

Common UNIX Printing System API Overview

**Michael R Sweet
Easy Software Products**

CUPS APIs

- Destinations and options
- Job submission, monitoring, and management
- PPD file access
- Lower-level HTTP and IPP
- Support functions

Destination and Option APIs

- Destinations:
 - *cupsFreeDests(), cupsGetDest(), cupsGetDests(), cupsSetDests()*
 - *New CUPS 1.3 APIs: cupsRemoveDest() and cupsSetDefaultDest()*
 - *Use to get a list of available destinations along with the user-specified defaults and instances*
- Options:
 - *cupsAddOption(), cupsFreeOptions(), cupsGetOption(), cupsParseOptions(), cupsRemoveOption()*
 - *Use to assemble a list of printer options for job submission or saving via cupsSetDests()*

Job Submission, Monitoring, and Management APIs

- Submission:
 - *cupsPrintFile() and cupsPrintFiles()*
 - *Uses option API to provide list of job options*
- Monitoring:
 - *cupsFreeJobs() and cupsGetJobs()*
 - *Job list can be limited to active or completed jobs, jobs for the current user, and jobs for a particular destination*
- Management:
 - *cupsCancelJob()*
 - *Uses job ID returned by cupsGetJobs(), cupsPrintFile(), or cupsPrintFiles()*

Print a File to the Default Printer

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

int num_dests;
cups_dest_t *dests, *default_dest;
int job_id;

num_dests = cupsGetDests(&dests);

default_dest = cupsGetDest(NULL, NULL, num_dests, dests);

job_id = cupsPrintFile(default_dest->name, "myfile.ps",
                       "my title",
                       default_dest->num_options,
                       default_dest->options);
```

PPD File Access APIs

- Get a queue's PPD file:
 - *cupsGetPPD()*, *ppdOpenFile()*, *ppdClose()*
- Localization:
 - *ppdLocalize()*
- Selecting options:
 - *cupsMarkOptions()*, *ppdConflicts()*, *ppdMarkDefaults()*,
ppdMarkOption()
- Accessing options:
 - *ppdFindOption()*, *ppdFirstOption()*, *ppdNextOption()*
 - *ppdFindCustomOption()*, *ppdFindCustomParam()*,
ppdFirstCustomParam(), *ppdNextCustomParam()*

Get the Default Printer's PPD File

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

int num_dests;
cups_dest_t *dests, *default_dest;
const char *ppd_filename;
ppd_file_t *ppd;

num_dests = cupsGetDests(&dests);

default_dest = cupsGetDest(NULL, NULL, num_dests, dests);

ppd_filename = cupsGetPPD(default_dest->name);

ppd = ppdOpenFile(ppd_filename);

ppdLocalize(ppd);
```

HTTP and IPP APIs

- HTTP functions
 - *httpAssembleURI(), httpAssembleURIf(), httpBlocking(), httpClose(), httpConnectEncrypt(), httpGet(), httpGetField(), httpPut(), httpPost(), httpSeparateURI(), httpSetBlocking(), and httpSetField()*
- IPP functions
 - *ippAdd*(), ippDelete(), ippDeleteAttribute(), ippFindAttribute(), ippNew(), ippNewRequest(), ippRead(), and ippWrite()*
- Convenience functions
 - *cupsDoAuthentication(), cupsDoFileRequest(), cupsDoRequest(), cupsGetFd(), cupsGetFile(), cupsPutFd(), cupsPutFile(), cupsSetPasswordCB(), and cupsSetUser()*

Get the Default Printer's State

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

http_t *http;
ipp_t *request, *response;
ipp_attribute_t *attr;
int num_dests;
cups_dest_t *dests, *default_dest;
char printer_uri[1024];

num_dests = cupsGetDests(&dests);

default_dest = cupsGetDest(NULL, NULL, num_dests, dests);

httpAssembleURIif(HTTP_URI_CODING_ALL, printer_uri,
                 sizeof(printer_uri), "ipp", NULL,
                 cupsServer(), ippPort(), "/printers/%s",
                 default_dest->name);
```

Get the Default Printer's State

```
http = httpConnectEncrypt(cupsServer(), ippPort(),
                          cupsEncryption());

request = ippNewRequest(IPP_GET_PRINTER_ATTRIBUTES);

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

response = cupsDoRequest(http, request);

attr = ippFindAttribute(response, "printer-state",
                        IPP_TAG_ENUM);

printf("printer state is %d.\n", attr->values[0].integer);

ippDelete(response);
httpClose(http);
```

Support Functions

- Array container
 - *Provides a variable-size array container with optional sorting*
- Files
 - *Read/write normal and compressed (gzip'd) files*
 - *Connect on TCP/IP port for read/write*
 - *Read text lines terminated by CR, LF, or CR LF*
 - *Find files using a path string*
- Directories
 - *Read directories efficiently on multiple OS's*