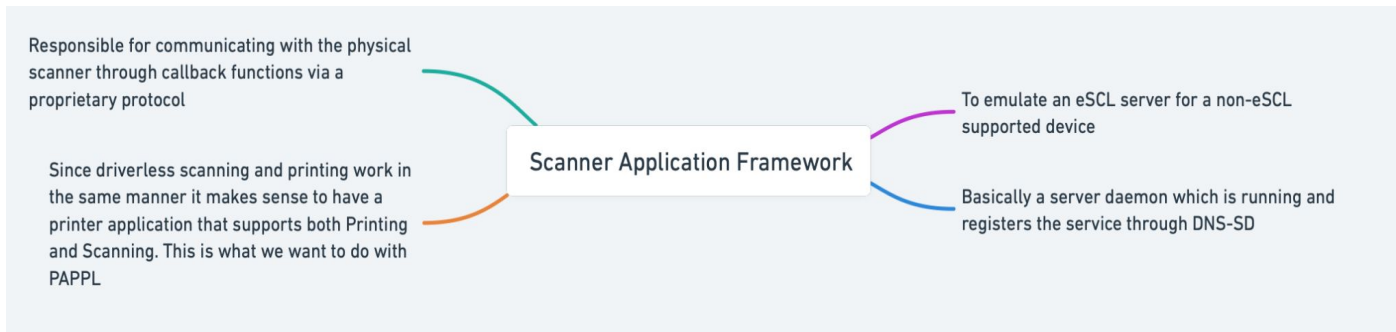
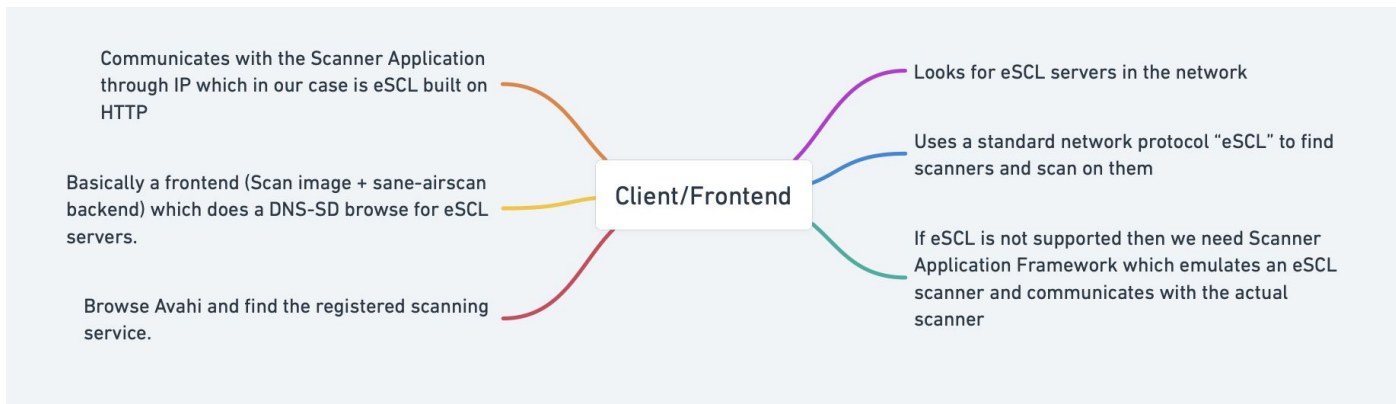


SCAN API BRIDGING IN PAPPL

GSoC '23 and '24

THE ARCHITECTURE



CREATING ESCL ENDPOINTS

- Go through the MOPRIA Scan Specifications to know the endpoints.
- HTTPS/eSCL endpoints needed for
 - 1) `/{{root}}/ScannerCapabilities`
 - 2) `/{{root}}/ScannerStatus`
 - 3) `/{{root}}/ScanBufferInfo`
 - 4) `/{{root}}/ScanJobs` and many more

ESCL ENDPOINTS

Implemented under pappl/
client.c under
HTTP_STATE_GET and
HTTP_STATE_POST

```
case HTTP_STATE_POST:
    if (strcmp(client->uri, "/eSCL/ScanJobs") == 0)
    {
        // Process the ScanJobs request
        size_t content_length = (size_t)httpGetLength2(client->http);
        char *xml_content = (char *)malloc(content_length + 1);
```

```
    if (strcmp(client->uri, "/eSCL/ScanJobs/"))
        // Check if the requested URI is "/eSCL/ScannerCapabilities"
        if (strcmp(client->uri, "/eSCL/ScannerCapabilities") == 0)
        {
            addOrUpdateIP(&ipList, client->host_field);
```

```
case HTTP_STATE_GET:
    const char *dummyFilePath = "NULL";
    addOrUpdateIP(&ipList, client->host_field);

    // Check if the requested URI is "/eSCL/ScanJobs/JobUri/NextDocument"
    if (isMatchingRequest(client->uri) == 1)
    {
        dummyFilePath = "/DummyDriver/pappl-160.png";

        // Send an external file...
        int fd;           // Resource file descriptor
        char buffer[8192]; // Copy buffer
        ssize_t bytes;     // Bytes read/written
```

CREATING XML PARSER

- Since eSCL communications is essentially through XML we would have to create a XML Parser that can sort of interpret various details and call the required functions.
- The XML parser is based on regex matching done using patterns as demonstrated in MOPRIA Scan Specifications.
- Divided the main function into many branched out functions for micro features.

XML PARSER

Interpret eSCL
communications

```
//  
// Functions...  
//  
  
char *readXmlContent(const char *filePath);  
  
void initScanSettingsXml(ScanSettingsXml *settings, const char *s);  
  
char *getString(const ScanSettingsXml *settings, const char *name, const char *pattern);  
  
double getNumber(const ScanSettingsXml *settings, const char *name, const char *pattern);  
  
bool ClientAlreadyAirScan(pappl_client_t *client);  
  
ScanSettingsXml *ScanSettingsFromXML(const char *xmlString, pappl_client_t *client);
```

```
ScanSettingsXml *ScanSettingsFromXML(  
    const char *xmlString, // I - XML string containing scan settings  
    pappl_client_t *client) // I - Pointer to the pappl_client_t structure  
{  
    ScanSettingsXml scanSettings;  
    initScanSettingsXml(&scanSettings, xmlString);  
  
    char *versionPattern = "<pwg:Version>([^<]*)</pwg:Version>";  
    char *version = getString(&scanSettings, "Version", versionPattern);  
  
    char *intentPattern = "<scan:Intent>([^<]*)</scan:Intent>";  
    char *intent = getString(&scanSettings, "Intent", intentPattern);  
  
    char *heightPattern = "<pwg:Height>([^<]*)</pwg:Height>";  
    char *height = getString(&scanSettings, "Height", heightPattern);  
  
    char *contentRegionUnitsPattern = "<cpwg:ContentRegionUnits>([^<]*)</cpwg:ContentRegionUnits>";  
    char *contentRegionUnits = getString(&scanSettings, "ContentRegionUnits", contentRegionUnitsPattern);  
  
    char *widthPattern = "<pwg:Width>([^<]*)</pwg:Width>";  
    double width = getNumber(&scanSettings, "Width", widthPattern);  
  
    char *xOffsetPattern = "<pwg:XOffset>([^<]*)</pwg:XOffset>";  
    double xOffset = getNumber(&scanSettings, "XOffset", xOffsetPattern);  
  
    char *yOffsetPattern = "<pwg:YOffset>([^<]*)</pwg:YOffset>";  
    double yOffset = getNumber(&scanSettings, "YOffset", yOffsetPattern);  
  
    char *inputSourcePattern = "<cpwg:InputSource>([^<]*)</cpwg:InputSource>";  
    char *inputSource = getString(&scanSettings, "InputSource", inputSourcePattern);  
  
    char *colorModePattern = "<scan:ColorMode>([^<]*)</scan:ColorMode>";  
    char *colorMode = getString(&scanSettings, "ColorMode", colorModePattern);
```

DUMMY DRIVER FOR EMULATION

- Add a dummy driver (initially xml files, later turned to text files) that emulates a driver fetching data from a scanner.
- The following scan driver emulation files had to be added:
 - 1) ScannerStatus
 - 2) ScannerCapabilites
 - 3) ScannerBufferInfo

DUMMY DRIVER

Example of Initial Buffer
Info file

```
<?xml version="1.0" encoding="UTF-8"?>
<scan:ScanBufferInfo
xsi:schemaLocation="http://schemas.hp.com/imaging/escl/20
11/05/03 escl.xsd"
xmlns:scan="http://schemas.hp.com/imaging/escl/2011/05/03
"
xmlns:httpdest="http://schemas.hp.com/imaging/httpdestina
tion/2011/10/13"
xmlns:pwg="http://www.pwg.org/schemas/2010/12/sm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<scan:ScanSettings>
<pwg:Version>2.6</pwg:Version>
<scan:Intent>Photo</scan:Intent>
<pwg:ScanRegions>
<pwg:ScanRegion>
<pwg:Height>1200</pwg:Height>
<pwg:ContentRegionUnits>escl:ThreeHundredthsOfInches</p
wg:ContentRegionUnits>
<pwg:Width>1800</pwg:Width>
<pwg:XOffset>0</pwg:XOffset>
<pwg:YOffset>0</pwg:YOffset>
</pwg:ScanRegion> </pwg:ScanRegions>
<scan:DocumentFormatExt>image/jpeg</scan:DocumentFormat
Ext>
<pwg:ContentType>Photo</pwg:ContentType>
<pwg:InputSource>Platen</pwg:InputSource>
<scan:XResolution>300</scan:XResolution>
<scan:YResolution>300</scan:YResolution>
<scan:ColorMode>Grayscale8</scan:ColorMode>
```


UPSTREAM
MOVEMENT OF API
BRIDGING

UPSTREAMING

- Addition of `pwg-scanner.c` in the `testsuite` directory,
- Updating `testpappl.c` to add a "scan" test to validate scanning.
- Update `testmainloop.c` to call the new scanner APIs prior to `papplMainloop`.
- Update the `Pappl-Retrofit` repository to retrofit the SANE Drivers.
- Final Upstreaming should finish by the end of OSPP Open Source Program.

PRESENTATION OF SCAN API PROJECT AT VARIOUS CONFERENCES

- 1) Ubuntu Summit 2023
- 2) Opportunity Open Source, India 2023 and 2024
- 3) FOSDEM 2025

