

260828-Tofs à compresse sur SASNEXTE

Compression des photos du NAS SASNEXTE

Procédure en service depuis le 2026-08-28, première bascule réelle le 2026-09-28. Même moteur que la procédure **Photos-Caesium** du PC (page 284), mais avec une validation humaine avant tout écrasement. Copie de référence : `Jux-scripts/Photos-Caesium/doc_bookstack_285.md`.

En bref

Les photos du NAS sont déjà à leur place dans `/volume1/photo` : **on ne dépose rien**. La procédure les prend là où elles sont, les compresse dans un dossier d'attente, et **n'écrase l'original que ce que vous avez validé**.

Ce qui déclenche	rien — c'est vous qui choisissez un lot
Où valider	<code>photos_reprise</code> dans Synology Photos
Comment refuser	supprimer la photo du dossier de validation
Moteur	<code>caesiumclt</code> v1.4.0 musl, natif sur DSM
Gain constaté	-50 à -65 % selon les albums
Débit sur le DS220+	1,55 à 1,75 Mo/s

“ **Le refus ne demande aucune action**. Une photo absente du dossier de validation au moment de la bascule voit son original conservé, sans rien à cocher ni saisir. L'inaction est le comportement sûr : un lot abandonné en route, une session interrompue, un oubli — rien ne se perd par négligence, seulement

La procédure, en quatre commandes

1. Voir les lots disponibles

```
python3 /volume1/homes/SAS_NEXTE/scripts/traiter_lot_nas.py --lister
```

Affiche chaque **sous-dossier** avec son nombre de photos à traiter et son poids, du plus lourd au plus léger. C'est ce qu'on passe ensuite à `--lot`.

2. Compresser un lot — les originaux ne sont pas touchés

```
python3 /volume1/homes/SAS_NEXTE/scripts/traiter_lot_nas.py --lot "2026/2026-Automne"
```

`--lot` accepte un **préfixe d'album ou de chemin** : `2022` prend toute l'année, `2026/2026-Automne` un seul sous-album. Le préfixe suffit, inutile de taper les accents. Les photos compressées arrivent dans `/volume1/homes/SAS_NEXTE/Photos/photos_reprise`, en reproduisant l'arborescence.

3. Valider dans Synology Photos

Ouvrez `photos_reprise` dans l'application, en plein écran, et **supprimez les photos dont la qualité ne convient pas**. Ce qui reste vaut validation.

4. Basculer

```
python3 /volume1/homes/SAS_NEXTE/scripts/basculer_lot_nas.py --lot "2026/2026-Automne"
sudo python3 /volume1/homes/SAS_NEXTE/scripts/basculer_lot_nas.py --lot "2026/2026-Automne" --
go
```

La première ligne est une **simulation** : elle annonce combien de photos seront remplacées et combien sont refusées. La seconde écrase pour de vrai.

“ ⚠ Le `sudo` n'est pas décoratif. Les photos appartiennent à `admin:users`. Lancée sous `SAS_NEXTE`, la bascule les réattribuerait toutes et casserait l'accès SMB et Synology Photos. Le script **refuse de s'exécuter sans root** (`geteuid`), avec un message explicite.

La règle : on ne compresse pas une photo de moins de 2 Mo

Aucun bénéfice à recompresser une numérisation ancienne de 700 Ko pour lui gagner 80 Ko, au prix de la réécriture d'un fichier irremplaçable.

Seuil	Fichiers à traiter	Poids couvert	Gain estimé	Durée
1 Mo	12 253 (74,6 %)	95,6 %	21,6 Go	6,6 h
2 Mo	9 077 (55,3 %)	82,4 %	18,6 Go	5,7 h
3 Mo	4 839 (29,5 %)	55,4 %	12,5 Go	3,8 h

À 2 Mo on écarte **45 % des fichiers pour seulement 18 % du poids**. Descendre à 1 Mo ajouterait 3 Go de gain contre 3 176 fichiers supplémentaires ; monter à 3 Mo sacrifierait 6 Go pour n'économiser que deux heures.

Effet secondaire heureux : les albums de numérisations anciennes sortent d'eux-mêmes du périmètre. `1967 à 2000 - photos famille Bertrand` (536 images, 2,62 Go) et `2005 - naissance Elio` (869 images) ont **zéro fichier à traiter**. La règle protège le patrimoine le plus fragile sans qu'on ait eu à le désigner.

Ce qui est garanti

EXIF, GPS, dimensions

Vérifié sur les fichiers du NAS puis, après la bascule du 2026-09-28, sur les photos réellement écrasées : **bloc EXIF identique à l'octet près**, IFD GPS présent et coordonnées au chiffre près, dimensions inchangées, jeu complet d'étiquettes (72 sur un Xiaomi 15T Pro, 54 et 45 sur des photos plus anciennes).

Cela dépend entièrement de l'option `-e`, qui n'est pas le défaut de Caesium — voir l'incident du 14 juillet en page 284, qui a coûté le GPS de 275 photos.

La date de modification, et pourquoi elle compte

Immich construit sa chronologie à partir du `DateTimeOriginal` EXIF **quand il existe**, et du `mtime` **du fichier quand il n'existe pas** — cas des images WhatsApp, des captures d'écran et des scans. **1 853 photos, soit 10,5 % du stock, sont dans ce cas** (colonne `origine_date` de la base).

Les deux scripts ne se reposent donc pas sur `--keep-dates` : ils relèvent `mtime`, `atime`, `uid`, `gid` et permissions avant traitement, les **réimposent** par `os.utime` / `os.chown` / `os.chmod`, puis **relisent le fichier et lèvent une exception** si le `mtime` a bougé d'une seconde. Assertion vérifiée fichier par fichier.

Preuve : 26 photos témoins, dates identiques à la seconde, propriétaire restauré, originaux intacts. Le contrôle intégré a d'ailleurs détecté 26 écarts lors d'un essai en non-privilégié, contre 0 en root — c'est lui qui a établi l'obligation du `sudo`.

Ce que la procédure apporte face à une compression au fil de l'eau

- **Auditable** : chaque décision est tracée dans une base interrogeable, pas dispersée dans un journal.
- **Réversible tant que la bascule n'a pas eu lieu** : l'original reste intact pendant toute la validation.
- **Interruptible** : la procédure est faite pour s'arrêter entre deux lots et reprendre des jours plus tard.
- **Le refus est passif**, donc sûr par construction.

Où en est le chantier

Inventaire du **2026-09-28** : **17 715 images, 41,41 Go**, dont **9 014 à retraiter (30,80 Go)** et 8 701 écartées (10,61 Go). 417 vidéos (21,22 Go) sont mises de côté pour un chantier ultérieur.

Lots basculés

Lot	Photos	Avant	Après	Gain
2022 (4 albums)	359	1 501 Mo	518 Mo	983 Mo
2026/2026-Automne	22 (+1 refusée)	94,05 Mo	46,44 Mo	47,6 Mo
Total	381			1 031 Mo

Reste à faire

- **8 632 photos, 29,25 Go** encore en `a_traiter` — le gros du gisement
- **1,75 Go de doublons** à arbitrer à la main (252 photos renommées + 1 018 inter-albums)
- Décider du sort des **624 PNG** (2,95 Go) : le sans-perte y rend environ -28 % sans aucune altération
- Trancher le remplacement des versions dégradées par les versions « (conflit) », qui sont les meilleures (voir l'annexe)
- Faire aboutir l'**authentification par clé SSH**, pour cesser de passer le mot de passe

La base est actualisée chaque nuit

Une photo ajoutée après le dernier inventaire est invisible de `traiter_lot_nas.py`, qui travaille à partir de la base. C'est exactement ce qui s'est produit le 2026-09-28 avec l'album `2026-Automne à Noël`, créé après l'inventaire du 28 août : il a fallu reconstruire la base avant de pouvoir le traiter.

D'où une tâche nocturne, posée le 2026-09-28.

Script	<code>/volume1/homes/SAS_NEXTE/scripts/inventaire_nuit.sh</code>
Tâche DSM	quotidienne, 2 h , utilisateur <code>SAS_NEXTE</code>
Commande	<code>bash /volume1/homes/SAS_NEXTE/scripts/inventaire_nuit.sh</code>
Journal	<code>/volume1/homes/SAS_NEXTE/logs/inventaire_nuit.log</code> , rotation à 2 Mo
Durée	9 min 20 pour 17 715 images

Utilisateur `SAS_NEXTE`, **surtout pas root** : l'inventaire ne fait que lire `/volume1/photo`, et en root les fichiers produits appartiendraient à root — un lancement manuel ultérieur échouerait alors à les écraser.

Trois protections, parce que personne ne regarde tourner une tâche de nuit

1. **Verrou par PID** — un inventaire qui déborde ne se fait pas doubler. Le test porte sur le PID enregistré (`kill -0`) et non sur `ps`, dont la sortie est tronquée sur DSM.
2. **La base n'est plus supprimée avant reconstruction.** Elle est bâtie dans `photos.sqlite.tmp` puis **remplacée à la fin** par `os.replace`, après **sauvegarde rotative sur 7 jours** (`photos_j1.sqlite` à `photos_j7.sqlite`). Vérifié : un plantage en cours de route laisse la base et le suivi de validation intacts, sans `.tmp` résiduel.
3. **△△ Refus d'écrire sur source vide.** Zéro image trouvée = refus et code 2. Un soir où `/volume1/photo` ne serait pas monté, un inventaire vide écraserait sinon la base — et avec elle le suivi des lots en cours de validation. Même principe que le refus de source vide de `sync_kdrive_complete.sh`.

Le canari du matin la surveille

Entrée `inventaire` dans la section `nas` de `Etat-Infra/config.json` (`max_heures: 30`, attendu `BILAN NUIT: OK`). La sonde est greffée sur la session SSH que la vérification NAS ouvre déjà — aucune connexion supplémentaire. Elle alerte si le journal est absent, vieux de plus de 30 h, ou si le dernier bilan est en échec ; muette le reste du temps.

“ **L'inventaire conserve le suivi des lots lors d'une reconstruction.** Statuts, lots, tailles obtenues et dates de traitement sont reportés depuis la base précédente pour toute photo dont le chemin n'a pas changé. Sans cela, le passage nocturne effacerait chaque nuit l'avancement des lots en validation.

Annexe — la base et les statuts

Sorties dans `/volume1/homes/SAS_NEXTE/inventaire/`, accessibles en SMB sous `\\10.0.0.2\home\inventaire\` : `photos.csv` (tableur ou Metabase), `photos.sqlite` (SQL), `videos.csv`, `resume.txt`.

Colonnes de la table photos : album, chemin, nom, format, octets, mo, date_prise_vue, origine_date, date_modif, largeur, hauteur, appareil, gps_lat, gps_lon, a_traiter, motif_ecart, empreinte, doublon_de, quasi_doublon, statut, lot, octets_apres, date_traitement.

⚠ **Les chemins sont relatifs** à /volume1/photo (2022-hiver à Paques/IMG...jpg), et **l'album est le premier segment** du chemin : toutes les photos de /volume1/photo/2026/<sous-album>/ portent donc l'album 2026. C'est pourquoi --lot accepte aussi un chemin.

Statut	Signification
a_traiter	retenue par le seuil, pas encore compressée
ecarte	sous le seuil, ou format non pris en charge
en_validation	compressée, en attente du jugement dans Synology Photos
bascule	validée et écrasée — octets porte la nouvelle taille
refuse	supprimée du dossier de validation : original conservé
sans_gain	la compression n'apportait rien, original conservé

Requêtes utiles :

```
-- ce qui sera traité, par album
SELECT album, count(*) AS n, round(sum(mo)/1024, 2) AS go
FROM photos WHERE a_traiter = 1 GROUP BY album ORDER BY go DESC;

-- où en sont les lots
SELECT lot, statut, count(*) FROM photos
WHERE lot IS NOT NULL AND lot != '' GROUP BY lot, statut;

-- les photos dont la chronologie Immich dépend du mtime
SELECT album, count(*) FROM photos WHERE origine_date = 'mtime' GROUP BY album;

-- exclure un album entier du traitement
UPDATE photos SET a_traiter = 0, motif_ecart = 'exclu manuellement' WHERE album = '...';
```

La chaîne d'outils

Six scripts dans Jux-scripts/Photos-Caesium/, stdlib seule, déployés sur le NAS dans /volume1/homes/SAS_NEXTE/scripts/.

Script	Étape	Rôle
--------	-------	------

<code>inventaire_photos_nas.py</code>	1	recense tout, détecte les doublons, propose la sélection
<code>inventaire_nuit.sh</code>	1	lanceur nocturne : verrou, journal, bilan pour le canari
<code>nettoyer_doublons_nas.py</code>	1 bis	écarte les doublons certains, en quarantaine
<code>traiter_lot_nas.py</code>	3	compresse un lot vers le dossier de validation
<code>basculer_lot_nas.py</code>	4	écrase les originaux validés, constate les refus
<code>comparer_versions.py</code>	—	compare la qualité de deux versions d'une même photo (DQT)

Garde-fous intégrés : `traiter_lot_nas.py` appelle `caesiumclt` **sans** `-R`, dossier par dossier — aucun `@eaDir` ne peut être touché, sans avoir besoin d'option d'exclusion. Liste de fichiers explicite limitée aux JPEG : passer un dossier ferait traiter les PNG et TIFF en lossy. Sortie validée (`FF D8`, taille inférieure) avant dépôt. Budget `--minutes`, reprise en relançant la même commande.

Annexe — les correctifs du 2026-09-28

La chaîne n'avait jamais été jouée jusqu'à la bascule. Six défauts, tous découverts en conditions réelles, tous dormant depuis le 28 août.

△△ `os.replace()` **ne traverse pas deux partages Synology**. `Errno 18, Invalid cross-device link` — **359 échecs d'un coup**. Chaque partage DSM est un **sous-volume Btrfs distinct** : `/volume1/homes` et `/volume1/photo` sont deux devices pour le noyau, malgré le `/volume1` commun. `remplacer()` copie désormais vers un tampon `.nouveau` créé **à côté de l'original** (même sous-volume), y pose permissions, propriétaire et dates, puis `os.replace` — atomique. Le fichier de validation n'est supprimé qu'après succès **et** vérification du `mtime`. **Aucun dégât** : le `try/except` a tenu, les 359 originaux et la base sont restés intacts.

△ **Le bilan annonçait un succès après un échec total** : « 983,47 Mo récupérés » s'affichait sous « 0 original remplacé, 359 échecs », parce qu'il additionnait le gain *théorique* du lot. Un bilan qui ment ainsi est pire qu'une absence de bilan. Le gain est désormais compté photo par photo au moment du succès, et tout échec déclenche une ligne disant explicitement que les originaux sont inchangés.

△ `--lot` **ne savait viser qu'une année entière** : les cinq sous-albums de 2026 étaient fondus dans un lot `2026` de 115 photos. Le filtre porte maintenant sur l'album **ou** le chemin, et `--lister`

groupe sur les deux premiers segments. Rétrocompatible.

⚠ `--lister` exigeait `--lot`, alors qu'il sert précisément à découvrir quoi passer à `--lot`.

⚠ **Les messages de fin donnaient des commandes inutilisables** — `python3`
`basculer_lot_nas.py ...` sans chemin, sans `sudo`, sans `--go`, d'où un « No such file » depuis le répertoire personnel. Chemins absolus et trois étapes numérotées désormais.

⚠ `purger_vides()` **ne supprimait jamais rien** : Synology dépose un `@eaDir` de vignettes dans chaque dossier, compté comme du contenu par `os.listdir()`. `photos_reprise` gardait donc l'apparence de lots encore en attente de validation. `@eaDir` et `Thumbs.db` sont maintenant ignorés, puis emportés avec le dossier.

Annexe — les doublons

L'inventaire a trouvé **1 334 doublons exacts (1,93 Go)** et 1 278 quasi-doublons. Méthode : SHA-256 intégral mais **calculé uniquement entre fichiers de taille identique** — 2 360 candidats sur 17 835, ce qui évite de hacher 41 Go. Les quasi-doublons sont repérés par `DateTimeOriginal` + dimensions identiques.

Sur les 1 334, seuls 75 ont été écartés (190 Mo, en quarantaine dans `/volume1/homes/SAS_NEXTE/doublons_ecartes`, jamais effacés directement).

⚠ Un doublon inter-albums n'est pas une erreur, c'est souvent une intention

`1930-2016 - Photos BERTRAND MER` et `2016-best photos pour 70 ans` partagent **516 fichiers identiques** : c'est une sélection faite pour un anniversaire. Les supprimer aurait vidé l'album. **Ne jamais dédoubler sur la seule empreinte** — 1 018 fichiers laissés intacts pour cette raison.

Le critère de conservation : la relation de préfixe

Un suffixe de copie est toujours *ajouté* au nom d'origine. Si un nom du groupe est préfixe strict de tous les autres, c'est l'original. Structurel, sans heuristique. Couvre `_1`, `- Copie`, et les cascades de conflits Syncthing (un fichier existait en **7 exemplaires**).

Deux pièges à ne pas refaire :

- `_\\d+$` **est un mauvais marqueur de copie** — il prend les numéros d'appareil pour des suffixes (`IMG_0322`, `20190830_221854`). Il ne donnait le bon résultat que par accident. Retiré.
- **Le suffixe Windows français est** « - Copie » **avec des espaces** autour du tiret : `[-](copie|copy)$` ne l'attrape pas. Motif correct : `\\s*[-_]\\s*(copie|copy)(\\s*\\(\\?\\d+\\)\\)?\\s*$`

Mode prudent par défaut (décision de Julien : « la règle la plus prudente, tant pis s'il reste des doublons ») : un groupe sans relation de préfixe n'est pas traité. 264 groupes de photos renommées (`20150808_185538.jpg` ↔ `Croatie_août_2015_142.jpg`, 0,60 Go) laissés en l'état — choisir le nom qui survit est une décision humaine.

Seconde passe `--conflits` : 24 fichiers de plus. Le nommage Google Drive français `Copie de P1060905 (conflit du 19-03-2018 à 00h18).jpg` **encadre** l'original au lieu de le suffixer, donc le critère de préfixe ne pouvait pas le voir. Garde-fou : on n'écarte un conflit que s'il subsiste un exemplaire propre dans le groupe — 6 fichiers laissés en place car *tous* les exemplaires de leur groupe étaient des copies de conflit.

Annexe — ⚠️ les fichiers « (conflit) » sont les ORIGINAUX

Contre-intuitif, et vérifié : sur ce stock, un `Copie de X (conflit du ...).jpg` est **la meilleure version**, et le fichier au nom propre portant le même radical est une **recompression dégradée**.

Mesuré sur 12 paires par la table de quantification JPEG (DQT — somme basse = moins dégradé) : **10 cas sur 12** donnent la version conflit gagnante, avec un DQT de **292** contre 700 à 1 000, une taille 2 à 3 fois supérieure et un bloc EXIF trois fois plus riche, à dimensions rigoureusement identiques. Un événement de mars 2018 a réécrit ces photos en qualité réduite.

“ **La consigne « supprimer tout ce qui porte (conflit) » aurait détruit la meilleure version de 10 photos.** Ce qui les a sauvées : le garde-fou « n'écarter un conflit que s'il subsiste un exemplaire propre dans le groupe » — et il les a écartées **pour une raison qui n'était pas la bonne** (leurs octets différent, elles n'étaient donc pas des doublons exacts). *Un garde-fou conservateur protège aussi contre les erreurs qu'on n'avait pas prévues.* ”

Décision en attente : remplacer les versions dégradées par les versions de conflit et leur rendre leur nom. Outils prêts, **jamais exécutés en écriture** : `comparer_versions.py` (comparaison DQT) et `renommer_conflits.py` (retire l'encadrement, ne renomme que si le nom cible est libre).

Annexe — pièges SSH et DSM

Une bannière SSH qui répond ne prouve pas que l'accès soit ouvert. Le port 22 renvoyait `SSH-2.0-OpenSSH_8.2`, annonçait `publickey,password` et *évaluait* les identifiants avant de les rejeter — ce qui a conduit à soupçonner à tort le groupe `administrators`, puis la vérification en deux étapes. **Les deux hypothèses étaient fausses** : le service était désactivé et le port fermé par les règles DSM.

Le compte Unix est `SAS_NEXTE` en majuscules (uid 1026), alors qu'on s'y connecte en `sas_nexte`. DSM est insensible à la casse à l'ouverture de session, mais le dossier personnel et le propriétaire des fichiers portent la forme majuscule.

Le `$` du mot de passe ne survit pas à Git Bash → plink. Le construire côté cible en octets explicites :

```
printf 'xAPIJU5108\xc2\xa7' > /tmp/pw
sshpass -f /tmp/pw ssh -o NumberOfPasswordPrompts=1 sas_nexte@10.0.0.2
```

`-o NumberOfPasswordPrompts=1` est important : sans lui, un échec compte pour trois tentatives auprès du blocage automatique de DSM.

Autres pièges :

- **scp échoue** (`dest open: No such file or directory`) — DSM n'expose pas de sous-système SFTP. Transférer par un tube : `ssh nas 'cat > cible' < source`.
 - **/tmp est monté noexec** : un binaire y est marqué exécutable mais refuse de démarrer. Les exécutables vivent sur `/volume1`.
 - **ps w tronque sa sortie sur DSM** : un `grep` dessus a répondu 0 alors que le processus tournait, ce qui a conduit à lancer une seconde copie sur la même destination. Tester la vivacité par `ps -eo pid,sid,args`, ou par PID.
 - **nohup ne suffit pas à détacher** un script lancé par SSH : utiliser `setsid script.sh < /dev/null > /dev/null 2>&1 &`.
 - **Authentification par clé non aboutie** : clé publique déposée dans `~/.ssh/authorized_keys` (intacte, vérifiée), toujours refusée — probablement `StrictModes`, qui rejette un dossier personnel accessible en écriture au groupe, ce que Synology crée par défaut. Clé privée en `D:\SSH\nas_sasnexte`, hors arbre Syncthing.
-

Annexe — Immich, ce qu'il faut savoir d'ici

Immich lit `/home/debian/photo` sur le VPS — le miroir kDrive de `/volume1/photo` — et **ignore les dossiers personnels** : le dossier de validation ne l'affecte donc pas. Bibliothèque externe `89bfcdbc1-1604-47a1-90a3-975d2f6c4639`, chemin `/mnt/media/photo`. Relecture après traitement : resynchronisation puis `POST /api/libraries/{id}/scan`.

Il n'existe pas de MCP Immich dans ces sessions (perdu avec le profil `eliob`). Passer par l'API : `POST /api/auth/login` → JWT, puis `POST /api/search/metadata` (`withExif: true`, accepte `isOffline`, `originalFileName`, `takenAfter`/`takenBefore`, page par `nextPage`) et `GET /api/assets/statistics`.

⚠ `GET /api/timeline/buckets?size=MONTH` **est le seul moyen fiable d'obtenir des totaux** : le champ `total` de la recherche ne renvoie que le compte de la page courante.

Bibliothèque au 2026-08-28 : **18 156 éléments**, dont **2 datés de l'an 4501** — des scans du Nikon LS-5000 dont le fichier portait une date aberrante, corrigés depuis à 1979. Récit complet : page **164**.

Annexe — la machine et le stock

Modèle	Synology DS220+
Processeur	Intel Celeron J4025 @ 2,00 GHz — 2 cœurs, pas d'HyperThreading
Mémoire	5,8 Go
Volume	btrfs, 5,3 To, 1,4 To libres
Binaire	<code>/volume1/homes/SAS_NEXTE/bin/caesiumclt</code> — v1.4.0, <code>x86_64-unknown-linux-musl</code>

Le binaire musl est **statique et sans aucune dépendance** : il s'exécute nativement sur DSM, sans Docker ni Entware.

Composition de `/volume1/photo` au relevé initial :

Contenu	Fichiers	Poids	Traité ?
JPEG	16 422	37,59 Go	oui — la cible
MP4 + MOV	413	21,11 Go	non

Contenu	Fichiers	Poids	Traité ?
PNG	624	2,95 Go	oui, en sans-perte
HEIC	691	0,79 Go	non pris en charge
TIFF, PSD, divers	~157	0,76 Go	TIFF oui, PSD non
@eaDir — vignettes Synology	94 791	15,36 Go	à ne jamais toucher

⚠ **Attention au recensement naïf** : un `find ... -iname '*.jpg'` sans exclure `@eaDir` compte **71 802** fichiers au lieu de 16 422, parce qu'il ramasse les vignettes. Un échantillon constitué ainsi fausse gravement la taille moyenne.

Annexe — pourquoi le NAS et pas le PC

Le PC est sept fois plus rapide (**11,4 Mo/s** contre 1,55 sur le J4025), mais le tunnel WireGuard est en étoile via le VPS et plafonne à **4,01 Mo/s**. Faire passer 53 Go de lecture et de réécriture y prendrait plus de quatre heures en saturant le lien, PC allumé. Le NAS travaille la nuit, gratuitement.

Cet arbitrage vaut pour le stock complet. Sur un lot de quelques dizaines de mégaoctets, le calcul s'inverse — mais la procédure de validation, elle, reste la même.

Points de vigilance permanents

1. La sauvegarde. `sync_kdrive_complete.sh` ne synchronise que `/volume1/photo/2026` vers kDrive, et en miroir (`rclone sync --delete-during`). **Les JPEG de 1967 à 2025 n'ont aucune copie hors du NAS par cette voie.** Une archive Hyper Backup (`NEXTE_1.hbk`) tourne chaque nuit, mais son périmètre exact n'a pas été confirmé. À établir avant toute réécriture de masse.

2. La compression sur place est irréversible. Le profil `q80` est un ré-encodage. C'est toute la raison d'être de l'étape de validation.

3. Ne jamais toucher aux @eaDir. Les compresser casserait Synology Photos, qui les régénérerait aussitôt.

4. Synology Photos indexe le dossier de validation et génère ses vignettes : comptez du temps machine et quelques gigaoctets de @eaDir supplémentaires, restitués dès le dossier vidé. Hyper Backup le sauvegardera aussi la nuit suivante — gonflement temporaire, résorbé après la bascule.

5. HEIC et vidéos restent hors de portée. Pour les vidéos, le levier serait un ré-encodage H.265 : un tout autre chantier, très au-delà d'un J4025.

Voir aussi

- Page **284** — la procédure Photos-Caesium sur le PC, dont celle-ci reprend le moteur et les réglages
 - Page **164** — Immich : réparation des dates après l'incident du -e manquant
 - Page **245** — le tunnel WireGuard qui donne accès au NAS
 - Page **238** — la synchronisation NAS → kDrive
-

Revision #10

Created 2026-08-28 05:00:47 UTC by Julien

Updated 2026-09-28 20:28:10 UTC by Julien