Scanning in PAPPL

By-
Bhavna Kosta
Indian Institute of Technology, Mandi

Discovery of Scanner

Devices are discovered by DNS-SD. There are records with appropriate service types for scanning and also the "scan=T" in the TXT record of the printing part. So for discovery, the IPP Scan specification defines a new DNS-SD sub-type ("_scan"), which means that an IPP client interested in scanning can browse for services of type "_scan._sub._ipp._tcp". The TXT record optionally contains an "rs" key that provides the resource path for the scan service endpoint. The presence of a "scan" key is a holdover from the Apple Bonjour Printing specification but is *not* an indication of IPP Scan support (just that the printer supports scanning in some way). The HP eSCL protocol provides a separate service advertisement using the "_uscan._tcp" DNS-SD service type. Due to AirPrint and Mopria requirements, this is the protocol that almost all MFPs support. The TXT keys and values are all very similar to the IPP ones and allow for correlation with the print service (service names and UUIDs are always the same).

Pairing of Scanner

Adding support for pairing scanners with printers, since the typical use case (multi-function printer) will have the scan-specific TXT keys added to the printer, and the printer-dns-sd-name value coming from the printer. IPP scanners generally will not have their own DNS-SD records since they are paired with IPP printers. IPP scanner registrations don't use the same TXT keys as printers. Scan-specific keys are added as IPP scanner registration consists, for the pairing API to associate the scanner with the printer.

The pappl_printer_t and pappl_scanner_t structures would have pointers to their respective counterparts.
The API would just be:
>>>>> extern void papplPrinterSetScanner(pappl_scanner_t *scanner, pappl_printer_t *printer) _PAPPL_PUBLIC;
and then the DNS-SD code would look for a non-NULL pointer to know whether to do a separate advertisement or just tack the scanner onto the printer's TXT record.

WRT adding the scanner TXT keys to the printer registration:
>>>> static _pappl_txt_t
>>>> add_scanner_values(_pappl_txt_t txt, pappl_scanner_t *scanner)
then call this function from _papplScannerRegisterDNSSDNoLock.

Then the logic in papplPrinterSetScanner should be something like:

```
>>> IF printer and scanner are already paired THEN
>>>     RETURN
>>>
>>> IF printer is already paired to a scanner THEN
>>>     unpair printer and scanner
>>>     register scanner as a standalone service
>>>
>>> IF scanner provided THEN
>>>     pair printer and scanner
>>>     unregister scanner
>>>
>>> re-register printer
```

Scanner Properties

The client polls the scanner's properties with a get-printer-attributes IPP request on the scanner URI (for IPP Scan) or with an appropriate command for eSCL. The Scanner Application knows the properties of the scanner it is serving for (or if it uses SANE, it asks the SANE backend) and answers appropriately to the client's request. To expand on this, there are a couple core options (attributes in IPP, elements in eSCL) that are important:

- "document-format-accepted" which controls the format of the scanned data - the Client provides one or more formats that it understands
- "input-attributes" which controls what you want scanned; this consists of several member attributes providing the color mode, scan area/media, resolution, and source, among other more esoteric settings.

For this pappl_scanner_t object is implemented and scan-specific header files are added with the updated attributes and capabilities of a scanner- changing print to input,

- No supplies for scanners.
- The LPD and AppSocket services aren't exposed for scanners.
- No raw or USB gadget stuff, either.
- No raw listeners or USB for scanning.
- No print-content-optimize for scanning.
- No identify-actions for scanners.

and equivalent driver functions are added for scan, as that in printing.

The Scan Job

Then the client sees a scanner it can scan on. The user sets options like scan area, resolution, quality, color, ADF mode, ... and requests the scan. The client sends appropriate commands (eSCL or IPP Scan) to the Scanner Application. If the Scanner Application uses SANE it calls the appropriate SANE backend with the user's option settings to obtain the scan from the physical scanner. It answers the client's request together with streaming back the scanned image data, converted using a filter function (can be of cups-filters) in case the format returned by the SANE backend is not a standard format of eSCL or IPP Scan.

The protocol sequence in IPP is:
- The Client sends a Create-Job request with "document-format-accepted" and "input-attributes" to start scanning (no other Client can scan until the job is completed)
- The scanner returns a "job-id" value in the response
- The Client then sends one or more Get-Next-Document-Data requests to read the scan data back.
- Depending on the format of the data, the scanner either returns all of the pages/images in the scan (this happens when scanning to PDF) or the current page/image (scan to JPEG or PNG).
- When all pages/images are scanned, the response to the Get-Next-Document-Data request will contain the "last-document" operation attribute with a value of 'true'.

Thank-you