Creating a **CUPS** Backend

Adam Tauno Williams
awilliam@whitemice.org
GPG D95ED383

Copyright

CUPS

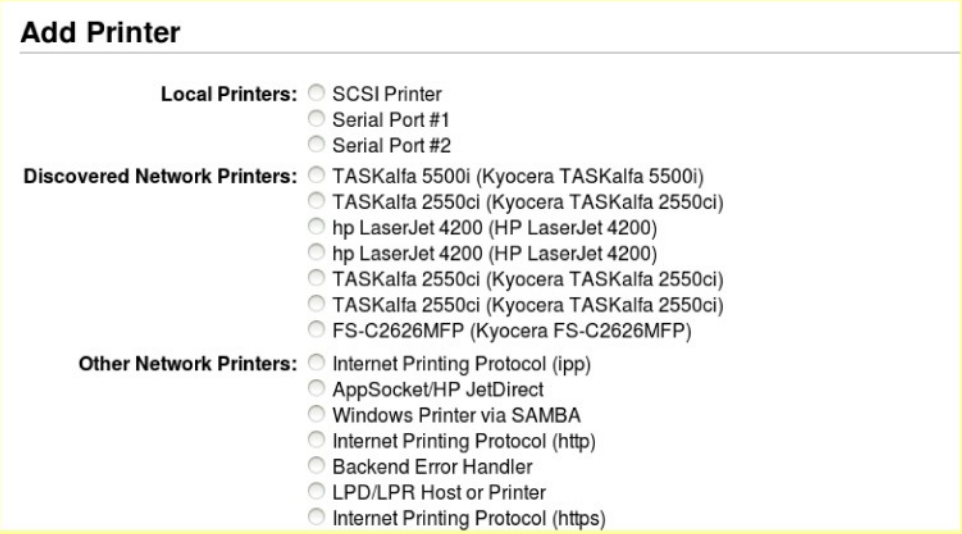
Common UNIX Printing System

If you have printed using LINUX or Mac OS/X you have used CUPS.

http://www.cups.org/

Backends

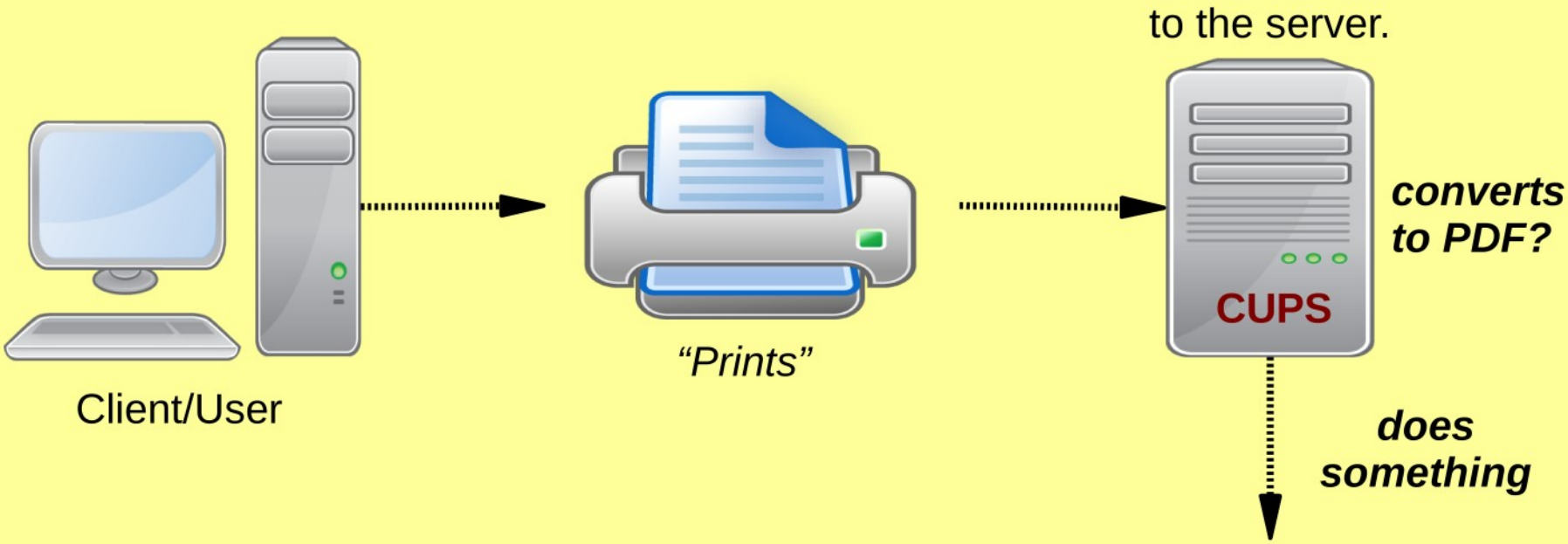
The options you see when you add a printer.



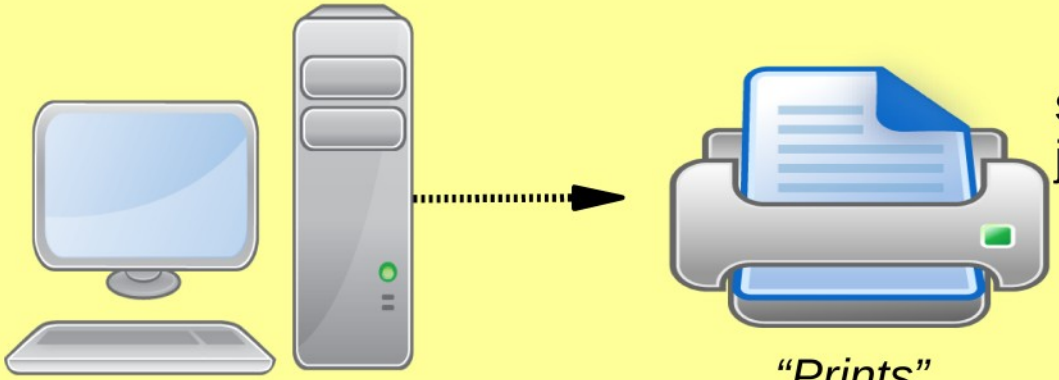
Why Create A Backend?

- Because everything can print!
  - This is a simple way to get data/documents into an application from any arbitrary thing.
    - Go paperless!
- Modify print jobs

You


to the server.
Client/User
"Prints"
CUPS
converts to PDF?
does something

Example



Client/User

"Prints"

sends a job via IPP

to the server.

converts to PDF

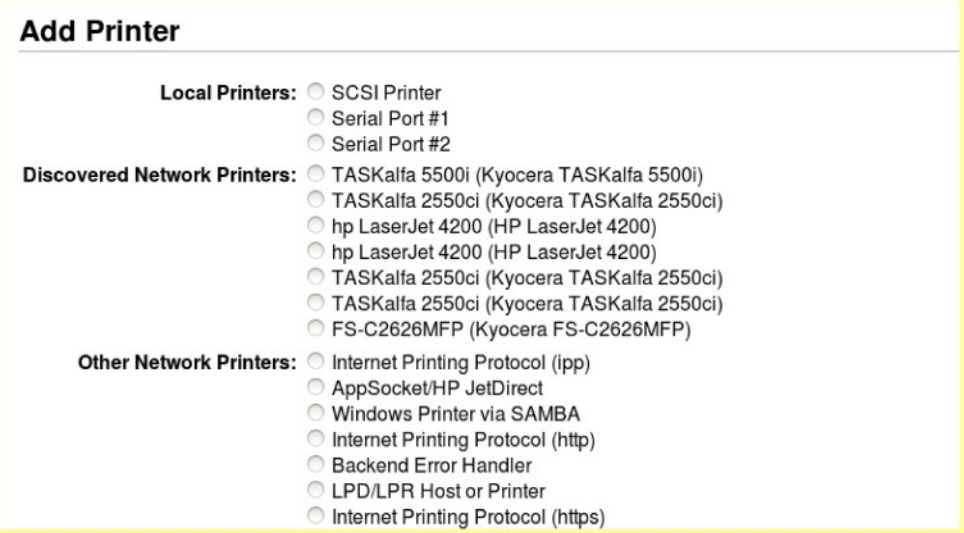
performs an HTTP PUT

Runs a work-flow, the PDF is the input

Files the PDF into a project folder, available as a WebDAV colleciton.

List Backends

linux-86wr:/home/awilliam # **lpinfo -v**
network http
network ipps
network ipp
network https
direct hp
network socket
network lpd
network smb
direct hpfax
network beh



What are these?

```
> ls -l /usr/lib/cups/backend/
total 324
-rwxr-xr-x 1 root root  7250 Sep 21  2013 beh
-rwxr-xr-x 1 root root 18976 Jan 15  2014 hp
-rwx------ 1 root root  9067 Jan 15  2014 hpfax
lrwxrwxrwx 1 root root     3 Mar 29 22:15 http -> ipp
lrwxrwxrwx 1 root root     3 Mar 29 22:15 https -> ipp
-rwx------ 1 root root 68912 Mar  4 04:46 ipp
lrwxrwxrwx 1 root root     3 Mar 29 22:15 ipps -> ipp
-rwx------ 1 root root 43928 Mar  4 04:46 lpd
-r-xr-xr-x 1 root root 35640 Mar  4 04:46 parallel
-r-xr-xr-x 1 root root 35624 Mar  4 04:46 serial
lrwxrwxrwx 1 root root    17 Mar 29 22:16 smb -> /usr/bin/smbspool
-r-xr-xr-x 1 root root 27368 Mar  4 04:46 snmp
-r-xr-xr-x 1 root root 35616 Mar  4 04:46 socket
-r-xr-xr-x 1 root root 35624 Mar  4 04:46 usb
```

Backends with world-execute run as the CUPS user, otherwise backends run as root.

Enumeration

- Called with no arguments
  - When you did lpinfo -v
  - Add a printer in the web UI.

```
if len(argv) > 5:
    ....
else:
    print('network cups-to-ogo '
        '\"Generic Postscript Printer Foomatic/Postscript (en)\" '
        '\"OpenGroupware\"')
    sys.exit(0)
```

- Output a whitespace delimited string describing the backend.
- Exit with a status of 0

Description

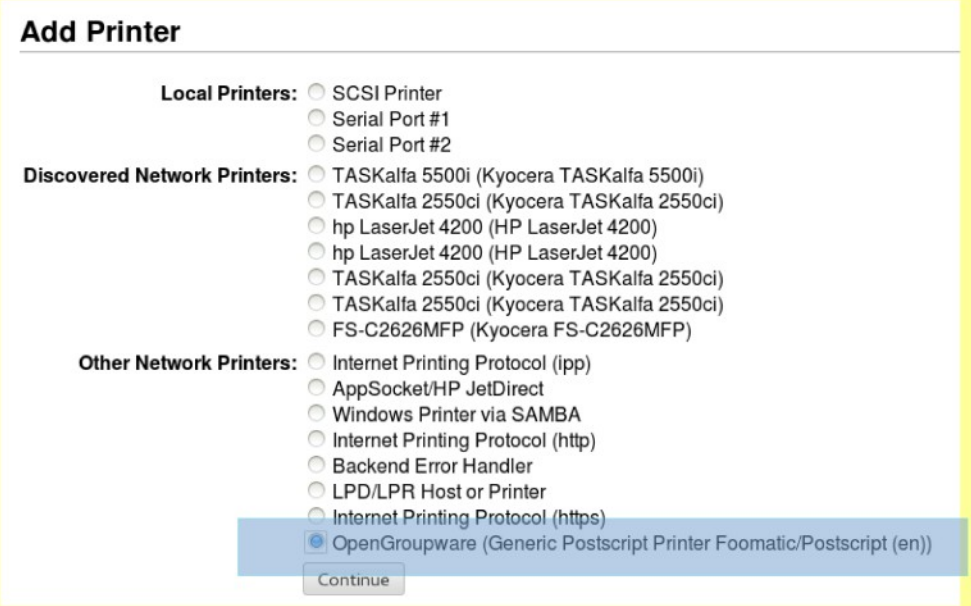
type backend-name default-printer-type display-name

- **type** = network, this is 2014
- **backend** = Backend name, the name of the script.
- **default-printer-type** – The rendering type of the printer.
- **display-name** = Display name for the administrator of the backend.

White-space separated values – so if your values contain whitespace they need to be quoted.

```
print(
    'network cups-to-ogo '
    '\"Generic Postscript Printer Foomatic/Postscript (en)\" '
    '\"OpenGroupware\"'
)
```

Now you see it.



You can create a printer

Add Printer

Connection: cups-to-ogo

Examples:

```
http://hostname:631/ipp/
http://hostname:631/ipp/port1
```

ogo_mailback (Idle, Accepting Jobs, Shared)

Maintenance | Administration

Description: Submit to MailBack Workflow

Location: MorCloud

Driver: Generic PostScript Printer Foomatic/Postscript (recommended) (color, 2-sided printing)

Connection: cups-to-ogo://dav/Workflow/Routes/TEST_MailBack/input?subject=${title}

Defaults: job-sheets=none, none media=na_letter_8.5x11in sides=one-sided

Jobs

But how does it do stuff?

```
if len(argv) > 5:
    job_id = argv[1]
    user_id = argv[2]
    title = argv[3]
    copies = argv[4]
    options = argv[5]
    if len(argv) == 7:
        filename = argv[6]
    else:
        filename = None
else:
```

If you get command arguments then it is a print job.

- If you have a seventh argument it is the file containing the job.
- If there is no filename then the data is coming in on standard-in.
  - This is what always happens.

Parameters?

CUPS pushes the queue's configured URI, including the backend prefix, into the processes environment.

```
device_uri = os.getenv('DEVICE_URI')
```

cups-to-ogo://dav/Workflow/Routes/TEST_MailBack/input?subject=${title}

Use of any any innumerable URI/URL parsing libraries.

```
pu = urlparse.urlparse(device_uri)
```

Standard In

Most of the time your job data will arrive via standard in; it is recommended to read this into a temporary file prior to processing.

```
rfile = TemporaryFile()
shutil.copyfileobj(sys.stdin, rfile)
rfile.seek(0)
do some stuff
rfile.close()
```

Response

- Status of job is exit code of backend:
  - 0 – Success / Complete
  - 1 – Failed
  - 2 – Authentication Required
  - 3 – Hold
  - 4 – Stop
  - 5 – Cancel
  - 6 – Retry

For Example

```
response = connection.getresponse()
syslog.syslog(
      syslog.LOG_INFO,
      'HTTP Result: {0}'.format(response.status, )
)
result = 1 # Backend failed, generic
if response.status in (200, 201, 202, 204, 301, ):
    result = 0  # Backend success
elif response.status in (401, 419, 511, 496, 496, ):
    result = 2  # Authentication Failed
elif response.status in (423, 409, 408, 429, ):
    result = 6  # Retry
elif response.status in (418, 451, ):
    result = 5  # Cancel Job, Teapot or illegal response
connection.close()
return result
```