Scanner Epson ET-4856 — Intégration Home Assistant & Paperless-ngx

Scanner Epson ET-4856 — Intégration Home Assistant & Paperless-ngx

Mise en place : 22 mars 2026
Dernière mise à jour : 29 juillet 2026 (v4 — recto-verso, exécution détachée, bouton RAZ)

---

1. Vue d'ensemble

L'imprimante Epson ET-4856 (identifiée comme *ET-4850 Series*) est pilotée en local via le protocole **eSCL** (AirScan/AirPrint) en HTTPS, sans aucune dépendance au logiciel Epson. Un script bash exécuté depuis le container Home Assistant déclenche les scans et dépose les fichiers directement dans le répertoire `consume` de Paperless-ngx pour ingestion automatique (OCR + indexation).

Architecture

```
Bouton Dashboard HA (NSPanel)
    → script.scanner_epson[_recto|_verso|_reset]
    → shell_command.epson_scan[_recto|_verso]
```

```
    → epson_scan_launcher.sh   (détache le processus en arrière-plan)
    → epson_scan.sh            (scan eSCL, traitement PDF, notification API HA)
        → POST eSCL /ScanJobs (HTTPS:443) sur l'imprimante
        → GET /NextDocument → PDF (tout le lot ADF en un seul fichier)
        → GET /NextDocument (404) + DELETE → fermeture propre du job
        → /media/scans/ (= consume Paperless)
    → Paperless ingère → OCR → indexation → archivage
```

Composants

Composant	Rôle	Emplacement
Epson ET-4856	Scanner (eSCL via HTTPS)	192.168.10.25:443
Home Assistant	Orchestration, dashboard, déclenchement	Container Docker sur Unraid
Paperless-ngx	Ingestion, OCR, indexation	Container Docker sur Unraid
epson_scan.sh	Script principal de scan eSCL (v4)	/config/scripts/epson_scan.sh
epson_scan_launcher.sh	Wrapper de mise en arrière-plan	/config/scripts/epson_scan_launcher.sh
epson_scan_reset.sh	Réinitialisation (bourrage / blocage)	/config/scripts/epson_scan_reset.sh

---

2. Protocole eSCL — Détails techniques

Découverte

Le scanner n'expose **pas** eSCL en HTTP (port 80). Il faut utiliser **HTTPS sur le port 443** avec l'option `-k` de curl (certificat auto-signé *SEIKO EPSON CORP.*).

```
# Statut du scanner
curl -s -k https://192.168.10.25/eSCL/ScannerStatus

# Capabilities complètes
curl -s -k https://192.168.10.25/eSCL/ScannerCapabilities
```

Côté interface web de l'imprimante, il n'y a **aucune mention explicite d'eSCL** : c'est le réglage **AirPrint** qui l'active (eSCL est le volet « scan » d'AirPrint). Le paramètre « Protocole Top Priority » (IPP / port9100) ne concerne que l'impression.

Capabilities

Paramètre	Vitre (Platen)	Chargeur (ADF)
Résolutions	100, 200, 300, 600, 1200 DPI	100, 200, 300, 600 DPI
Modes couleur	RGB24, Grayscale8, BlackAndWhite1	RGB24, Grayscale8, BlackAndWhite1
Formats sortie	PDF, JPEG	PDF, JPEG
Duplex matériel	N/A	**Non supporté** (AdfSimplexInputCaps uniquement)
Capacité	1 page	50 feuilles max
Dimensions max	2550 x 3510 (≈A4)	2550 x 4200

⚠ Comportement clé de l'ET-4856 : un seul PDF par lot ADF

Contrairement à ce qu'on pourrait attendre, l'ET-4856 ne renvoie **pas** une page par appel à `NextDocument`. Un unique appel scanne **tout le chargeur ADF** et renvoie un **PDF multi-pages complet**. Toute logique de boucle « page par page » est donc inutile pour ce modèle — et c'était la cause d'un bug d'ordre des pages en recto-verso.

Séquence HTTP d'un scan

1. `POST /eSCL/ScanJobs` avec XML de paramètres → Réponse **201** + header `Location` = URL du job
2. `GET {Location}/NextDocument` → Télécharge le PDF (tout le lot)
3. `GET {Location}/NextDocument` (2^{e} appel) → Renvoie 404, signale la fin au scanner
4. `DELETE {Location}` → Libère le job, le scanner repasse en `Idle`

Sans les étapes 3 et 4, le scanner reste bloqué sur « Numérisation » à l'écran LCD.

Exemple de XML ScanSettings

```
<?xml version="1.0" encoding="UTF-8"?>
<scan:ScanSettings xmlns:pwg="http://www.pwg.org/schemas/2010/12/sm"
  xmlns:scan="http://schemas.hp.com/imaging/escl/2011/05/03">
  <pwg:Version>2.0</pwg:Version>
  <pwg:ScanRegions>
    <pwg:ScanRegion>
      <pwg:Width>2550</pwg:Width>
      <pwg:Height>3510</pwg:Height>
      <pwg:ContentRegionUnits>escl:ThreeHundredthsOfInches</pwg:ContentRegionUnits>
      <pwg:XOffset>0</pwg:XOffset>
      <pwg:YOffset>0</pwg:YOffset>
    </pwg:ScanRegion>
  </pwg:ScanRegions>
  <pwg:InputSource>Platen</pwg:InputSource>
  <scan:ColorMode>RGB24</scan:ColorMode>
  <scan:XResolution>300</scan:XResolution>
  <scan:YResolution>300</scan:YResolution>
  <pwg:DocumentFormat>application/pdf</pwg:DocumentFormat>
  <scan:Intent>Document</scan:Intent>
</scan:ScanSettings>
```

---

3. Contraintes de l'environnement HA (importantes)

3.1 — Container Alpine / BusyBox

Le container HA tourne sous Alpine Linux avec BusyBox. Les scripts évitent donc :

- `grep -P` (Perl regex, non supporté) → remplacé par `sed -n 's/.../p'`
- `stat` → remplacé par `wc -c`
- heredoc `cat << EOF` pour le XML → chaîne dans une variable

3.2 — Timeout de 60 s des shell_command

Home Assistant tue tout `shell_command` après **60 secondes**, sans possibilité de configurer ce délai. Un scan ADF de 5 feuilles recto-verso dépasse largement ce seuil : le processus était tué *avant* d'avoir pu fermer le job eSCL → **imprimante bloquée sur « Numérisation »**, et aucune notification côté HA.

3.3 — Mode restreint des shell_command avec templates

Quand un `shell_command` contient des templates Jinja2 (`{{ states(...) }}`), HA l'exécute dans un environnement restreint où les opérateurs shell `>`, `>>`, `|`, `~` et `&` **ne sont PAS interprétés** — ils sont pris littéralement. Écrire `nohup ... &` directement dans le shell_command ne fonctionne donc pas.

Solution retenue : un script *launcher* intermédiaire. Le `shell_command` appelle simplement `bash launcher.sh args` (aucun opérateur), et c'est le launcher — exécuté dans un vrai bash sans restriction — qui fait le `nohup ... &`.

3.4 — Persistance de poppler-utils

`pdfunite` et `pdfseparate` (package `poppler-utils`) sont nécessaires pour le traitement PDF. Or `apk add` ne survit pas au redémarrage du container, et l'automation d'installation au démarrage s'est révélée non fiable (timing réseau, timeout de 60 s sur un téléchargement de ~200 Mo).

Solution retenue : copier les binaires et leurs librairies dans `/config/poppler-persist/` (volume monté depuis Unraid, donc permanent), et pointer dessus via `PATH` / `LD_LIBRARY_PATH` en tête de `epson_scan.sh`.

```
# Copie initiale (à refaire uniquement si /config est réinitialisé)
docker exec homeassistant sh -c '
mkdir -p /config/poppler-persist
for pkg in poppler-utils poppler cairo pixman nss nspr libxrender openjpeg libheif-openjpeg;
do
  apk info -L $pkg 2>/dev/null | while read f; do
    if [ -f "/$f" ]; then
      mkdir -p "/config/poppler-persist/$(dirname $f)"
      cp -n "/$f" "/config/poppler-persist/$f" 2>/dev/null
    fi
  done
done
find /config/poppler-persist -name "pdfunite" -o -name "pdfseparate"
```

```
'
```

---

4. Configuration du container Home Assistant

Volume monté

Le répertoire `consume` de Paperless est monté dans le container HA :

```
-v '/mnt/user/Sauvegarde_Jean-Michel/Documents_Importants/consume/':'/media/scans':'rw'
```

Jeton d'accès API Home Assistant

Comme le script tourne détaché, il notifie lui-même HA via l'API REST locale. Un **jeton d'accès longue durée** (créé depuis le profil utilisateur HA) est stocké dans `/config/.ha_token` :

```
echo "LE_JETON" > /mnt/user/appdata/homeassistant/.ha_token
chmod 600 /mnt/user/appdata/homeassistant/.ha_token

# Vérification
docker exec homeassistant bash -c '
TOKEN=$(cat /config/.ha_token)
curl -s -o /dev/null -w "HTTP %{http_code}\n" \
  -H "Authorization: Bearer ${TOKEN}" http://localhost:8123/api/
'
# → doit renvoyer HTTP 200
```

Le script appelle ensuite `input_boolean/turn_on|turn_off` et `persistent_notification/create` pour tenir HA informé en fin de traitement, quelle que soit la durée du scan.

---

5. Les trois scripts

5.1 — epson_scan_launcher.sh

Wrapper minimal qui détache le scan du timeout HA. C'est lui qui est appelé par les `shell_command`.

```
#!/bin/bash
nohup bash /config/scripts/epson_scan.sh "$@" > /tmp/last_scan.log 2>&1 &
disown
exit 0
```

5.2 — epson_scan.sh (v4)

Paramètres positionnels :

Position	Paramètre	Valeurs	Défaut
$1	Résolution	100, 200, 300, 600	300
$2	Source	Platen, Feeder	Platen
$3	Mode couleur	RGB24, Grayscale8, BlackAndWhite1	RGB24
$4	Format	pdf, jpeg	pdf
$5	Intent	Document, Photo, TextAndGraphic	Document
$6	**Mode**	**simplex, recto, verso**	simplex
$7	Nom document	Texte libre (optionnel)	scan_[date]

Comportement selon le mode :

- **simplex + Platen** : scanne 1 page depuis la vitre → PDF dans `/media/scans/`
- **simplex + Feeder** : scanne tout le chargeur → PDF multi-pages
- **recto** : scanne tout le chargeur, **garde le PDF en attente** dans `/media/scans/.duplex_pending/`, active `input_boolean.scan_attente_verso`
- **verso** : scanne tout le chargeur (pile retournée), **éclate les deux PDF** avec `pdfseparate`, intercale, fusionne avec `pdfunite` → un seul PDF final

Codes de sortie :

- Succès : `OK:/chemin/fichier.pdf:taille:N pages` (exit 0)
- Succès recto : `RECTO_OK:N:Retournez la pile...` (exit 0)
- Erreur : `ERREUR: description` (exit 1)

5.3 — epson_scan_reset.sh

Filet de sécurité, appelé par le bouton **RAZ**. Il enchaîne :

1. `pkill -f epson_scan.sh` — tue un scan encore en cours en arrière-plan
2. Parcourt tous les `JobUri` du `ScannerStatus` et les ferme (`GET NextDocument` + `DELETE`)
3. Supprime le lot recto-verso en attente (`.duplex_pending`)
4. Nettoie les fichiers temporaires (`/tmp/scan_adf_*`, `/tmp/scan_headers_*`)
5. Remet `scan_en_cours` et `scan_attente_verso` à `off` via l'API HA
6. Relit l'état final et envoie une notification récapitulative

Sortie type :
`RESET_OK:jobs_fermes=2:lot_duplex_nettoye=1:etat_scanner=Idle:etat_adf=ScannerAdfEmpty`

---

6. Logique du recto-verso manuel

L'ADF ne gère pas le duplex matériel. Le recto-verso se fait donc en **deux passages**, avec retournement manuel de toute la pile.

Principe

```
Passage 1 (rectos) : l'ADF scanne les pages 1, 3, 5, 7...
[L'utilisateur retourne TOUTE la pile d'un bloc]
Passage 2 (versos) : l'ADF tire d'abord le verso de la DERNIÈRE feuille
                     → pages scannées dans l'ordre 8, 6, 4, 2
Fusion : rectos en positions impaires (1,3,5,7)
         versos INVERSÉS en positions paires (2,4,6,8)
         → PDF final dans l'ordre 1,2,3,4,5,6,7,8 ✓
```

Implémentation (méthode robuste)

La manipulation de tableaux en shell BusyBox s'étant révélée peu fiable (premier essai : ordre 1-3-4-2 au lieu de 1-2-3-4), l'intercalage se fait par **renommage positionnel** puis tri alphabétique :

1. `pdfseparate recto_batch.pdf → r-001.pdf, r-002.pdf...`
2. `pdfseparate verso_batch.pdf → v-001.pdf, v-002.pdf...`
3. Chaque recto *i* (ordre normal, `sort`) est copié en `page_$((i*2+1)).pdf`
4. Chaque verso *j* (ordre **inversé**, `sort -r`) est copié en `page_$((j*2+2)).pdf`

5. `pdfunite $(ls page_*.pdf | sort)` → PDF final dans le bon ordre

---

7. Configuration Home Assistant

configuration.yaml

Input selects

```
input_select:
  scan_source:
    name: Source de scan
    options: ["Vitre", "Chargeur (ADF)"]
    initial: "Vitre"
    icon: mdi:scanner
  scan_resolution:
    name: Résolution
    options: ["100", "200", "300", "600"]
    initial: "300"
    icon: mdi:resize
  scan_color:
    name: Mode couleur
    options: ["Couleur", "Niveaux de gris", "Noir et blanc"]
    initial: "Couleur"
    icon: mdi:palette
  scan_format:
    name: Format de sortie
    options: ["PDF", "JPEG"]
    initial: "PDF"
    icon: mdi:file-document
```

Input booleans

```
input_boolean:
  scan_en_cours:
    name: Scan en cours
    icon: mdi:progress-clock
```

```
  scan_attente_verso:
    name: En attente versos
    icon: mdi:rotate-3d-variant
```

Shell commands

⚠ **Aucun opérateur shell** (`>`, `&`, `|`) dans ces lignes — voir §3.3. Elles appellent toutes le *launcher*.

```
shell_command:
  epson_scan: "bash /config/scripts/epson_scan_launcher.sh {{
states('input_select.scan_resolution') }} {{ 'Feeder' if states('input_select.scan_source') ==
'Chargeur (ADF)' else 'Platen' }} {{ {'Couleur':'RGB24','Niveaux de gris':'Grayscale8','Noir
et blanc':'BlackAndWhite1'}[states('input_select.scan_color')] }} {{
states('input_select.scan_format') | lower }} Document simplex"
  epson_scan_recto: "bash /config/scripts/epson_scan_launcher.sh {{
states('input_select.scan_resolution') }} Feeder {{ {'Couleur':'RGB24','Niveaux de
gris':'Grayscale8','Noir et blanc':'BlackAndWhite1'}[states('input_select.scan_color')] }} {{
states('input_select.scan_format') | lower }} Document recto"
  epson_scan_verso: "bash /config/scripts/epson_scan_launcher.sh {{
states('input_select.scan_resolution') }} Feeder {{ {'Couleur':'RGB24','Niveaux de
gris':'Grayscale8','Noir et blanc':'BlackAndWhite1'}[states('input_select.scan_color')] }} {{
states('input_select.scan_format') | lower }} Document verso"
  epson_scan_reset: "bash /config/scripts/epson_scan_reset.sh"
```

Command line sensors (statut)

On utilise `command_line` plutôt que `rest` : le client HTTP Python de HA gère mal le certificat auto-signé de l'imprimante, même avec `verify_ssl: false`.

```
command_line:
  - sensor:
      name: "Epson Scanner Status"
      unique_id: epson_et4856_scanner_status
      icon: mdi:scanner
      command: "curl -s -k https://192.168.10.25/eSCL/ScannerStatus 2>/dev/null"
      scan_interval: 30
      value_template: >
        {% set matches = value | regex_findall('<pwg:State>(.*?)</pwg:State>') %}
        {% if matches %}
```

```
          {% set map = {'Idle':'Prêt','Processing':'En
cours','Testing':'Test','Stopped':'Arrêté'} %}
          {{ map.get(matches[0], matches[0]) }}
        {% else %}
          Indisponible
        {% endif %}
  - sensor:
      name: "Epson ADF Status"
      unique_id: epson_et4856_adf_status
      icon: mdi:tray-full
      command: "curl -s -k https://192.168.10.25/eSCL/ScannerStatus 2>/dev/null"
      scan_interval: 30
      value_template: >
        {% set matches = value | regex_findall('<scan:AdfState>(.*?)</scan:AdfState>') %}
        {% if matches %}
          {% set map = {'ScannerAdfEmpty':'Vide','ScannerAdfLoaded':'Papier
chargé','ScannerAdfJam':'Bourrage'} %}
          {{ map.get(matches[0], matches[0]) }}
        {% else %}
          Indisponible
        {% endif %}
```

scripts.yaml

Les scripts HA sont volontairement **minimalistes** : ils allument juste l'indicateur et lancent la commande. C'est `epson_scan.sh` qui, une fois réellement terminé, éteint les booléens et envoie la notification via l'API.

```
scanner_epson:
  alias: "Scanner un document"
  icon: mdi:scanner
  mode: single
  sequence:
    - action: input_boolean.turn_on
      target:
        entity_id: input_boolean.scan_en_cours
    - action: shell_command.epson_scan

scanner_epson_recto:
```

```
  alias: "Scanner rectos (duplex)"
  icon: mdi:file-document-multiple
  mode: single
  sequence:
    - action: input_boolean.turn_on
      target:
        entity_id: input_boolean.scan_en_cours
    - action: shell_command.epson_scan_recto

scanner_epson_verso:
  alias: "Scanner versos (duplex)"
  icon: mdi:file-document-check
  mode: single
  sequence:
    - action: input_boolean.turn_on
      target:
        entity_id: input_boolean.scan_en_cours
    - action: shell_command.epson_scan_verso

scanner_epson_reset:
  alias: "Réinitialiser scanner"
  icon: mdi:restart-alert
  mode: single
  sequence:
    - action: shell_command.epson_scan_reset
```

---

8. Dashboard (onglet Scan)

Situé dans le dashboard `lovelace.dashboard_roborock`, onglet `path: scan`. Utilise les cards **Mushroom** et **state-switch** (HACS) pour rester cohérent avec le reste du dashboard. Layout compact `vertical` car affiché sur un **NSPanel**.

Composition

1. Ligne 1 : statut Scanner + statut Chargeur ADF (couleurs conditionnelles)
2. Ligne 2 : selects Source + Résolution
3. Ligne 3 : selects Couleur + Format

4. Ligne 4 : les 3 boutons d'action côte à côte

Boutons

Bouton	États	Action
Scanner	Vert « Scanner » → Orange « Scan... »	`script.scanner_epson` — respecte le select Source
Recto-Verso	Bleu « Recto-Verso » → Orange « Verso / Retournez »	`scanner_epson_recto` puis `scanner_epson_verso` — **force toujours l'ADF**
RAZ	Rouge, fixe	`script.scanner_epson_reset`

Le bouton Recto-Verso bascule automatiquement d'un état à l'autre selon `input_boolean.scan_attente_verso`, guidant l'utilisateur dans le workflow en 2 étapes.

Entités utilisées

Entité	Rôle
sensor.epson_scanner_status	État du scanner (Prêt / En cours)
sensor.epson_adf_status	État du chargeur (Vide / Papier chargé / Bourrage)
input_select.scan_source	Vitre / Chargeur ADF (bouton Scanner uniquement)
input_select.scan_resolution	Résolution DPI
input_select.scan_color	Couleur / gris / N&B
input_select.scan_format	PDF / JPEG
input_boolean.scan_en_cours	Indicateur scan en cours
input_boolean.scan_attente_verso	Bascule l'état du bouton Recto-Verso
script.scanner_epson[_recto|_verso|_reset]	Scripts déclenchés par les boutons

---

9. Dépannage

Le scanner reste bloqué sur « Numérisation » (écran LCD)

Le job eSCL n'a pas été fermé (scan interrompu, bourrage, processus tué). **Appuyer sur le bouton RAZ du dashboard**, ou en CLI :

```
docker exec homeassistant bash /config/scripts/epson_scan_reset.sh
```

Le scanner répond HTTP 503

Un job précédent est encore actif. Vérifier l'état, puis lancer le RAZ :

```
docker exec homeassistant curl -s -k https://192.168.10.25/eSCL/ScannerStatus | sed 's/>\n
```

Rien n'arrive dans Paperless

Diagnostic en 4 étapes :

```
# 1. Le scanner répond-il ?
docker exec homeassistant curl -s -k https://192.168.10.25/eSCL/ScannerStatus | head -c 300

# 2. pdfunite est-il accessible ?
docker exec homeassistant ls -la /config/poppler-persist/usr/bin/pdfunite

# 3. Le processus tourne-t-il encore ?
docker exec homeassistant ps aux | grep epson_scan

# 4. Que dit le journal du dernier scan ?
docker exec homeassistant cat /tmp/last_scan.log

# 5. Un lot recto-verso resté en attente ?
docker exec homeassistant ls -la /media/scans/.duplex_pending/
```

« Timed out running command ... after: 60s » dans les logs HA

Le `shell_command` n'appelle pas le *launcher* mais directement `epson_scan.sh`. Vérifier :

```
docker exec homeassistant grep -A 4 "^shell_command:" /config/configuration.yaml
```

Chaque ligne `epson_scan*` doit appeler `epson_scan_launcher.sh`, **sans** `nohup`, **sans** `>`, **sans** `&` (voir §3.3).

« pdfunite: command not found » / fusion PDF échouée

Vérifier que la copie persistante existe et que le script pointe dessus :

```
docker exec homeassistant ls -la /config/poppler-persist/usr/bin/
docker exec homeassistant head -12 /config/scripts/epson_scan.sh   # doit contenir les 2
export PATH/LD_LIBRARY_PATH
```

Si `/config/poppler-persist/` est vide, refaire la copie initiale (voir §3.4).

Ordre des pages incorrect en recto-verso

Vérifier que le script est bien en v4 et utilise `pdfseparate` :

```
docker exec homeassistant head -3 /config/scripts/epson_scan.sh    # doit indiquer v4
docker exec homeassistant grep -c "pdfseparate" /config/scripts/epson_scan.sh
```

« grep: unrecognized option: P »

Script obsolète utilisant `grep -P`, incompatible BusyBox. Redéployer la version v4.

Doublons dans scripts.yaml

Un `cat >>` lancé deux fois crée des clés dupliquées. HA les signale en warning :

```
docker exec homeassistant grep -c "^scanner_epson" /config/scripts.yaml
docker exec homeassistant grep -i "duplicate key" /config/home-assistant.log | tail -5
```

Éditions de configuration : privilégier sed depuis l'hôte

Deux pièges rencontrés à répétition :

- `docker exec homeassistant python3 << 'EOF'` **ne transmet pas le heredoc** (il manque l'option `-i`) → la commande semble réussir mais ne fait rien
- `docker exec ... cat >> /config/fichier` écrit sur **l'hôte**, pas dans le container (la redirection est évaluée par le shell local)

→ Éditer directement les fichiers sous `/mnt/user/appdata/homeassistant/` avec `sed -i`, puis vérifier avec `docker exec homeassistant grep ...`.

---

10. Fichiers — Récapitulatif

Fichier (Unraid)	Fichier (container HA)	Rôle
.../homeassistant/scripts/epson_scan.sh	/config/scripts/epson_scan.sh	Script principal de scan (v4)
.../homeassistant/scripts/epson_scan_launcher.sh	/config/scripts/epson_scan_launcher.sh	Wrapper arrière-plan
.../homeassistant/scripts/epson_scan_reset.sh	/config/scripts/epson_scan_reset.sh	Réinitialisation / RAZ
.../homeassistant/.ha_token	/config/.ha_token	Jeton API HA (chmod 600)
.../homeassistant/poppler-persist/	/config/poppler-persist/	pdfunite / pdfseparate persistants

Fichier (Unraid)	Fichier (container HA)	Rôle
.../homeassistant/configuration.yaml	/config/configuration.yaml	input_select, input_boolean, shell_command, sensors
.../homeassistant/scripts.yaml	/config/scripts.yaml	Scripts HA scanner_epson*
.../homeassistant/.storage/lovelace.dashboard_roborock	—	Dashboard (onglet Scan)
/mnt/user/Sauvegarde.../consume/	/media/scans/	Sortie des scans = consume Paperless
—	/media/scans/.duplex_pending/	Lot recto-verso en attente (temporaire)
—	/tmp/last_scan.log	Journal du dernier scan lancé

---

11. Historique des problèmes résolus

Symptôme	Cause	Correctif
Aucune réponse sur `http://IP/eSCL/...`	eSCL exposé en HTTPS uniquement	Utiliser `https://` + `curl -k`
Boucle infinie / `grep: unrecognized option: P`	BusyBox ne supporte pas `grep -P`	Remplacé par `sed -n 's/.../p'`
HTTP 503 au lancement	Jobs fantômes d'un scan précédent	Fermeture propre + bouton RAZ
Imprimante bloquée sur « Numérisation »	Job jamais fermé après téléchargement	Ajout `GET NextDocument` (404) + `DELETE`
Ordre des pages 1-3-4-2 en recto-verso	Concaténation au lieu d'intercalage, versos non inversés	`pdfseparate` + renommage positionnel + `sort -r`
`RECTO_OK:1` pour 2 feuilles	L'ET-4856 renvoie tout le lot ADF en un seul PDF	Suppression de la boucle page-par-page, comptage via `pdfinfo`
Sensor scanner « unavailable »	`rest` sensor + certificat auto-signé	Bascule sur `command_line` sensor avec curl

Symptôme	Cause	Correctif
Fusion échouée après redémarrage HA	`apk add` non persistant	Copie dans `/config/poppler-persist/` + PATH
Blocage systématique sur gros lots (5 feuilles+)	Timeout 60 s des shell_command HA	Exécution détachée via *launcher* + notification API
`nohup ... &` sans effet dans shell_command	Mode restreint avec templates : `>` et `&` non interprétés	Backgrounding déporté dans `epson_scan_launcher.sh`

---

12. Améliorations possibles

- **Trigger Telegram** : déclencher un scan via commande bot
- **Bouton Zigbee physique** : scanner en un appui depuis l'imprimante
- **Nommage depuis le dashboard** : le script accepte déjà un 7e paramètre (nom du document), mais l'ajout d'un champ `input_text` éditable dans une vue `sections` n'a pas abouti (entité créée en YAML = `editable: false` ; créée via l'UI = card invisible)
- **Niveaux d'encre** : intégration HACS `ha-epson-workforce` — tester `http://192.168.10.25/PRESENTATION/HTML/TOP/PRTINFO.HTML`
- **Barre de progression** : exploiter `ImagesCompleted` du `ScannerStatus` pour un sensor de progression pendant le scan
- **Détection de bourrage** : automation sur `sensor.epson_adf_status` = « Bourrage » → notification + proposition de RAZ
-

Revision #2
Created 2026-03-22 20:07:10 UTC by thymon
Updated 2026-07-29 10:09:28 UTC by thymon