_attribute_t *srcattr, int quickcopy);
```

Parameters

dst	Destination IPP message
srcattr	Attribute to copy
quickcopy	1 for a referenced copy, 0 for normal

Return Value

New attribute

Discussion

The specified attribute, `attr`, is copied to the destination IPP message. When `quickcopy` is non-zero, a "shallow" reference copy of the attribute is created - this should only be done as long as the original source IPP message will not be freed for the life of the destination.

ippCopyAttributes

CUPS 1.6/macOS 10.8

Copy attributes from one IPP message to another.

```
int ippCopyAttributes(ipp_t *dst, ipp_t *src, int quickcopy, ipp_copycb_t cb, void *context);
```

Parameters

dst	Destination IPP message
src	Source IPP message
quickcopy	1 for a referenced copy, 0 for normal
cb	Copy callback or `NULL` for none
context	Context pointer

Return Value

1 on success, 0 on error

Discussion

Zero or more attributes are copied from the source IPP message, `src`, to the destination IPP message, `dst`. When `quickcopy` is non-zero, a "shallow" reference copy of the attribute is created - this should only be done as long as the original source IPP message will not be freed for the life of the destination.

The `cb` and `context` parameters provide a generic way to "filter" the attributes that are copied - the function must return 1 to copy the attribute or 0 to skip it. The function may also choose to do a partial copy of the source attribute itself.

ippCreateRequestedArray

CUPS 1.7/macOS 10.9

Create a CUPS array of attribute names from the given requested-attributes attribute.

```
cups_array_t *ippCreateRequestedArray(ipp_t *request);
```

Parameters

request	IPP request

Return Value

CUPS array or `NULL` if all

Discussion

This function creates a (sorted) CUPS array of attribute names matching the list of "requested-attribute" values supplied in an IPP request. All IANA- registered values are supported in addition to the CUPS IPP extension attributes.

The `request` parameter specifies the request message that was read from the client. `NULL` is returned if all attributes should be returned. Otherwise, the result is a sorted array of attribute names, where `cupsArrayFind(array, "attribute-name")` will return a non-NULL pointer. The array must be freed using the `cupsArrayDelete` function.

ippDateToTime

Convert from RFC 2579 Date/Time format to time in seconds.

```
time_t ippDateToTime(const ipp_uchar_t *date);
```

Parameters

date	RFC 2579 date info

Return Value

UNIX time value

ippDelete

Delete an IPP message.

```
void ippDelete(ipp_t *ipp);
```

Parameters

ipp	IPP message

ippDeleteAttribute *CUPS 1.1.19/macOS 10.3*

Delete a single attribute in an IPP message.

```
void ippDeleteAttribute(ipp_t *ipp, ipp_attribute_t *attr);
```

Parameters

ipp	IPP message
attr	Attribute to delete

ippDeleteValues *CUPS 1.6/macOS 10.8*

Delete values in an attribute.

```
int ippDeleteValues(ipp_t *ipp, ipp_attribute_t **attr, int element, int count);
```

Parameters

ipp	IPP message
attr	Attribute
element	Index of first value to delete (0-based)
count	Number of values to delete

Return Value

1 on success, 0 on failure

Discussion

The `element` parameter specifies the first value to delete, starting at 0. It must be less than the number of values returned by `ippGetCount`.

The `attr` parameter may be modified as a result of setting the value.

Deleting all values in an attribute deletes the attribute.

ippEnumString

Return a string corresponding to the enum value.

```
const char *ippEnumString(const char *attrname, int enumvalue);
```

Parameters

attrname	Attribute name
enumvalue	Enum value

Return Value

Enum string

ippEnumValue

Return the value associated with a given enum string.

```
int ippEnumValue(const char *attrname, const char *enumstring);
```

Parameters

attrname	Attribute name
enumstring	Enum string

Return Value

Enum value or -1 if unknown

ippErrorString

Return a name for the given status code.

```
const char *ippErrorString(ipp_status_t error);
```

Parameters

error	Error status

Return Value

Text string

ippErrorValue

CUPS 1.2/macOS 10.5

Return a status code for the given name.

```
ipp_status_t ippErrorValue(const char *name);
```

Parameters

name	Name

Return Value

IPP status code

ippFindAttribute

Find a named attribute in a request.

```
ipp_attribute_t *ippFindAttribute(ipp_t *ipp, const char *name, ipp_tag_t type);
```

Parameters

ipp	IPP message
name	Name of attribute
type	Type of attribute

Return Value

Matching attribute

Discussion

Starting with CUPS 2.0, the attribute name can contain a hierarchical list of attribute and

member names separated by slashes, for example "media-col/media-size".

ippFindNextAttribute

Find the next named attribute in a request.

```
ipp_attribute_t *ippFindNextAttribute(ipp_t *ipp, const char *name, ipp_tag_t type);
```

Parameters

ipp	IPP message
name	Name of attribute
type	Type of attribute

Return Value

Matching attribute

Discussion

Starting with CUPS 2.0, the attribute name can contain a hierarchical list of attribute and member names separated by slashes, for example "media-col/media-size".

ippFirstAttribute

CUPS 1.6/macOS 10.8

Return the first attribute in the message.

```
ipp_attribute_t *ippFirstAttribute(ipp_t *ipp);
```

Parameters

ipp	IPP message

Return Value

First attribute or `NULL` if none

ippGetBoolean

CUPS 1.6/macOS 10.8

Get a boolean value for an attribute.

```
int ippGetBoolean(ipp_attribute_t *attr, int element);
```

Parameters

attr	IPP attribute
element	Value number (0-based)

Return Value

Boolean value or 0 on error

Discussion

The `element` parameter specifies which value to get from 0 to `ippGetCount(attr)` - 1.

ippGetCollection

CUPS 1.6/macOS 10.8

Get a collection value for an attribute.

```
ipp_t *ippGetCollection(ipp_attribute_t *attr, int element);
```

Parameters

attr	IPP attribute
element	Value number (0-based)

Return Value

Collection value or `NULL` on error

Discussion

The `element` parameter specifies which value to get from 0 to `ippGetCount(attr)` - 1.

ippGetCount

CUPS 1.6/macOS 10.8

Get the number of values in an attribute.

```
int ippGetCount(ipp_attribute_t *attr);
```

Parameters

attr	IPP attribute

Return Value

Number of values or 0 on error

ippGetDate

CUPS 1.6/macOS 10.8

Get a dateTime value for an attribute.

```
const ipp_uchar_t *ippGetDate(ipp_attribute_t *attr, int element);
```

Parameters

attr	IPP attribute
element	Value number (0-based)

Return Value

dateTime value or `NULL`

Discussion

The `element` parameter specifies which value to get from 0 to `ippGetCount(attr)` - 1.

ippGetGroupTag

CUPS 1.6/macOS 10.8

Get the group associated with an attribute.

```
ipp_tag_t ippGetGroupTag(ipp_attribute_t *attr);
```

Parameters

attr	IPP attribute

Return Value

Group tag or `IPP_TAG_ZERO` on error

ippGetInteger

CUPS 1.6/macOS 10.8

Get the integer/enum value for an attribute.

```
int ippGetInteger(ipp_attribute_t *attr, int element);
```

Parameters

attr	IPP attribute
element	Value number (0-based)

Return Value

Value or 0 on error

Discussion

The `element` parameter specifies which value to get from 0 to `ippGetCount(attr)` - 1.

ippGetName

CUPS 1.6/macOS 10.8

Get the attribute name.

```
const char *ippGetName(ipp_attribute_t *attr);
```

Parameters

attr	IPP attribute

Return Value

Attribute name or `NULL` for separators

ippGetOctetString

CUPS 1.7/macOS 10.9

Get an octetString value from an IPP attribute.

```
void *ippGetOctetString(ipp_attribute_t *attr, int element, int *datalen);
```

Parameters

attr	IPP attribute
element	Value number (0-based)
datalen	Length of octetString data

Return Value

Pointer to octetString data

Discussion

The `element` parameter specifies which value to get from 0 to `ippGetCount(attr)` - 1.

ippGetOperation

CUPS 1.6/macOS 10.8

Get the operation ID in an IPP message.

```
ipp_op_t ippGetOperation(ipp_t *ipp);
```

Parameters

ipp	IPP request message

Return Value

Operation ID or 0 on error

ippGetRange

CUPS 1.6/macOS 10.8

Get a rangeOfInteger value from an attribute.

```
int ippGetRange(ipp_attribute_t *attr, int element, int *uppervalue);
```

Parameters

attr	IPP attribute
element	Value number (0-based)
uppervalue	Upper value of range

Return Value

Lower value of range or 0

Discussion

The `element` parameter specifies which value to get from 0 to `ippGetCount(attr)` - 1.

ippGetRequestId

CUPS 1.6/macOS 10.8

Get the request ID from an IPP message.

```
int ippGetRequestId(ipp_t *ipp);
```

Parameters

ipp	IPP message

Return Value

Request ID or 0 on error

ippGetResolution

CUPS 1.6/macOS 10.8

Get a resolution value for an attribute.

```
int ippGetResolution(ipp_attribute_t *attr, int element, int *yres, ipp_res_t *units);
```

Parameters

attr	IPP attribute
element	Value number (0-based)
yres	Vertical/feed resolution
units	Units for resolution

Return Value

Horizontal/cross feed resolution or 0

Discussion

The `element` parameter specifies which value to get from 0 to `ippGetCount(attr)` - 1.

ippGetState

CUPS 1.6/macOS 10.8

Get the IPP message state.

```
ipp_state_t ippGetState(ipp_t *ipp);
```

Parameters

ipp	IPP message

Return Value

IPP message state value

ippGetStatusCode

CUPS 1.6/macOS 10.8

Get the status code from an IPP response or event message.

```
ipp_status_t ippGetStatusCode(ipp_t *ipp);
```

Parameters

ipp	IPP response or event message

Return Value

Status code in IPP message

ippGetString

```
const char *ippGetString(ipp_attribute_t *attr, int element, const char **language);
```

Parameters

attr	IPP attribute
element	Value number (0-based)
language	Language code (`NULL` for don't care)

Return Value

Get the string and optionally the language code for an attribute.

The `element` parameter specifies which value to get from 0 to `ippGetCount(attr)` - 1.

ippGetValueTag

CUPS 1.6/macOS 10.8

Get the value tag for an attribute.

```
ipp_tag_t ippGetValueTag(ipp_attribute_t *attr);
```

Parameters

attr	IPP attribute

Return Value

Value tag or `IPP_TAG_ZERO` on error

ippGetVersion

CUPS 1.6/macOS 10.8

Get the major and minor version number from an IPP message.

```
int ippGetVersion(ipp_t *ipp, int *minor);
```

Parameters

ipp	IPP message
minor	Minor version number or `NULL` for don't care

Return Value

Major version number or 0 on error

ippLength

Compute the length of an IPP message.

```
size_t ippLength(ipp_t *ipp);
```

Parameters

ipp	IPP message

Return Value

Size of IPP message

ippNew

Allocate a new IPP message.

```
ipp_t *ippNew(void);
```

Return Value

New IPP message

ippNewRequest

CUPS 1.2/macOS 10.5

Allocate a new IPP request message.

```
ipp_t *ippNewRequest(ipp_op_t op);
```

Parameters

op	Operation code

Return Value

IPP request message

Discussion

The new request message is initialized with the "attributes-charset" and "attributes-natural-language" attributes added. The "attributes-natural-language" value is derived from the current locale.

ippNewResponse

CUPS 1.7/macOS 10.9

Allocate a new IPP response message.

```
ipp_t *ippNewResponse(ipp_t *request);
```

Parameters

request	IPP request message

Return Value

IPP response message

Discussion

The new response message is initialized with the same "version-number", "request-id", "attributes-charset", and "attributes-natural-language" as the provided request message. If the "attributes-charset" or "attributes-natural-language" attributes are missing from the request, 'utf-8' and a value derived from the current locale are substituted, respectively.

ippNextAttribute

CUPS 1.6/macOS 10.8

Return the next attribute in the message.

```
ipp_attribute_t *ippNextAttribute(ipp_t *ipp);
```

Parameters

ipp	IPP message

Return Value

Next attribute or `NULL` if none

ippOpString *CUPS 1.2/macOS 10.5*

Return a name for the given operation id.

```
const char *ippOpString(ipp_op_t op);
```

Parameters

op	Operation ID

Return Value

Name

ippOpValue *CUPS 1.2/macOS 10.5*

Return an operation id for the given name.

```
ipp_op_t ippOpValue(const char *name);
```

Parameters

name	Textual name

Return Value

Operation ID

ippPort

Return the default IPP port number.

```
int ippPort(void);
```

Return Value

Port number

ippRead

Read data for an IPP message from a HTTP connection.

```
ipp_state_t ippRead(http_t *http, ipp_t *ipp);
```

Parameters

http	HTTP connection
ipp	IPP data

Return Value

Current state

ippReadFile

CUPS 1.1.19/macOS 10.3

Read data for an IPP message from a file.

```
ipp_state_t ippReadFile(int fd, ipp_t *ipp);
```

Parameters

fd	HTTP data
ipp	IPP data

Return Value

Current state

ippReadIO

CUPS 1.2/macOS 10.5

Read data for an IPP message.

```
ipp_state_t ippReadIO(void *src, ipp_iocb_t cb, int blocking, ipp_t *parent, ipp_t *ipp);
```

Parameters

src	Data source
cb	Read callback function

blocking	Use blocking IO?
parent	Parent request, if any
ipp	IPP data

Return Value

Current state

ippSetBoolean

CUPS 1.6/macOS 10.8

Set a boolean value in an attribute.

```
int ippSetBoolean(ipp_t *ipp, ipp_attribute_t **attr, int element, int boolvalue);
```

Parameters

ipp	IPP message
attr	IPP attribute
element	Value number (0-based)
boolvalue	Boolean value

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `attr` parameter may be modified as a result of setting the value.

The `element` parameter specifies which value to set from 0 to `ippGetCount(attr)`.

ippSetCollection

CUPS 1.6/macOS 10.8

Set a collection value in an attribute.

```
int ippSetCollection(ipp_t *ipp, ipp_attribute_t **attr, int element, ipp_t *colvalue);
```

Parameters

ipp	IPP message
attr	IPP attribute
element	Value number (0-based)
colvalue	Collection value

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `attr` parameter may be modified as a result of setting the value.

The `element` parameter specifies which value to set from 0 to `ippGetCount(attr)`.

ippSetDate

CUPS 1.6/macOS 10.8

Set a dateTime value in an attribute.

```
int ippSetDate(ipp_t *ipp, ipp_attribute_t **attr, int element, const ipp_uchar_t *datevalue);
```

Parameters

ipp	IPP message
attr	IPP attribute
element	Value number (0-based)
datevalue	dateTime value

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `attr` parameter may be modified as a result of setting the value.

The `element` parameter specifies which value to set from 0 to `ippGetCount(attr)`.

ippSetGroupTag

CUPS 1.6/macOS 10.8

Set the group tag of an attribute.

```
int ippSetGroupTag(ipp_t *ipp, ipp_attribute_t **attr, ipp_tag_t group_tag);
```

Parameters

ipp	IPP message
attr	Attribute
group_tag	Group tag

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `attr` parameter may be modified as a result of setting the value.

The `group` parameter specifies the IPP attribute group tag: none (`IPP_TAG_ZERO`, for member attributes), document (`IPP_TAG_DOCUMENT`), event notification (`IPP_TAG_EVENT_NOTIFICATION`), operation (`IPP_TAG_OPERATION`), printer (`IPP_TAG_PRINTER`), subscription (`IPP_TAG_SUBSCRIPTION`), or unsupported (`IPP_TAG_UNSUPPORTED_GROUP`).

ippSetInteger

CUPS 1.6/macOS 10.8

Set an integer or enum value in an attribute.

```
int ippSetInteger(ipp_t *ipp, ipp_attribute_t **attr, int element, int intvalue);
```

Parameters

ipp	IPP message
attr	IPP attribute
element	Value number (0-based)
intvalue	Integer/enum value

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `attr` parameter may be modified as a result of setting the value.

The `element` parameter specifies which value to set from 0 to `ippGetCount(attr)`.

ippSetName

CUPS 1.6/macOS 10.8

Set the name of an attribute.

```
int ippSetName(ipp_t *ipp, ipp_attribute_t **attr, const char *name);
```

Parameters

ipp	IPP message
attr	IPP attribute
name	Attribute name

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `attr` parameter may be modified as a result of setting the value.

ippSetOctetString

CUPS 1.7/macOS 10.9

Set an octetString value in an IPP attribute.

```
int ippSetOctetString(ipp_t *ipp, ipp_attribute_t **attr, int element, const void *data, int datalen);
```

Parameters

ipp	IPP message
attr	IPP attribute
element	Value number (0-based)
data	Pointer to octetString data
datalen	Length of octetString data

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `attr` parameter may be modified as a result of setting the value.

The `element` parameter specifies which value to set from 0 to `ippGetCount(attr)`.

ippSetOperation

CUPS 1.6/macOS 10.8

Set the operation ID in an IPP request message.

```
int ippSetOperation(ipp_t *ipp, ipp_op_t op);
```

Parameters

ipp	IPP request message
op	Operation ID

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

ippSetPort

Set the default port number.

```
void ippSetPort(int p);
```

Parameters

p	Port number to use

ippSetRange

CUPS 1.6/macOS 10.8

Set a rangeOfInteger value in an attribute.

```
int ippSetRange(ipp_t *ipp, ipp_attribute_t **attr, int element, int lowervalue, int uppervalue);
```

Parameters

ipp	IPP message
attr	IPP attribute
element	Value number (0-based)
lowervalue	Lower bound for range
uppervalue	Upper bound for range

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `attr` parameter may be modified as a result of setting the value.

The `element` parameter specifies which value to set from 0 to `ippGetCount(attr)`.

ippSetRequestId

CUPS 1.6/macOS 10.8

Set the request ID in an IPP message.

```
int ippSetRequestId(ipp_t *ipp, int request_id);
```

Parameters

ipp	IPP message
request_id	Request ID

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `request_id` parameter must be greater than 0.

ippSetResolution

CUPS 1.6/macOS 10.8

Set a resolution value in an attribute.

```
int ippSetResolution(ipp_t *ipp, ipp_attribute_t **attr, int element, ipp_res_t unitsvalue, int
xresvalue, int yresvalue);
```

Parameters

ipp	IPP message
attr	IPP attribute
element	Value number (0-based)
unitsvalue	Resolution units
xresvalue	Horizontal/cross feed resolution
yresvalue	Vertical/feed resolution

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `attr` parameter may be modified as a result of setting the value.

The `element` parameter specifies which value to set from 0 to `ippGetCount(attr)`.

ippSetState

CUPS 1.6/macOS 10.8

Set the current state of the IPP message.

```
int ippSetState(ipp_t *ipp, ipp_state_t state);
```

Parameters

ipp	IPP message
state	IPP state value

Return Value

1 on success, 0 on failure

ippSetStatusCode

CUPS 1.6/macOS 10.8

Set the status code in an IPP response or event message.

```
int ippSetStatusCode(ipp_t *ipp, ipp_status_t status);
```

Parameters

ipp	IPP response or event message
status	Status code

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

ippSetString

CUPS 1.6/macOS 10.8

Set a string value in an attribute.

```
int ippSetString(ipp_t *ipp, ipp_attribute_t **attr, int element, const char *strvalue);
```

Parameters

ipp	IPP message
attr	IPP attribute
element	Value number (0-based)
strvalue	String value

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `attr` parameter may be modified as a result of setting the value.

The `element` parameter specifies which value to set from 0 to `ippGetCount(attr)`.

ippSetStringf

CUPS 1.7/macOS 10.9

Set a formatted string value of an attribute.

```
int ippSetStringf(ipp_t *ipp, ipp_attribute_t **attr, int element, const char *format, ...);
```

Parameters

ipp	IPP message
attr	IPP attribute
element	Value number (0-based)
format	Printf-style format string
...	Additional arguments as needed

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `attr` parameter may be modified as a result of setting the value.

The `element` parameter specifies which value to set from 0 to `ippGetCount(attr)`.

The `format` parameter uses formatting characters compatible with the printf family of standard functions. Additional arguments follow it as needed. The formatted string is truncated as needed to the maximum length of the corresponding value type.

ippSetStringfv

CUPS 1.7/macOS 10.9

Set a formatted string value of an attribute.

```
int ippSetStringfv(ipp_t *ipp, ipp_attribute_t **attr, int element, const char *format, va_list ap);
```

Parameters

ipp	IPP message
attr	IPP attribute
element	Value number (0-based)
format	Printf-style format string
ap	Pointer to additional arguments

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `attr` parameter may be modified as a result of setting the value.

The `element` parameter specifies which value to set from 0 to `ippGetCount(attr)`.

The `format` parameter uses formatting characters compatible with the printf family of standard functions. Additional arguments follow it as needed. The formatted string is truncated as needed to the maximum length of the corresponding value type.

ippSetValueTag

CUPS 1.6/macOS 10.8

Set the value tag of an attribute.

```
int ippSetValueTag(ipp_t *ipp, ipp_attribute_t **attr, ipp_tag_t value_tag);
```

Parameters

ipp	IPP message
attr	IPP attribute
value_tag	Value tag

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The `attr` parameter may be modified as a result of setting the value.

Integer (`IPP_TAG_INTEGER`) values can be promoted to rangeOfInteger (`IPP_TAG_RANGE`) values, the various string tags can be promoted to name (`IPP_TAG_NAME`) or nameWithLanguage (`IPP_TAG_NAMELANG`) values, text (`IPP_TAG_TEXT`) values can be promoted to textWithLanguage (`IPP_TAG_TEXTLANG`) values, and all values can be demoted to the various out-of-band value tags such as no-value (`IPP_TAG_NOVALUE`). All other changes will be rejected.

Promoting a string attribute to nameWithLanguage or textWithLanguage adds the language code in the "attributes-natural-language" attribute or, if not present, the language code for the current locale.

ippSetVersion

CUPS 1.6/macOS 10.8

Set the version number in an IPP message.

```
int ippSetVersion(ipp_t *ipp, int major, int minor);
```

Parameters

ipp	IPP message
major	Major version number (major.minor)
minor	Minor version number (major.minor)

Return Value

1 on success, 0 on failure

Discussion

The `ipp` parameter refers to an IPP message previously created using the `ippNew`, `ippNewRequest`, or `ippNewResponse` functions.

The valid version numbers are currently 1.0, 1.1, 2.0, 2.1, and 2.2.

ippStateString

CUPS 2.0/OS 10.10

Return the name corresponding to a state value.

```
const char *ippStateString(ipp_state_t state);
```

Parameters

state	State value

Return Value

State name

ippTagString

CUPS 1.4/macOS 10.6

Return the tag name corresponding to a tag value.

```
const char *ippTagString(ipp_tag_t tag);
```

Parameters

tag	Tag value

Return Value

Tag name

Discussion

The returned names are defined in RFC 8011 and the IANA IPP Registry.

ippTagValue

CUPS 1.4/macOS 10.6

Return the tag value corresponding to a tag name.

```
ipp_tag_t ippTagValue(const char *name);
```

Parameters

name	Tag name

Return Value

Tag value

Discussion

The tag names are defined in RFC 8011 and the IANA IPP Registry.

ippTimeToDate

Convert from time in seconds to RFC 2579 format.

```
const ipp_uchar_t *ippTimeToDate(time_t t);
```

Parameters

t	Time in seconds

Return Value

RFC-2579 date/time data

ippValidateAttribute *CUPS 1.7/macOS 10.9*

Validate the contents of an attribute.

```
int ippValidateAttribute(ipp_attribute_t *attr);
```

Parameters

attr	Attribute

Return Value

1 if valid, 0 otherwise

Discussion

This function validates the contents of an attribute based on the name and value tag. 1 is returned if the attribute is valid, 0 otherwise. On failure, `cupsLastErrorString` is set to a human-readable message.

ippValidateAttributes

CUPS 1.7/macOS 10.9

Validate all attributes in an IPP message.

```
int ippValidateAttributes(ipp_t *ipp);
```

Parameters

ipp	IPP message

Return Value

1 if valid, 0 otherwise

Discussion

This function validates the contents of the IPP message, including each attribute. Like `ippValidateAttribute`, `cupsLastErrorString` is set to a human-readable message on failure.

ippWrite

Write data for an IPP message to a HTTP connection.

```
ipp_state_t ippWrite(http_t *http, ipp_t *ipp);
```

Parameters

http	HTTP connection
ipp	IPP data

Return Value

Current state

ippWriteFile

CUPS 1.1.19/macOS 10.3

Write data for an IPP message to a file.

```
ipp_state_t ippWriteFile(int fd, ipp_t *ipp);
```

Parameters

fd	HTTP data
ipp	IPP data

Return Value

Current state

ippWriteIO

CUPS 1.2/macOS 10.5

Write data for an IPP message.

```
ipp_state_t ippWriteIO(void *dst, ipp_iocb_t cb, int blocking, ipp_t *parent, ipp_t *ipp);
```

Parameters

dst	Destination
cb	Write callback function
blocking	Use blocking IO?
parent	Parent IPP message
ipp	IPP data

Return Value

Current state

Data Types

cups_client_cert_cb_t

CUPS 1.5/macOS 10.7

Client credentials callback

```
typedef int(*)(http_t *http, void *tls, cups_array_t *distinguished_names, void
*user_data)cups_client_cert_cb_t;
```

cups_dest_cb_t

CUPS 1.6/macOS 10.8

Destination enumeration callback

```
typedef int(*)(void *user_data, unsigned flags, cups_dest_t *dest)cups_dest_cb_t;
```

cups_dest_t

Destination

```
typedef struct cups_dest_s cups_dest_t;
```

cups_dinfo_t

CUPS 1.6/macOS 10.8

Destination capability and status information

```
typedef struct _cups_dinfo_s cups_dinfo_t;
```

cups_job_t

Job

```
typedef struct cups_job_s cups_job_t;
```

cups_option_t

Printer Options

```
typedef struct cups_option_s cups_option_t;
```

cups_password_cb2_t

CUPS 1.4/macOS 10.6

New password callback

```
typedef const char *(*)(const char *prompt, http_t *http, const char *method, const char *resource,
void *user_data)cups_password_cb2_t;
```

cups_ptype_t

Printer type/capability bits

```
typedef unsigned cups_ptype_t;
```

cups_server_cert_cb_t

CUPS 1.5/macOS 10.7

Server credentials callback

```
typedef int(*)(http_t *http, void *tls, cups_array_t *certs, void *user_data)cups_server_cert_cb_t;
```

cups_size_t

CUPS 1.6/macOS 10.8

Media Size

```
typedef struct cups_size_s cups_size_t;
```

http_addr_t

CUPS 1.2/macOS 10.5

Socket address union, which makes using IPv6 and other address types easier and more portable.

```
typedef union _http_addr_u / http_addr_t;
```

http_encoding_t

HTTP transfer encoding values

```
typedef enum http_encoding_e http_encoding_t;
```

http_encryption_t

HTTP encryption values

```
typedef enum http_encryption_e http_encryption_t;
```

http_field_t

HTTP field names

```
typedef enum http_field_e http_field_t;
```

http_keepalive_t

HTTP keep-alive values

```
typedef enum http_keepalive_e http_keepalive_t;
```

http_state_t

HTTP state values; states are server-oriented...

```
typedef enum http_state_e http_state_t;
```

http_t

HTTP connection type

```
typedef struct _http_s http_t;
```

http_timeout_cb_t

CUPS 1.5/macOS 10.7

HTTP timeout callback

```
typedef int(*)(http_t *http, void *user_data)http_timeout_cb_t;
```

http_trust_t

CUPS 2.0/OS 10.10

Level of trust for credentials

```
typedef enum http_trust_e http_trust_t;
```

http_uri_coding_t

URI en/decode flags

```
typedef enum http_uri_coding_e http_uri_coding_t;
```

http_uri_status_t

CUPS 1.2

URI separation status

```
typedef enum http_uri_status_e http_uri_status_t;
```

ipp_attribute_t

IPP attribute

```
typedef struct _ipp_attribute_s ipp_attribute_t;
```

ipp_copycb_t

CUPS 1.6/macOS 10.8

ippCopyAttributes callback function

```
typedef int(*)(void *context, ipp_t *dst, ipp_attribute_t *attr)ipp_copycb_t;
```

ipp_iocb_t

CUPS 1.2/macOS 10.5

ippReadIO/ippWriteIO callback function

```
typedef ssize_t(*)(void *context, ipp_uchar_t *buffer, size_t bytes) ipp_iocb_t;
```

ipp_orient_t

Orientation values

```
typedef enum ipp_orient_e ipp_orient_t;
```

ipp_pstate_t

Printer state values

```
typedef enum ipp_pstate_e ipp_pstate_t;
```

ipp_quality_t

Print quality values

```
typedef enum ipp_quality_e ipp_quality_t;
```

ipp_res_t

Resolution units

```
typedef enum ipp_res_e ipp_res_t;
```

ipp_rstate_t

resource-state values

```
typedef enum ipp_rstate_e ipp_rstate_t;
```

ipp_sstate_t

system-state values

```
typedef enum ipp_sstate_e ipp_sstate_t;
```

ipp_state_t

ipp_t state values

```
typedef enum ipp_state_e ipp_state_t;
```

ipp_t

IPP request/response data

```
typedef struct _ipp_s ipp_t;
```

Structures

cups_dest_s

Destination

```
struct cups_dest_s {
    char *name, *instance;
    int is_default;
    int num_options;
    cups_option_t *options;
};
```

Members

instance	Local instance name or NULL
is_default	Is this printer the default?
num_options	Number of options
options	Options

cups_job_s

Job

```
struct cups_job_s {
    time_t completed_time;
    time_t creation_time;
    char *dest;
    char *format;
    int id;
    int priority;
    time_t processing_time;
    int size;
    ipp_jstate_t state;
    char *title;
    char *user;
};
```

Members

completed_time	Time the job was completed
creation_time	Time the job was created
dest	Printer or class name
format	Document format
id	The job ID
priority	Priority (1-100)
processing_time	Time the job was processed
size	Size in kilobytes
state	Job state
title	Title/job name

user	User that submitted the job

cups_option_s

Printer Options

```
struct cups_option_s {
    char *name;
    char *value;
};
```

Members

name	Name of option
value	Value of option

cups_size_s

CUPS 1.6/macOS 10.8

Media Size

```
struct cups_size_s {
    char media[128];
    int width, length, bottom, left, right, top;
};
```

Members

media[128]	Media name to use
top	Top margin in hundredths of millimeters

Constants

cups_ptype_e

Printer type/capability bit constants

Constants

CUPS_PRINTER_AUTHENTICATED	***CUPS 1.2/macOS 10.5***	Printer requires authentication
CUPS_PRINTER_BIND		Can bind output

CUPS_PRINTER_BW	Can do B&W printing
CUPS_PRINTER_CLASS	Printer class
CUPS_PRINTER_COLLATE	Can quickly collate copies
CUPS_PRINTER_COLOR	Can do color printing
CUPS_PRINTER_COMMANDS *CUPS 1.2/macOS 10.5*	Printer supports maintenance commands
CUPS_PRINTER_COPIES	Can do copies in hardware
CUPS_PRINTER_COVER	Can cover output
CUPS_PRINTER_DEFAULT	Default printer on network
CUPS_PRINTER_DISCOVERED *CUPS 1.2/macOS 10.5*	Printer was discovered
CUPS_PRINTER_DUPLEX	Can do two-sided printing
CUPS_PRINTER_FAX	Fax queue
CUPS_PRINTER_LARGE	Can print on D/E/A1/A0-size media
CUPS_PRINTER_LOCAL	Local printer or class
CUPS_PRINTER_MEDIUM	Can print on Tabloid/B/C/A3/A2-size media
CUPS_PRINTER_NOT_SHARED *CUPS 1.2/macOS 10.5*	Printer is not shared
CUPS_PRINTER_PUNCH	Can punch output
CUPS_PRINTER_REJECTING	Printer is rejecting jobs
CUPS_PRINTER_REMOTE	Remote printer or class
CUPS_PRINTER_SMALL	Can print on Letter/Legal/A4-size media
CUPS_PRINTER_SORT	Can sort output
CUPS_PRINTER_STAPLE	Can staple output
CUPS_PRINTER_VARIABLE	Can print on rolls and custom-size media

http_encoding_e

HTTP transfer encoding values

Constants

HTTP_ENCODING_CHUNKED	Data is chunked
HTTP_ENCODING_FIELDS	Sending HTTP fields
HTTP_ENCODING_LENGTH	Data is sent with Content-Length

http_encryption_e

HTTP encryption values

Constants

HTTP_ENCRYPTION_ALWAYS	Always encrypt (SSL)
HTTP_ENCRYPTION_IF_REQUESTED	Encrypt if requested (TLS upgrade)
HTTP_ENCRYPTION_NEVER	Never encrypt
HTTP_ENCRYPTION_REQUIRED	Encryption is required (TLS upgrade)

http_field_e

HTTP field names

Constants

HTTP_FIELD_ACCEPT_ENCODING *CUPS 1.7/macOS 10.9*	Accepting-Encoding field
HTTP_FIELD_ACCEPT_LANGUAGE	Accept-Language field
HTTP_FIELD_ACCEPT_RANGES	Accept-Ranges field
HTTP_FIELD_ALLOW *CUPS 1.7/macOS 10.9*	Allow field
HTTP_FIELD_AUTHENTICATION_INFO *CUPS 2.2.9)*	Authentication-Info field (
HTTP_FIELD_AUTHORIZATION	Authorization field
HTTP_FIELD_CONNECTION	Connection field
HTTP_FIELD_CONTENT_ENCODING	Content-Encoding field
HTTP_FIELD_CONTENT_LANGUAGE	Content-Language field
HTTP_FIELD_CONTENT_LENGTH	Content-Length field
HTTP_FIELD_CONTENT_LOCATION	Content-Location field
HTTP_FIELD_CONTENT_MD5	Content-MD5 field
HTTP_FIELD_CONTENT_RANGE	Content-Range field
HTTP_FIELD_CONTENT_TYPE	Content-Type field
HTTP_FIELD_CONTENT_VERSION	Content-Version field
HTTP_FIELD_DATE	Date field
HTTP_FIELD_HOST	Host field
HTTP_FIELD_IF_MODIFIED_SINCE	If-Modified-Since field

HTTP_FIELD_IF_UNMODIFIED_SINCE	If-Unmodified-Since field
HTTP_FIELD_KEEP_ALIVE	Keep-Alive field
HTTP_FIELD_LAST_MODIFIED	Last-Modified field
HTTP_FIELD_LINK	Link field
HTTP_FIELD_LOCATION	Location field
HTTP_FIELD_MAX	Maximum field index
HTTP_FIELD_RANGE	Range field
HTTP_FIELD_REFERER	Referer field
HTTP_FIELD_RETRY_AFTER	Retry-After field
HTTP_FIELD_SERVER *CUPS 1.7/macOS 10.9*	Server field
HTTP_FIELD_TRANSFER_ENCODING	Transfer-Encoding field
HTTP_FIELD_UNKNOWN	Unknown field
HTTP_FIELD_UPGRADE	Upgrade field
HTTP_FIELD_USER_AGENT	User-Agent field
HTTP_FIELD_WWW_AUTHENTICATE	WWW-Authenticate field

http_keepalive_e

HTTP keep-alive values

Constants

HTTP_KEEPALIVE_OFF	No keep alive support
HTTP_KEEPALIVE_ON	Use keep alive

http_state_e

HTTP state values; states are server-oriented...

Constants

HTTP_STATE_CONNECT	CONNECT command, waiting for blank line
HTTP_STATE_DELETE	DELETE command, waiting for blank line
HTTP_STATE_ERROR	Error on socket
HTTP_STATE_GET	GET command, waiting for blank line

HTTP_STATE_GET_SEND	GET command, sending data
HTTP_STATE_HEAD	HEAD command, waiting for blank line
HTTP_STATE_OPTIONS	OPTIONS command, waiting for blank line
HTTP_STATE_POST	POST command, waiting for blank line
HTTP_STATE_POST_RECV	POST command, receiving data
HTTP_STATE_POST_SEND	POST command, sending data
HTTP_STATE_PUT	PUT command, waiting for blank line
HTTP_STATE_PUT_RECV	PUT command, receiving data
HTTP_STATE_STATUS	Command complete, sending status
HTTP_STATE_TRACE	TRACE command, waiting for blank line
HTTP_STATE_UNKNOWN_METHOD *CUPS 1.7/macOS 10.9*	Unknown request method, waiting for blank line
HTTP_STATE_UNKNOWN_VERSION *CUPS 1.7/macOS 10.9*	Unknown request method, waiting for blank line
HTTP_STATE_WAITING	Waiting for command

http_status_e

HTTP status codes

Constants

HTTP_STATUS_ACCEPTED	DELETE command was successful
HTTP_STATUS_BAD_GATEWAY	Bad gateway
HTTP_STATUS_BAD_REQUEST	Bad request
HTTP_STATUS_CONFLICT	Request is self-conflicting
HTTP_STATUS_CONTINUE	Everything OK, keep going...
HTTP_STATUS_CREATED	PUT command was successful
HTTP_STATUS_CUPS_AUTHORIZATION_CANCELED *CUPS 1.4*	User canceled authorization
HTTP_STATUS_CUPS_PKI_ERROR *CUPS 1.5/macOS 10.7*	Error negotiating a secure connection
HTTP_STATUS_ERROR	An error response from httpXxxx()
HTTP_STATUS_EXPECTATION_FAILED	The expectation given in an Expect header field was not met
HTTP_STATUS_FORBIDDEN	Forbidden to access this URI

HTTP_STATUS_FOUND	Document was found at a different URI
HTTP_STATUS_GATEWAY_TIMEOUT	Gateway connection timed out
HTTP_STATUS_GONE	Server has gone away
HTTP_STATUS_LENGTH_REQUIRED	A content length or encoding is required
HTTP_STATUS_METHOD_NOT_ALLOWED	Method is not allowed
HTTP_STATUS_MOVED_PERMANENTLY	Document has moved permanently
HTTP_STATUS_MULTIPLE_CHOICES	Multiple files match request
HTTP_STATUS_NONE *CUPS 1.7/macOS 10.9*	No Expect value
HTTP_STATUS_NOT_ACCEPTABLE	Not Acceptable
HTTP_STATUS_NOT_AUTHORITATIVE	Information isn't authoritative
HTTP_STATUS_NOT_FOUND	URI was not found
HTTP_STATUS_NOT_IMPLEMENTED	Feature not implemented
HTTP_STATUS_NOT_MODIFIED	File not modified
HTTP_STATUS_NOT_SUPPORTED	HTTP version not supported
HTTP_STATUS_NO_CONTENT	Successful command, no new data
HTTP_STATUS_OK	OPTIONS/GET/HEAD/POST/TRACE command was successful
HTTP_STATUS_PARTIAL_CONTENT	Only a partial file was received/sent
HTTP_STATUS_PAYMENT_REQUIRED	Payment required
HTTP_STATUS_PRECONDITION	Precondition failed
HTTP_STATUS_PROXY_AUTHENTICATION	Proxy Authentication is Required
HTTP_STATUS_REQUESTED_RANGE	The requested range is not satisfiable
HTTP_STATUS_REQUEST_TIMEOUT	Request timed out
HTTP_STATUS_REQUEST_TOO_LARGE	Request entity too large
HTTP_STATUS_RESET_CONTENT	Content was reset/recreated
HTTP_STATUS_SEE_OTHER	See this other link
HTTP_STATUS_SERVER_ERROR	Internal server error
HTTP_STATUS_SERVICE_UNAVAILABLE	Service is unavailable
HTTP_STATUS_SWITCHING_PROTOCOLS	HTTP upgrade to TLS/SSL
HTTP_STATUS_TEMPORARY_REDIRECT	Temporary redirection
HTTP_STATUS_UNAUTHORIZED	Unauthorized to access host
HTTP_STATUS_UNSUPPORTED_MEDIATYPE	The requested media type is unsupported

HTTP_STATUS_UPGRADE_REQUIRED	Upgrade to SSL/TLS required
HTTP_STATUS_URI_TOO_LONG	URI too long
HTTP_STATUS_USE_PROXY	Must use a proxy to access this URI

http_trust_e

CUPS 2.0/OS 10.10

Level of trust for credentials

Constants

HTTP_TRUST_CHANGED	Credentials have changed
HTTP_TRUST_EXPIRED	Credentials are expired
HTTP_TRUST_INVALID	Credentials are invalid
HTTP_TRUST_OK	Credentials are OK/trusted
HTTP_TRUST_RENEWED	Credentials have been renewed
HTTP_TRUST_UNKNOWN	Credentials are unknown/new

http_uri_coding_e

URI en/decode flags

Constants

HTTP_URI_CODING_ALL	En/decode everything
HTTP_URI_CODING_HOSTNAME	En/decode the hostname portion
HTTP_URI_CODING_MOST	En/decode all but the query
HTTP_URI_CODING_NONE	Don't en/decode anything
HTTP_URI_CODING_QUERY	En/decode the query portion
HTTP_URI_CODING_RESOURCE	En/decode the resource portion
HTTP_URI_CODING_RFC6874	Use RFC 6874 address format
HTTP_URI_CODING_USERNAME	En/decode the username portion

http_uri_status_e

CUPS 1.2

URI separation status

Constants

HTTP_URI_STATUS_BAD_ARGUMENTS	Bad arguments to function (error)
HTTP_URI_STATUS_BAD_HOSTNAME	Bad hostname in URI (error)
HTTP_URI_STATUS_BAD_PORT	Bad port number in URI (error)
HTTP_URI_STATUS_BAD_RESOURCE	Bad resource in URI (error)
HTTP_URI_STATUS_BAD_SCHEME	Bad scheme in URI (error)
HTTP_URI_STATUS_BAD_URI	Bad/empty URI (error)
HTTP_URI_STATUS_BAD_USERNAME	Bad username in URI (error)
HTTP_URI_STATUS_MISSING_RESOURCE	Missing resource in URI (warning)
HTTP_URI_STATUS_MISSING_SCHEME	Missing scheme in URI (warning)
HTTP_URI_STATUS_OK	URI decoded OK
HTTP_URI_STATUS_OVERFLOW	URI buffer for httpAssembleURI is too small
HTTP_URI_STATUS_UNKNOWN_SCHEME	Unknown scheme in URI (warning)

ipp_finishings_e

Finishings values

Constants

IPP_FINISHINGS_BALE	Bale (any type)
IPP_FINISHINGS_BIND	Bind
IPP_FINISHINGS_BIND_BOTTOM	Bind on bottom
IPP_FINISHINGS_BIND_LEFT	Bind on left
IPP_FINISHINGS_BIND_RIGHT	Bind on right
IPP_FINISHINGS_BIND_TOP	Bind on top
IPP_FINISHINGS_BOOKLET_MAKER	Fold to make booklet
IPP_FINISHINGS_COAT	Apply protective liquid or powder coating
IPP_FINISHINGS_COVER	Add cover
IPP_FINISHINGS_EDGE_STITCH	Stitch along any side
IPP_FINISHINGS_EDGE_STITCH_BOTTOM	Stitch along bottom edge
IPP_FINISHINGS_EDGE_STITCH_LEFT	Stitch along left side
IPP_FINISHINGS_EDGE_STITCH_RIGHT	Stitch along right side

IPP_FINISHINGS_EDGE_STITCH_TOP	Stitch along top edge
IPP_FINISHINGS_FOLD	Fold (any type)
IPP_FINISHINGS_FOLD_ACCORDION	Accordion-fold the paper vertically into four sections
IPP_FINISHINGS_FOLD_DOUBLE_GATE	Fold the top and bottom quarters of the paper towards the midline, then fold in half vertically
IPP_FINISHINGS_FOLD_ENGINEERING_Z	Fold the paper vertically into two small sections and one larger, forming an elongated Z
IPP_FINISHINGS_FOLD_GATE	Fold the top and bottom quarters of the paper towards the midline
IPP_FINISHINGS_FOLD_HALF	Fold the paper in half vertically
IPP_FINISHINGS_FOLD_HALF_Z	Fold the paper in half horizontally, then Z-fold the paper vertically
IPP_FINISHINGS_FOLD_LEFT_GATE	Fold the top quarter of the paper towards the midline
IPP_FINISHINGS_FOLD_LETTER	Fold the paper into three sections vertically; sometimes also known as a C fold
IPP_FINISHINGS_FOLD_PARALLEL	Fold the paper in half vertically two times, yielding four sections
IPP_FINISHINGS_FOLD_POSTER	Fold the paper in half horizontally and vertically; sometimes also called a cross fold
IPP_FINISHINGS_FOLD_RIGHT_GATE	Fold the bottom quarter of the paper towards the midline
IPP_FINISHINGS_FOLD_Z	Fold the paper vertically into three sections, forming a Z
IPP_FINISHINGS_JOG_OFFSET	Offset for binding (any type)
IPP_FINISHINGS_LAMINATE	Apply protective (solid) material
IPP_FINISHINGS_NONE	No finishing
IPP_FINISHINGS_PUNCH	Punch (any location/count)
IPP_FINISHINGS_PUNCH_BOTTOM_LEFT	Punch 1 hole bottom left
IPP_FINISHINGS_PUNCH_BOTTOM_RIGHT	Punch 1 hole bottom right
IPP_FINISHINGS_PUNCH_DUAL_BOTTOM	Punch 2 holes bottom edge
IPP_FINISHINGS_PUNCH_DUAL_LEFT	Punch 2 holes left side
IPP_FINISHINGS_PUNCH_DUAL_RIGHT	Punch 2 holes right side
IPP_FINISHINGS_PUNCH_DUAL_TOP	Punch 2 holes top edge
IPP_FINISHINGS_PUNCH_MULTIPLE_BOTTOM	Punch multiple holes bottom edge

IPP_FINISHINGS_PUNCH_MULTIPLE_LEFT	Punch multiple holes left side
IPP_FINISHINGS_PUNCH_MULTIPLE_RIGHT	Punch multiple holes right side
IPP_FINISHINGS_PUNCH_MULTIPLE_TOP	Punch multiple holes top edge
IPP_FINISHINGS_PUNCH_QUAD_BOTTOM	Punch 4 holes bottom edge
IPP_FINISHINGS_PUNCH_QUAD_LEFT	Punch 4 holes left side
IPP_FINISHINGS_PUNCH_QUAD_RIGHT	Punch 4 holes right side
IPP_FINISHINGS_PUNCH_QUAD_TOP	Punch 4 holes top edge
IPP_FINISHINGS_PUNCH_TOP_LEFT	Punch 1 hole top left
IPP_FINISHINGS_PUNCH_TOP_RIGHT	Punch 1 hole top right
IPP_FINISHINGS_PUNCH_TRIPLE_BOTTOM	Punch 3 holes bottom edge
IPP_FINISHINGS_PUNCH_TRIPLE_LEFT	Punch 3 holes left side
IPP_FINISHINGS_PUNCH_TRIPLE_RIGHT	Punch 3 holes right side
IPP_FINISHINGS_PUNCH_TRIPLE_TOP	Punch 3 holes top edge
IPP_FINISHINGS_SADDLE_STITCH	Staple interior
IPP_FINISHINGS_STAPLE	Staple (any location/method)
IPP_FINISHINGS_STAPLE_BOTTOM_LEFT	Staple bottom left corner
IPP_FINISHINGS_STAPLE_BOTTOM_RIGHT	Staple bottom right corner
IPP_FINISHINGS_STAPLE_DUAL_BOTTOM	Two staples on bottom
IPP_FINISHINGS_STAPLE_DUAL_LEFT	Two staples on left
IPP_FINISHINGS_STAPLE_DUAL_RIGHT	Two staples on right
IPP_FINISHINGS_STAPLE_DUAL_TOP	Two staples on top
IPP_FINISHINGS_STAPLE_TOP_LEFT	Staple top left corner
IPP_FINISHINGS_STAPLE_TOP_RIGHT	Staple top right corner
IPP_FINISHINGS_STAPLE_TRIPLE_BOTTOM	Three staples on bottom
IPP_FINISHINGS_STAPLE_TRIPLE_LEFT	Three staples on left
IPP_FINISHINGS_STAPLE_TRIPLE_RIGHT	Three staples on right
IPP_FINISHINGS_STAPLE_TRIPLE_TOP	Three staples on top
IPP_FINISHINGS_TRIM	Trim (any type)
IPP_FINISHINGS_TRIM_AFTER_COPIES	Trim output after each copy
IPP_FINISHINGS_TRIM_AFTER_DOCUMENTS	Trim output after each document
IPP_FINISHINGS_TRIM_AFTER_JOB	Trim output after job
IPP_FINISHINGS_TRIM_AFTER_PAGES	Trim output after each page

ipp_jstate_e

Job states

Constants

IPP_JSTATE_ABORTED	Job has aborted due to error
IPP_JSTATE_CANCELED	Job has been canceled
IPP_JSTATE_COMPLETED	Job has completed successfully
IPP_JSTATE_HELD	Job is held for printing
IPP_JSTATE_PENDING	Job is waiting to be printed
IPP_JSTATE_PROCESSING	Job is currently printing
IPP_JSTATE_STOPPED	Job has been stopped

ipp_op_e

IPP operations

Constants

IPP_OP_ALLOCATE_PRINTER_RESOURCES	Allocate-Printer-Resources: Use resources for a printer.
IPP_OP_CANCEL_CURRENT_JOB	Cancel-Current-Job: Cancel the current job
IPP_OP_CANCEL_JOB	Cancel-Job: Cancel a job
IPP_OP_CANCEL_JOBS	Cancel-Jobs: Cancel all jobs (administrative)
IPP_OP_CANCEL_MY_JOBS	Cancel-My-Jobs: Cancel a user's jobs
IPP_OP_CANCEL_RESOURCE	Cancel-Resource: Uninstall a resource.
IPP_OP_CANCEL_SUBSCRIPTION *CUPS 1.2/macOS 10.5*	Cancel-Subscription: Cancel a subscription
IPP_OP_CLOSE_JOB	Close-Job: Close a job and start printing
IPP_OP_CREATE_JOB	Create-Job: Create an empty print job
IPP_OP_CREATE_JOB_SUBSCRIPTIONS *CUPS 1.2/macOS 10.5*	Create-Job-Subscriptions: Create one of more job subscriptions
IPP_OP_CREATE_PRINTER	Create-Printer: Create a new service.
IPP_OP_CREATE_PRINTER_SUBSCRIPTIONS *CUPS 1.2/macOS 10.5*	Create-Printer-Subscriptions: Create one or more printer subscriptions

IPP_OP_CREATE_RESOURCE	Create-Resource: Create a new (empty) resource.
IPP_OP_CREATE_RESOURCE_SUBSCRIPTIONS	Create-Resource-Subscriptions: Create event subscriptions for a resource.
IPP_OP_CREATE_SYSTEM_SUBSCRIPTIONS	Create-System-Subscriptions: Create event subscriptions for a system.
IPP_OP_CUPS_ADD_MODIFY_CLASS	CUPS-Add-Modify-Class: Add or modify a class
IPP_OP_CUPS_ADD_MODIFY_PRINTER	CUPS-Add-Modify-Printer: Add or modify a printer
IPP_OP_CUPS_AUTHENTICATE_JOB *CUPS 1.2/macOS 10.5*	CUPS-Authenticate-Job: Authenticate a job
IPP_OP_CUPS_CREATE_LOCAL_PRINTER *CUPS 2.2*	CUPS-Create-Local-Printer: Create a local (temporary) printer
IPP_OP_CUPS_DELETE_CLASS	CUPS-Delete-Class: Delete a class
IPP_OP_CUPS_DELETE_PRINTER	CUPS-Delete-Printer: Delete a printer
IPP_OP_CUPS_GET_DEFAULT	CUPS-Get-Default: Get the default printer
IPP_OP_CUPS_GET_DEVICES *DEPRECATED*	CUPS-Get-Devices: Get a list of supported devices
IPP_OP_CUPS_GET_DOCUMENT *CUPS 1.4/macOS 10.6*	CUPS-Get-Document: Get a document file
IPP_OP_CUPS_GET_PPD *DEPRECATED*	CUPS-Get-PPD: Get a PPD file
IPP_OP_CUPS_GET_PPDS *DEPRECATED*	CUPS-Get-PPDs: Get a list of supported drivers
IPP_OP_CUPS_GET_PRINTERS	CUPS-Get-Printers: Get a list of printers and/or classes
IPP_OP_CUPS_INVALID	Invalid operation name for `ippOpValue`
IPP_OP_CUPS_MOVE_JOB	CUPS-Move-Job: Move a job to a different printer
IPP_OP_CUPS_SET_DEFAULT	CUPS-Set-Default: Set the default printer
IPP_OP_DEALLOCATE_PRINTER_RESOURCES	Deallocate-Printer-Resources: Stop using resources for a printer.
IPP_OP_DELETE_PRINTER	Delete-Printer: Delete an existing service.
IPP_OP_DISABLE_ALL_PRINTERS	Disable-All-Printers: Stop accepting new jobs on all services.
IPP_OP_DISABLE_PRINTER	Disable-Printer: Reject new jobs for a printer

IPP_OP_ENABLE_ALL_PRINTERS	Enable-All-Printers: Start accepting new jobs on all services.
IPP_OP_ENABLE_PRINTER	Enable-Printer: Accept new jobs for a printer
IPP_OP_GET_JOBS	Get-Jobs: Get a list of jobs
IPP_OP_GET_JOB_ATTRIBUTES	Get-Job-Attribute: Get information about a job
IPP_OP_GET_NOTIFICATIONS ***CUPS 1.2/macOS 10.5***	Get-Notifications: Get notification events
IPP_OP_GET_PRINTERS	Get-Printers: Get a list of services.
IPP_OP_GET_PRINTER_ATTRIBUTES	Get-Printer-Attributes: Get information about a printer
IPP_OP_GET_PRINTER_SUPPORTED_VALUES	Get-Printer-Supported-Values: Get supported values
IPP_OP_GET_SUBSCRIPTIONS ***CUPS 1.2/macOS 10.5***	Get-Subscriptions: Get list of subscriptions
IPP_OP_GET_SUBSCRIPTION_ATTRIBUTES ***CUPS 1.2/macOS 10.5***	Get-Subscription-Attributes: Get subscription information
IPP_OP_GET_SYSTEM_ATTRIBUTES	Get-System-Attributes: Get system object attributes.
IPP_OP_GET_SYSTEM_SUPPORTED_VALUES	Get-System-Supported-Values: Get supported values for system object attributes.
IPP_OP_HOLD_JOB	Hold-Job: Hold a job for printing
IPP_OP_HOLD_NEW_JOBS	Hold-New-Jobs: Hold new jobs
IPP_OP_IDENTIFY_PRINTER	Identify-Printer: Make the printer beep, flash, or display a message for identification
IPP_OP_INSTALL_RESOURCE	Install-Resource: Install a resource.
IPP_OP_PAUSE_ALL_PRINTERS	Pause-All-Printers: Stop all services immediately.
IPP_OP_PAUSE_ALL_PRINTERS_AFTER_CURRENT_JOB	Pause-All-Printers-After-Current-Job: Stop all services after processing the current jobs.
IPP_OP_PAUSE_PRINTER	Pause-Printer: Stop a printer
IPP_OP_PAUSE_PRINTER_AFTER_CURRENT_JOB	Pause-Printer-After-Current-Job: Stop printer after the current job
IPP_OP_PRINT_JOB	Print-Job: Print a single file

IPP_OP_PROMOTE_JOB	Promote-Job: Promote a job to print sooner
IPP_OP_REGISTER_OUTPUT_DEVICE	Register-Output-Device: Register a remote service.
IPP_OP_RELEASE_HELD_NEW_JOBS	Release-Held-New-Jobs: Release new jobs that were previously held
IPP_OP_RELEASE_JOB	Release-Job: Release a job for printing
IPP_OP_RENEW_SUBSCRIPTION *CUPS 1.2/macOS 10.5*	Renew-Subscription: Renew a printer subscription
IPP_OP_RESTART_JOB *DEPRECATED*	Restart-Job: Reprint a job
IPP_OP_RESTART_SYSTEM	Restart-System: Restart all services.
IPP_OP_RESUME_ALL_PRINTERS	Resume-All-Printers: Start job processing on all services.
IPP_OP_RESUME_JOB	Resume-Job: Resume the current job
IPP_OP_RESUME_PRINTER	Resume-Printer: Start a printer
IPP_OP_SCHEDULE_JOB_AFTER	Schedule-Job-After: Schedule a job to print after another
IPP_OP_SEND_DOCUMENT	Send-Document: Add a file to a job
IPP_OP_SEND_RESOURCE_DATA	Send-Resource-Data: Upload the data for a resource.
IPP_OP_SET_JOB_ATTRIBUTES	Set-Job-Attributes: Set job values
IPP_OP_SET_PRINTER_ATTRIBUTES	Set-Printer-Attributes: Set printer values
IPP_OP_SET_RESOURCE_ATTRIBUTES	Set-Resource-Attributes: Set resource object attributes.
IPP_OP_SET_SYSTEM_ATTRIBUTES	Set-System-Attributes: Set system object attributes.
IPP_OP_SHUTDOWN_ALL_PRINTERS	Shutdown-All-Printers: Shutdown all services.
IPP_OP_SHUTDOWN_ONE_PRINTER	Shutdown-One-Printer: Shutdown a service.
IPP_OP_STARTUP_ALL_PRINTERS	Startup-All-Printers: Startup all services.
IPP_OP_STARTUP_ONE_PRINTER	Startup-One-Printer: Start a service.
IPP_OP_SUSPEND_CURRENT_JOB	Suspend-Current-Job: Suspend the current job
IPP_OP_VALIDATE_JOB	Validate-Job: Validate job values prior to submission

ipp_orient_e

Orientation values

Constants

IPP_ORIENT_LANDSCAPE	90 degrees counter-clockwise
IPP_ORIENT_NONE	No rotation
IPP_ORIENT_PORTRAIT	No rotation
IPP_ORIENT_REVERSE_LANDSCAPE	90 degrees clockwise
IPP_ORIENT_REVERSE_PORTRAIT	180 degrees

ipp_pstate_e

Printer state values

Constants

IPP_PSTATE_IDLE	Printer is idle
IPP_PSTATE_PROCESSING	Printer is working
IPP_PSTATE_STOPPED	Printer is stopped

ipp_quality_e

Print quality values

Constants

IPP_QUALITY_DRAFT	Draft quality
IPP_QUALITY_HIGH	High quality
IPP_QUALITY_NORMAL	Normal quality

ipp_res_e

Resolution units

Constants

IPP_RES_PER_CM	Pixels per centimeter

IPP_RES_PER_INCH	Pixels per inch

ipp_rstate_e

resource-state values

Constants

IPP_RSTATE_ABORTED	Resource has been aborted and is pending deletion.
IPP_RSTATE_AVAILABLE	Resource is available for installation.
IPP_RSTATE_CANCELED	Resource has been canceled and is pending deletion.
IPP_RSTATE_INSTALLED	Resource is installed.
IPP_RSTATE_PENDING	Resource is created but has no data yet.

ipp_sstate_e

system-state values

Constants

IPP_SSTATE_IDLE	At least one printer is idle and none are processing a job.
IPP_SSTATE_PROCESSING	At least one printer is processing a job.
IPP_SSTATE_STOPPED	All printers are stopped.

ipp_state_e

ipp_t state values

Constants

IPP_STATE_ATTRIBUTE	One or more attributes need to be sent/received
IPP_STATE_DATA	IPP request data needs to be sent/received
IPP_STATE_ERROR	An error occurred
IPP_STATE_HEADER	The request header needs to be sent/received
IPP_STATE_IDLE	Nothing is happening/request completed

ipp_status_e

IPP status code values

Constants

IPP_STATUS_CUPS_INVALID	Invalid status name for `ippErrorValue`
IPP_STATUS_ERROR_ACCOUNT_AUTHORIZATION_FAILED	client-error-account-authorization-failed
IPP_STATUS_ERROR_ACCOUNT_CLOSED	client-error-account-closed
IPP_STATUS_ERROR_ACCOUNT_INFO_NEEDED	client-error-account-info-needed
IPP_STATUS_ERROR_ACCOUNT_LIMIT_REACHED	client-error-account-limit-reached
IPP_STATUS_ERROR_ATTRIBUTES_NOT_SETTABLE	client-error-attributes-not-settable
IPP_STATUS_ERROR_ATTRIBUTES_OR_VALUES	client-error-attributes-or-values-not-supported
IPP_STATUS_ERROR_BAD_REQUEST	client-error-bad-request
IPP_STATUS_ERROR_BUSY	server-error-busy
IPP_STATUS_ERROR_CHARSET	client-error-charset-not-supported
IPP_STATUS_ERROR_COMPRESSION_ERROR	client-error-compression-error
IPP_STATUS_ERROR_COMPRESSION_NOT_SUPPORTED	client-error-compression-not-supported
IPP_STATUS_ERROR_CONFLICTING	client-error-conflicting-attributes
IPP_STATUS_ERROR_CUPS_ACCOUNT_AUTHORIZATION_FAILED *DEPRECATED*	cups-error-account-authorization-failed
IPP_STATUS_ERROR_CUPS_ACCOUNT_CLOSED	cups-error-account-closed @deprecate@
IPP_STATUS_ERROR_CUPS_ACCOUNT_INFO_NEEDED *DEPRECATED*	cups-error-account-info-needed
IPP_STATUS_ERROR_CUPS_ACCOUNT_LIMIT_REACHED *DEPRECATED*	cups-error-account-limit-reached
IPP_STATUS_ERROR_CUPS_AUTHENTICATION_CANCELED *CUPS 1.5/macOS 10.7*	cups-authentication-canceled - Authentication canceled by user
IPP_STATUS_ERROR_CUPS_PKI *CUPS 1.5/macOS 10.7*	cups-pki-error - Error negotiating a secure connection
IPP_STATUS_ERROR_CUPS_UPGRADE_REQUIRED *CUPS 1.5/macOS 10.7*	cups-upgrade-required - TLS upgrade required
IPP_STATUS_ERROR_DEVICE	server-error-device-error
IPP_STATUS_ERROR_DOCUMENT_ACCESS	client-error-document-access-error

IPP_STATUS_ERROR_DOCUMENT_FORMAT_ERROR	client-error-document-format-error
IPP_STATUS_ERROR_DOCUMENT_FORMAT_NOT_SUPPORTED	client-error-document-format-not-supported
IPP_STATUS_ERROR_DOCUMENT_PASSWORD	client-error-document-password-error
IPP_STATUS_ERROR_DOCUMENT_PERMISSION	client-error-document-permission-error
IPP_STATUS_ERROR_DOCUMENT_SECURITY	client-error-document-security-error
IPP_STATUS_ERROR_DOCUMENT_UNPRINTABLE	client-error-document-unprintable-error
IPP_STATUS_ERROR_FORBIDDEN	client-error-forbidden
IPP_STATUS_ERROR_GONE	client-error-gone
IPP_STATUS_ERROR_IGNORED_ALL_SUBSCRIPTIONS	client-error-ignored-all-subscriptions
IPP_STATUS_ERROR_INTERNAL	server-error-internal-error
IPP_STATUS_ERROR_JOB_CANCELED	server-error-job-canceled
IPP_STATUS_ERROR_MULTIPLE_JOBS_NOT_SUPPORTED	server-error-multiple-document-jobs-not-supported
IPP_STATUS_ERROR_NOT_ACCEPTING_JOBS	server-error-not-accepting-jobs
IPP_STATUS_ERROR_NOT_AUTHENTICATED	client-error-not-authenticated
IPP_STATUS_ERROR_NOT_AUTHORIZED	client-error-not-authorized
IPP_STATUS_ERROR_NOT_FETCHABLE	client-error-not-fetchable
IPP_STATUS_ERROR_NOT_FOUND	client-error-not-found
IPP_STATUS_ERROR_NOT_POSSIBLE	client-error-not-possible
IPP_STATUS_ERROR_OPERATION_NOT_SUPPORTED	server-error-operation-not-supported
IPP_STATUS_ERROR_PRINTER_IS_DEACTIVATED	server-error-printer-is-deactivated
IPP_STATUS_ERROR_REQUEST_ENTITY	client-error-request-entity-too-large
IPP_STATUS_ERROR_REQUEST_VALUE	client-error-request-value-too-long
IPP_STATUS_ERROR_SERVICE_UNAVAILABLE	server-error-service-unavailable
IPP_STATUS_ERROR_TEMPORARY	server-error-temporary-error

IPP_STATUS_ERROR_TIMEOUT	client-error-timeout
IPP_STATUS_ERROR_TOO_MANY_DOCUMENTS	server-error-too-many-documents
IPP_STATUS_ERROR_TOO_MANY_JOBS	server-error-too-many-jobs
IPP_STATUS_ERROR_TOO_MANY_SUBSCRIPTIONS	client-error-too-many-subscriptions
IPP_STATUS_ERROR_URI_SCHEME	client-error-uri-scheme-not-supported
IPP_STATUS_ERROR_VERSION_NOT_SUPPORTED	server-error-version-not-supported
IPP_STATUS_OK	successful-ok
IPP_STATUS_OK_CONFLICTING	successful-ok-conflicting-attributes
IPP_STATUS_OK_EVENTS_COMPLETE	successful-ok-events-complete
IPP_STATUS_OK_IGNORED_OR_SUBSTITUTED	successful-ok-ignored-or-substituted-attributes
IPP_STATUS_OK_IGNORED_SUBSCRIPTIONS	successful-ok-ignored-subscriptions
IPP_STATUS_OK_TOO_MANY_EVENTS	successful-ok-too-many-events

ipp_tag_e

Value and group tag values for attributes

Constants

IPP_TAG_ADMINDEFINE	Admin-defined value
IPP_TAG_BOOLEAN	Boolean value
IPP_TAG_CHARSET	Character set value
IPP_TAG_CUPS_INVALID	Invalid tag name for `ippTagValue`
IPP_TAG_DATE	Date/time value
IPP_TAG_DEFAULT	Default value
IPP_TAG_DELETEATTR	Delete-attribute value
IPP_TAG_DOCUMENT	Document group
IPP_TAG_END	End-of-attributes
IPP_TAG_ENUM	Enumeration value

IPP_TAG_EVENT_NOTIFICATION	Event group
IPP_TAG_INTEGER	Integer value
IPP_TAG_JOB	Job group
IPP_TAG_KEYWORD	Keyword value
IPP_TAG_LANGUAGE	Language value
IPP_TAG_MIMETYPE	MIME media type value
IPP_TAG_NAME	Name value
IPP_TAG_NAMELANG	Name-with-language value
IPP_TAG_NOTSETTABLE	Not-settable value
IPP_TAG_NOVALUE	No-value value
IPP_TAG_OPERATION	Operation group
IPP_TAG_PRINTER	Printer group
IPP_TAG_RANGE	Range value
IPP_TAG_RESOLUTION	Resolution value
IPP_TAG_RESOURCE	Resource group
IPP_TAG_STRING	Octet string value
IPP_TAG_SUBSCRIPTION	Subscription group
IPP_TAG_SYSTEM	System group
IPP_TAG_TEXT	Text value
IPP_TAG_TEXTLANG	Text-with-language value
IPP_TAG_UNKNOWN	Unknown value
IPP_TAG_UNSUPPORTED_GROUP	Unsupported attributes group
IPP_TAG_UNSUPPORTED_VALUE	Unsupported value
IPP_TAG_URI	URI value
IPP_TAG_URISCHEME	URI scheme value
IPP_TAG_ZERO	Zero tag - used for separators