scanservjs (Scanner Web-UI)

```
version: "3.8"

services:
  scanservjs:
    image: sbs20/scanservjs:v3.0.4
    container_name: scanservjs
    privileged: true
    #network_mode: host #BENOETIGT BEIM SUCHEN MIT 'airscan-discover fuer eSCL und WSD
Scanner'
    ports:
      - "8098:8080"
    environment:
      #DELIMITER: '|'
      #SANED_NET_HOSTS: "10.0.100.30;10.0.100.31"
      #AIRSCAN_DEVICES: '"HWS PM1 ∗□□" = "http://192.168.200.18/eSCL"|"HWS PM2 ∗□□" =
"http://192.168.200.15/eSCL"|"HWS 1.OG ∗□□□ = "http://192.168.200.119/eSCL"'
      #AIRSCAN_DEVICES='"Canon MFD" = "http://192.168.0.10/eSCL";"EPSON MFD" =
"http://192.168.0.11/eSCL"'
      #AIRSCAN_DEVICES: |
      #  "Canon MFD" = "http://192.168.0.10/eSCL";
      #  "EPSON MFD" = "http://192.168.0.11/eSCL"
      #PIXMA_HOSTS: "192.168.200.33"
      #SCANIMAGE_LIST_IGNORE: "true"
      #DEVICES:
"net:10.0.100.30:plustek:libusb:001:003;net:10.0.100.31:plustek:libusb:001:003;airscan:e0:Cano
n TR8500 series;airscan:e1:EPSON Cool Series"
      OCR_LANG: "deu+eng"
    volumes:
      - /var/run/dbus:/var/run/dbus
      - scans:/var/lib/scanservjs/output
      - scans-backup:/var/lib/scanservjs/output-backup
      - cfg:/etc/scanservjs
      - saned:/etc/sane.d
    restart: unless-stopped
```

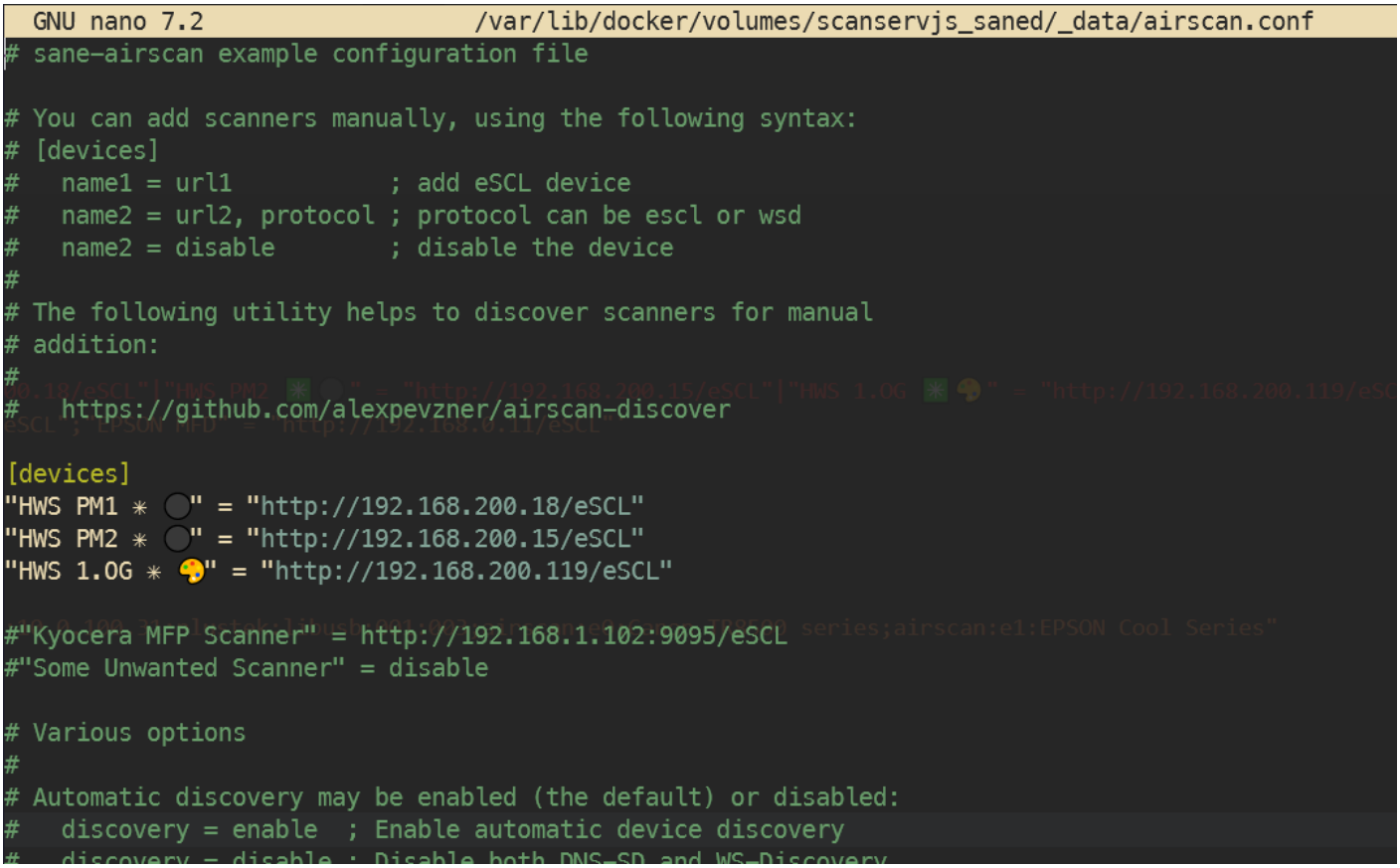
```
volumes:
  scans:
  scans-backup:
  cfg:
  saned:
```

Problem: Es werden immer nur die ersten Trennzeichen erkannt, das heißt Scanner PM2 und 1.OG fallen unter ein Device, siehe Log:

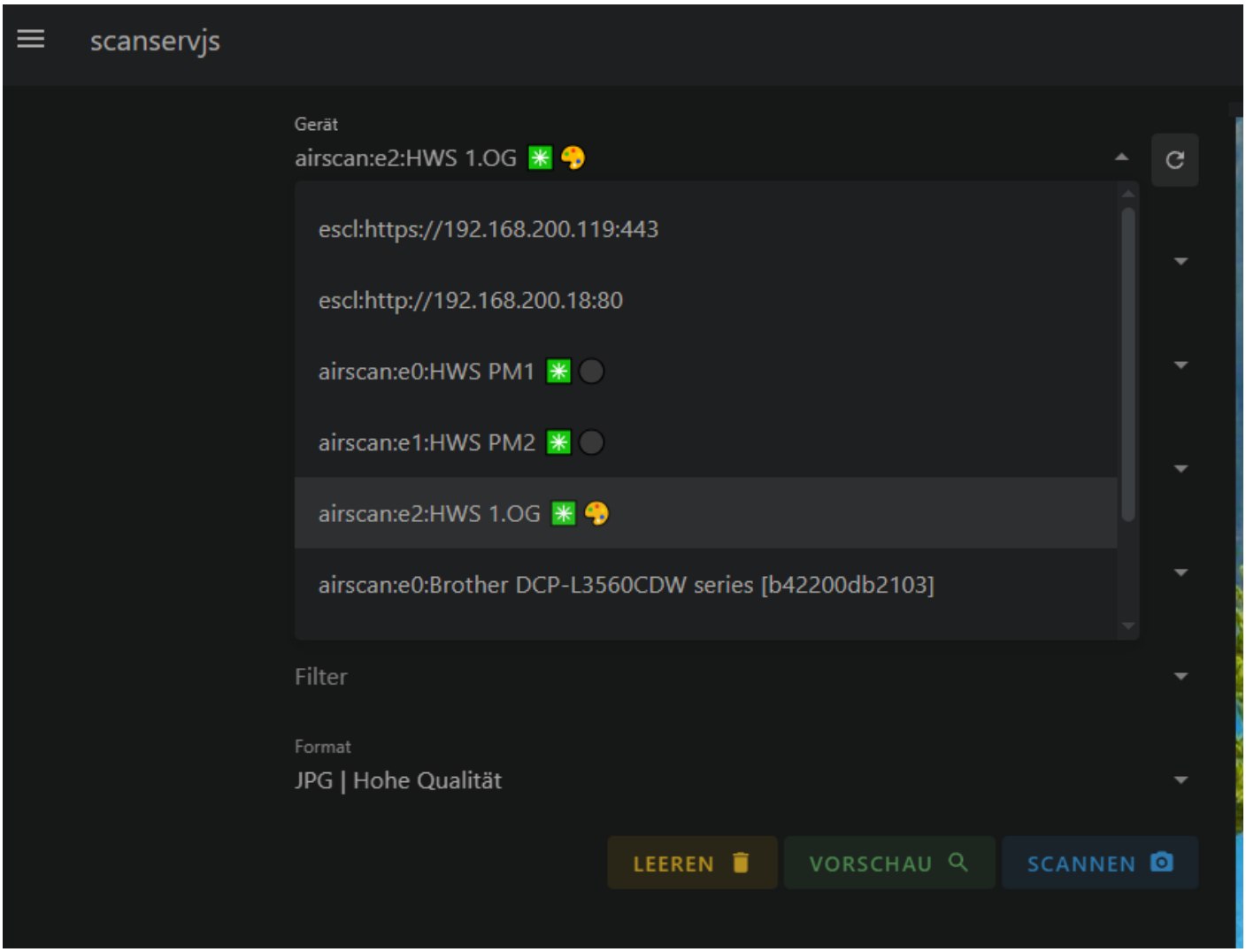
```
# Insert airscan devices
if [ ! -z "$AIRSCAN_DEVICES" ]; then
  devices=$(echo $AIRSCAN_DEVICES | sed "s/$DELIMITER/\n/")
  for device in $devices; do
    sed -i "/^\[devices\]/a $device" /etc/sane.d/airscan.conf
  done
fi
+ [ ! -z "HWS PM1 ∗□□" = "http://192.168.200.18/eSCL"|"HWS PM2 ∗□□" =
"http://192.168.200.15/eSCL"|"HWS 1.OG ∗□□□ = "http://192.168.200.119/eSCL" ]
+ echo "HWS PM1 ∗□□" = "http://192.168.200.18/eSCL"|"HWS PM2 ∗□□" =
"http://192.168.200.15/eSCL"|"HWS 1.OG ∗□□□ = "http://192.168.200.119/eSCL"
+ sed s/|/\n/
+ devices="HWS PM1 ∗□□" = "http://192.168.200.18/eSCL"
"HWS PM2 ∗□□" = "http://192.168.200.15/eSCL"|"HWS 1.OG ∗□□□ = "http://192.168.200.119/eSCL"
+ sed -i /^\[devices\]/a "HWS PM1 ∗□□" = "http://192.168.200.18/eSCL" /etc/sane.d/airscan.conf
+ sed -i /^\[devices\]/a "HWS PM2 ∗□□" = "http://192.168.200.15/eSCL"|"HWS 1.OG ∗□□□ =
"http://192.168.200.119/eSCL" /etc/sane.d/airscan.conf
# Insert pixma hosts
if [ ! -z "$PIXMA_HOSTS" ]; then
  hosts=$(echo $PIXMA_HOSTS | sed "s/$DELIMITER/\n/")
  for host in $hosts; do
    echo "bjnp://$host" >> /etc/sane.d/pixma.conf
```

Lösung: wie oben im docker-compose, den Ordner /etc/sane.d als Volume mounten und die airscan.conf darin per Hand bearbeiten:

```
[devices]
"HWS PM1 ∗□□" = "http://192.168.200.30/eSCL"
"HWS PM1 □□□ = "http://192.168.200.31:80/wsd/scanservice.cgi", wsd
"HWS PM2 ∗□□" = "http://192.168.200.32/eSCL"
"HWS PM3 □□□ = "http://192.168.200.33:80/wsd/scanservice.cgi", wsd
"HWS 1.OG ∗□□□□□□□□ "http://192.168.200.34/eSCL"
```


GNU nano 7.2 /var/lib/docker/volumes/scanservjs_saned/_data/airscan.conf
sane-airscan example configuration file
You can add scanners manually, using the following syntax:
[devices]
name1 = url1 ; add eSCL device
name2 = url2, protocol ; protocol can be escl or wsd
name2 = disable ; disable the device
The following utility helps to discover scanners for manual
addition:
https://github.com/alexpevzner/airscan-discover
[devices]
"HWS PM1 ✳ ⚫" = "http://192.168.200.18/eSCL"
"HWS PM2 ✳ ⚫" = "http://192.168.200.15/eSCL"
"HWS 1.0G ✳ 🐠" = "http://192.168.200.119/eSCL"
#"Kyocera MFP Scanner" = http://192.168.1.102:9095/eSCL
#"Some Unwanted Scanner" = disable
Various options
Automatic discovery may be enabled (the default) or disabled:
discovery = enable ; Enable automatic device discovery

>



Testen, ob eSCL funktioniert

URL des Druckers mit Anhang aufrufen:

http://<DRUCKER-IP>/eSCL/ScannerCapabilities

http://<DRUCKER-IP>/eSCL/ScannerStatus

zb http://10.53.1.20/eSCL/ScannerCapabilities

WSD-Scanner suchen

In Docker Container execen und

```
airscan-discover
```

ausfuehren.

ACHTUNG! Zum discovern eines zb Canon Pixma GM4050 bzw GM4000 series muss network_mode: host im docker-compose.yml aktiviert sein. Danach funktioniert das scannen aber auch ohne network_mode: host

```
[...]
    container_name: scanservjs
    privileged: true
    network_mode: host
    ports:
[...]
```

```
> airscan-discover
[devices]
  Brother DCP-L3560CDW series [b42200db2103] = http://192.168.200.34:80/eSCL/, eSCL
  Brother DCP-L3560CDW series [b42200db2103] = https://192.168.200.34:443/eSCL/, eSCL
  Brother DCP-L3560CDW series [b42200db2103] =
http://192.168.200.34:80/WebServices/ScannerService, WSD
  Brother DCP-L3560CDW series [b42200db2103] =
http://[FE80::7697:79FF:FEEA:4635%252]:80/WebServices/ScannerService, WSD
  Brother MFC-L2710DW series = http://192.168.200.32:80/eSCL/, eSCL
  Brother MFC-L2710DW series = http://192.168.200.32:80/WebServices/ScannerService, WSD
  Brother MFC-L2710DW series =
http://[FE80::4ED5:77FF:FE6D:67EF%252]:80/WebServices/ScannerService, WSD
  Brother MFC-L2750DW series = http://192.168.200.30:80/eSCL/, eSCL
  Brother MFC-L2750DW series = http://192.168.200.30:80/WebServices/ScannerService, WSD
  Brother MFC-L2750DW series =
http://[FE80::BEF4:D4FF:FE44:75D%252]:80/WebServices/ScannerService, WSD
  CANON INC. GM4000 series = http://192.168.200.33:80/wsd/scanservice.cgi, WSD
  CANON INC. GM4000 series = http://[fe80::6e3c:7cff:fe0e:c3da%252]:80/wsd/scanservice.cgi,
WSD
  CANON INC. GM4000 series = http://192.168.200.31:80/wsd/scanservice.cgi, WSD
  CANON INC. GM4000 series = http://[fe80::6e3c:7cff:fe0f:db71%252]:80/wsd/scanservice.cgi,
WSD
```

```
root@pl-mini-01-hws:/usr/lib/scanservjs# airscan-discover
[devices]
  Brother DCP-L3560CDW series [b42200db2103] = http://192.168.200.34:80/eSCL/, e
SCL
  Brother DCP-L3560CDW series [b42200db2103] = https://192.168.200.34:443/eSCL/,
 eSCL
  Brother DCP-L3560CDW series [b42200db2103] = http://192.168.200.34:80/WebServi
ces/ScannerService, WSD
  Brother DCP-L3560CDW series [b42200db2103] = http://[FE80::7697:79FF:FEEA:4635
%252]:80/WebServices/ScannerService, WSD
  Brother MFC-L2710DW series = http://192.168.200.32:80/eSCL/, eSCL
  Brother MFC-L2710DW series = http://192.168.200.32:80/WebServices/ScannerServi
ce, WSD
  Brother MFC-L2710DW series = http://[FE80::4ED5:77FF:FE6D:67EF%252]:80/WebServ
ices/ScannerService, WSD
```

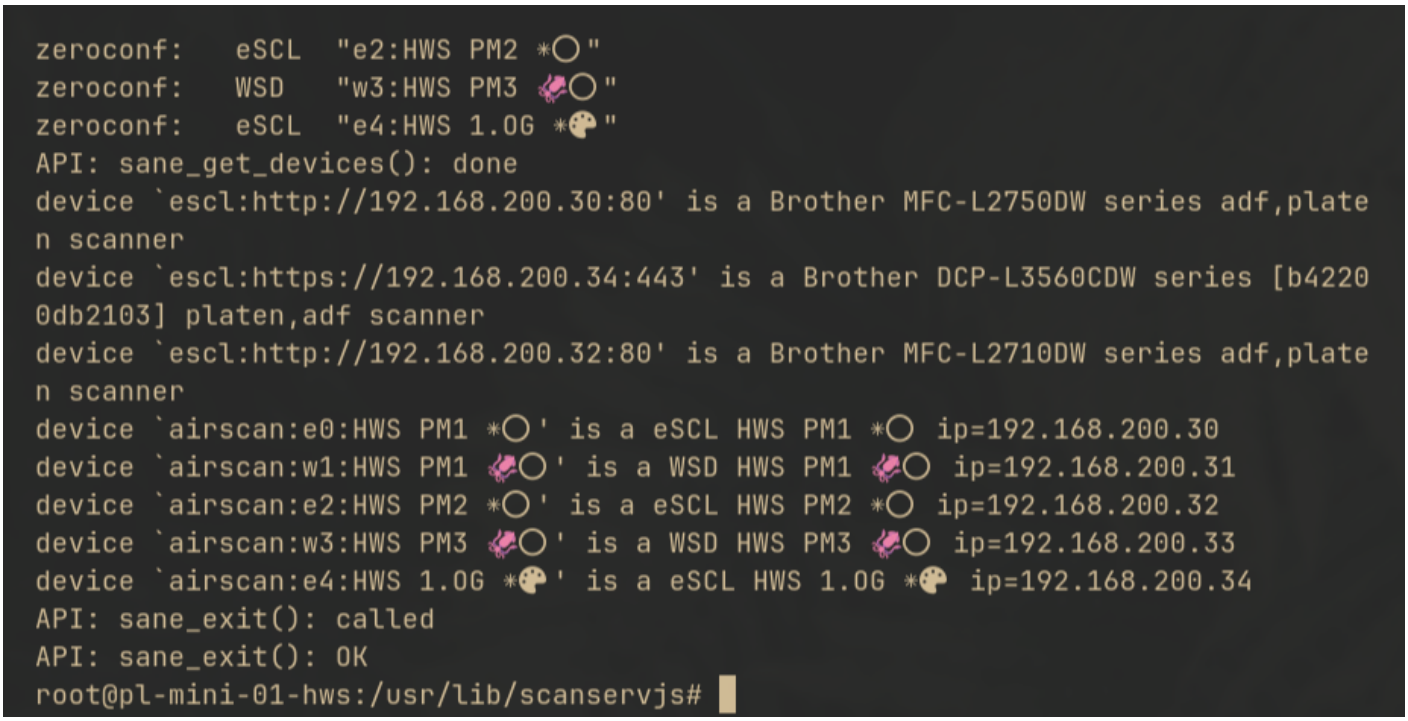
danach die sane.d/airscan.conf anpassen:

```
      /var/lib/docker/volumes/scanservjs_saned/_data/airscan.conf *
#   name2 = disable        ; disable the device
#
# The following utility helps to discover scanners for manual
# addition:
#
#   https://github.com/alexpevzner/airscan-discover

[devices]
"HWS PM1 *◯" = "http://192.168.200.30/eSCL"
"HWS PM1 🦑◯" = "http://192.168.200.31:80/wsd/scanservice.cgi", wsd
"HWS PM2 *◯" = "http://192.168.200.32/eSCL"
"HWS PM3 🦑◯" = "http://192.168.200.33:80/wsd/scanservice.cgi", wsd
"HWS 1.OG *🎨" = "http://192.168.200.34/eSCL"
```

und

```
scanimage -L
```

eingeben:



config anpassen und in config.local.js umbenennen:

Eikes Beispiel:

```
[...]
  afterDevices(devices) {
    const deviceNames = {
      'airscan:e0:FBS Friedrich UG ✳□|□□□□□□:□□'FBS Friedrich UG ✳□|□□□□□□,□□
      'airscan:e1:FBS EG-PM ✳□|□': 'FBS EG-PM ✳□|□',
      'airscan:e2:FBS Friedrich I. ✳□|□□□□□□:□□'FBS Friedrich I. ✳□|□□□□□□,□□
      'airscan:e3:FBS Friedrich II. ✳□|□□□□□□:□□'FBS Friedrich II. ✳□|□□□□□□,□□
      'airscan:e4:FBS Friedrich III. ✳□|□□□□□□:□□'FBS Friedrich III. ✳□|□□□□□□□□
    };

    devices
      .filter(d => d.id in deviceNames)
      .forEach(d => d.name = deviceNames[d.id]);

    devices
      .filter(d => d.id.includes('FBS'))
      .forEach(device => {
        device.features['--mode'].default = 'Color';
        device.features['--resolution'].default = 150;
        device.features['--resolution'].options = [75, 150, 300, 600];
        device.features['--brightness'].default = 0;
```

```
        device.features['--contrast'].default = 5;
        device.features['-x'].default = 192;
        device.features['-y'].default = 272;
        /* device.settings.pipeline.default = 'PNG'; */
        device.settings.pipeline.default = '@:pipeline.ocr | PDF (JPG | @:pipeline.high-
quality)';
        /* Disable batch modes if they are not available on your printer: */
        /*device.settings.batchMode.options = ['none', 'manual']; */
    });
  },

  async afterScan(fileInfo) {
[...]
```

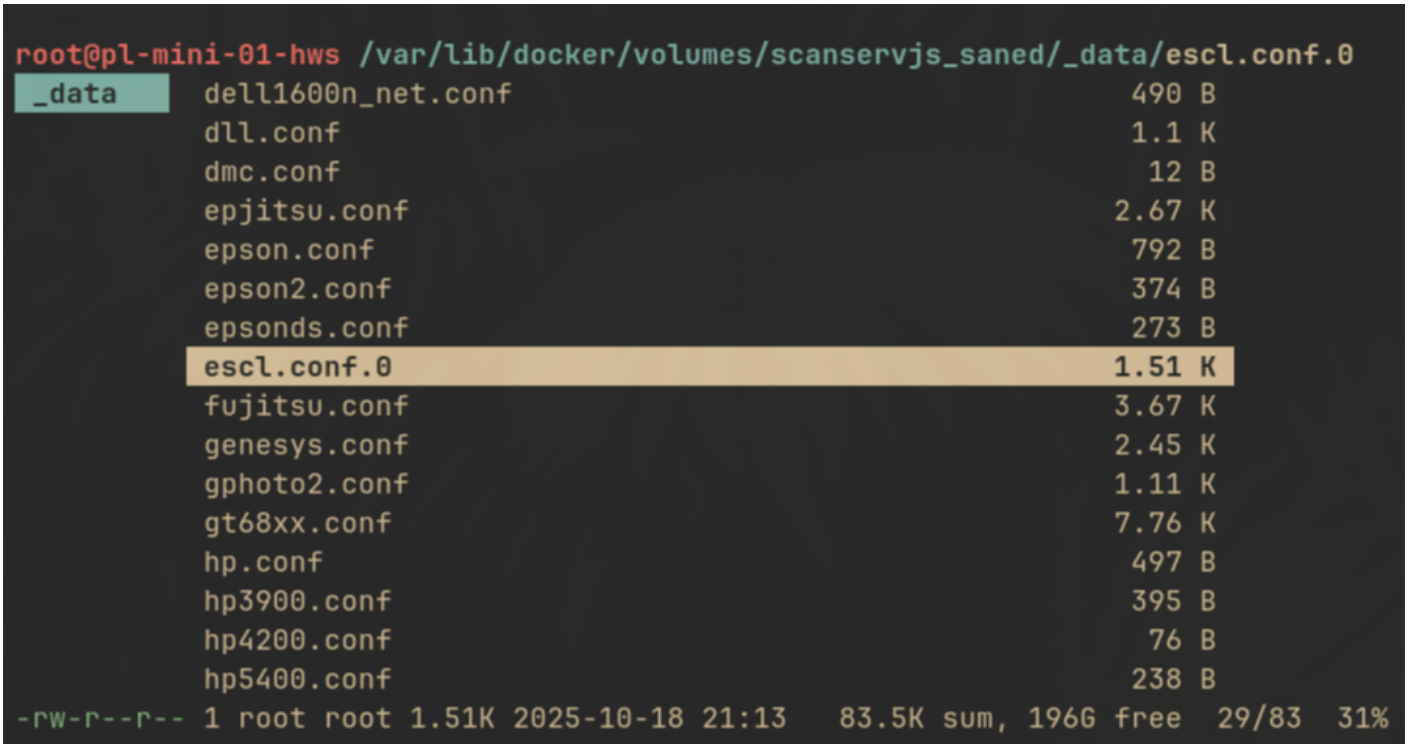
https://medium.com/@davidclaeys/scanservjs-make-your-own-scan-server-21539f64265c

```
module.exports = {
  afterDevices(devices) {
 const deviceNames = {
      /*
        'device id':'device name'
      */
      'airscan:e0:Hp Envy Pro 6442': 'Hp Envy Pro 6442'
    };

/*
  replace the id in the filter
*/
 devices
      .filter(d => d.id == 'airscan:e0:Hp Envy Pro 6442')
      .forEach(device => {
        device.features['--resolution'].default = 400;
        device.features['--resolution'].options = [100, 150, 200, 300, 400, 600];
        /*
          Disable batch modes if they are not available on your printer
        */
         device.settings.batchMode.options = ['none', 'manual'];
```

```
        /*
        Specify the default pipeline
        */
        device.settings.pipeline.default = ['PNG'];
    });

  devices
      .filter(d => d.id in deviceNames)
      .forEach(d => d.name = deviceNames[d.id]);
  }
};
```

escl:// automatische Suche deaktivieren

escl.conf umbenennen:



Scans autom. backupen und löschen

Weil die Scans von jedem zugänglich sind, der die Seite aufrufen kann, sollten die Scans gelöscht und zusätzlich gebackupt werden.
Wie oben in der docker-compose.yml beschrieben, ein Volume scans-backup anlegen. Dazu dieses Skript zb unter

/home/user/scanservjs-backup.sh

anlegen und mit `chmod +x /home/user/scanservjs-backup.sh` ausführbar machen:

```
#!/bin/bash

# Scanservjs Backup und Bereinigungsskript
# Erstellt Backups und löscht alte Dateien (älter als 60 Minuten)

# Verzeichnisse definieren
SOURCE_DIR="/var/lib/docker/volumes/scanservjs_scans"
BACKUP_DIR="/var/lib/docker/volumes/scanservjs_scans-backup"
LOG_FILE="/var/log/scanservjs-backup.log"

# Zeitstempel für Logging
TIMESTAMP=$(date '+%Y-%m-%d %H:%M:%S')

# Logging-Funktion
log_message() {
    echo "[$TIMESTAMP] $1" | tee -a "$LOG_FILE"
}

# Skript starten
log_message "=== Backup-Skript gestartet ==="

# Prüfen ob Quellverzeichnis existiert
if [ ! -d "$SOURCE_DIR" ]; then
    log_message "FEHLER: Quellverzeichnis $SOURCE_DIR existiert nicht!"
    exit 1
fi

# Backup-Verzeichnis erstellen falls nicht vorhanden
if [ ! -d "$BACKUP_DIR" ]; then
    mkdir -p "$BACKUP_DIR"
    log_message "Backup-Verzeichnis $BACKUP_DIR wurde erstellt"
fi

# Dateien kopieren (mit rsync für effizientes Backup)
log_message "Starte Backup von $SOURCE_DIR nach $BACKUP_DIR"
rsync -av --progress "$SOURCE_DIR/" "$BACKUP_DIR/" >> "$LOG_FILE" 2>&1
```

```
if [ $? -eq 0 ]; then
    log_message "Backup erfolgreich abgeschlossen"
else
    log_message "FEHLER beim Backup!"
    exit 1
fi

# Alte Dateien löschen (älter als 60 Minuten)
log_message "Suche nach Dateien älter als 60 Minuten in $SOURCE_DIR"

# Zähler für gelöschte Dateien
DELETED_COUNT=0

# Dateien finden und löschen
while IFS= read -r -d '' file; do
    log_message "Lösche: $file"
    rm -f "$file"
    if [ $? -eq 0 ]; then
        ((DELETED_COUNT++))
    else
        log_message "FEHLER beim Löschen von: $file"
    fi
done < <(find "$SOURCE_DIR" -type f -mmin +60 -print0)

log_message "Anzahl gelöschter Dateien: $DELETED_COUNT"
log_message "=== Backup-Skript beendet ==="
echo "" >> "$LOG_FILE"

exit 0
```

dann mit crontab -e minütlich laufen lassen:

```
* * * * * /home/user/scanservjs-backup.sh
```

Revision #12
Created 2025-10-06 19:08:02 UTC
Updated 2025-10-29 22:23:56 UTC