

260903-Capture du texte sur applications Android vers Joplin

Récupérer le texte affiché par une application Android — un article de presse, typiquement — et le déposer dans **Joplin** ou dans le coffre **Obsidian**, depuis le PC.

Mis en place le 2026-09-03 sur le poste `julie`. **Reproductible sur un autre PC**, la procédure d'installation figure plus bas.

Le principe

Le script appelle `uiautomator dump`, un outil livré avec Android, qui exporte l'arbre d'accessibilité de l'écran en XML. On en tire les attributs `text` et `content-desc`.

“ **Ce n'est PAS de la reconnaissance optique.** On récupère le texte *réel* de l'application, avec ses accents, sa ponctuation et sa casse exacts. Aucune erreur de lecture possible, contrairement à un OCR sur capture d'écran.

Rien n'est installé sur la tablette. Tout passe par `adb`, déjà présent avec le SDK Android.

Ce qu'il faut sur le PC

Élément	Rôle
Python 3.12	le script n'utilise que la bibliothèque standard — rien à installer en plus
<code>adb</code>	fourni par <code>platform-tools</code> du SDK Android
Un appareil joignable	l'émulateur <code>emulator-5554</code> , ou un vrai téléphone en débogage USB

Le script est dans l'arbre Syncthing, donc **déjà présent sur toutes les machines** :

```
Jux-scripts/Android-Texte/capturer_texte.py
Jux-scripts/Android-Texte/README.md
```

Usage

```
# ecran courant, affiche dans la console
py -3.12 capturer_texte.py

# vers un fichier / le presse-papiers
py -3.12 capturer_texte.py -o article.txt
py -3.12 capturer_texte.py --presse-papiers

# vers Joplin ou Obsidian
py -3.12 capturer_texte.py --joplin
py -3.12 capturer_texte.py --obsidian
```

Option	Défaut	Rôle
<code>-o</code> , <code>--fichier</code>	—	fichier UTF-8
<code>--presse-papiers</code>	—	copie dans le presse-papiers Windows
<code>--joplin</code>	—	crée une note Joplin
<code>--carnet</code>	Captures Android	carnet Joplin, créé s'il manque
<code>--obsidian</code>	—	dépose un <code>.md</code> dans le coffre
<code>--coffre</code>	D:\Syncthing\Jux_Obsidian	racine du coffre
<code>--dossier</code>	Captures Android	sous-dossier du coffre
<code>--titre</code>	deviné	titre de la note
<code>--defiler</code>	—	fait défiler et accumule
<code>--serie</code>	emulator-5554	identifiant adb
<code>--adb</code>	—	chemin d' <code>adb.exe</code>
<code>--tri-position</code>	—	trier par coordonnées (voir pièges)

Codes de sortie : `0` succès, `1` erreur adb, `2` échec Joplin, `3` échec Obsidian.

Les deux destinations

Obsidian — la plus simple

Écrit un `.md` dans `<coffre>\Captures Android\`, avec l'en-tête YAML des notes existantes :

```
---
title: ...
created: 2026-09-03 04:54:54Z
updated: 2026-09-03 04:54:54Z
source: capture Android
---
```

Ni jeton, ni application ouverte, ni service à activer. Le coffre étant dans Syncthing, la note part sur tous les appareils par la synchronisation habituelle. Les noms de fichier sont assainis pour Windows et un suffixe `(2)`, `(3)`... évite d'écraser.

Joplin — via le Web Clipper, en local

Mise en place, une seule fois par PC :

1. Joplin → **Outils** → **Options** → **Web Clipper** → **Activer le service**
2. Copier le jeton affiché dans `D:\Android\joplin_token.txt`

La variable d'environnement `JOPLIN_TOKEN` est prioritaire si elle existe.

“ ⚠ **Le jeton doit être HORS de l'arbre Syncthing.** Déposé par erreur dans `D:\Syncthing\Jux_univers\Mes Notepads\` le 2026-09-03, il s'est répliqué sur les sept appareils du maillage avant d'être déplacé. Le risque reste modéré — l'API n'écoute que sur `127.0.0.1` du PC, détenir le jeton ailleurs ne donne aucun accès distant — mais la régénération est immédiate depuis les options de Joplin. Même principe que `/home/debian/.komga_scan.env` côté VPS.

Pourquoi pas le Joplin Server du VPS

`joplin.juxjux.ovh` est un **serveur de synchronisation**, pas une API de notes : les items y sont sérialisés pour la synchro, on ne peut pas y créer une note proprement. **La seule voie praticable est l'API locale du client de bureau**, sur `http://127.0.0.1:41184`.

Corollaire : Joplin doit être ouvert sur le PC au moment de la capture.

Les lanceurs du bureau

Deux fichiers `.cmd` sur le bureau : `Capturer vers Joplin` et `Capturer vers Obsidian`.

Ils vérifient la présence de Python et du script, écrivent une copie du texte dans `%TEMP%\derniere_capture.txt`, consignent les erreurs dans `D:\Android\capture_texte.log` et affichent les dernières lignes du journal en cas d'échec.

“ **Pourquoi des `.cmd` et pas des raccourcis `.lnk`** : un `.lnk` créé depuis Claude Code vers une cible sous `%LOCALAPPDATA%` **fige un chemin virtualisé** et ne fait rien du tout (voir page 282, section 6). Un `.cmd` est du texte brut : ses chemins sont littéraux et fonctionnent depuis l'explorateur.

Pièges rencontrés

⚠ 1. L'ordre du document, pas les coordonnées

Une première version triait les fragments par coordonnées `bounds`. **Faux** : dès qu'une partie du contenu est hors écran — le cas courant avec une WebView — ses `bounds` deviennent aberrants et **les paragraphes ressortent mélangés**.

L'ordre de l'arbre XML est l'ordre du document, donc l'ordre de lecture. C'est le comportement par défaut. `--tri-position` rétablit l'ancien, pour de rares interfaces natives dont l'arbre ne suit pas la mise en page.

⚠ 2. `--defiler` est souvent inutile

Une **WebView** expose l'intégralité du document chargé à l'accessibilité, pas seulement la partie visible. Un article de presse entier est capturé en un seul appel.

`--defiler` ne sert vraiment que pour les **listes à chargement progressif** — fil d'actualité, boîte de réception — où le contenu n'existe pas tant qu'on n'y est pas descendu.

⚠ 3. Le titre est deviné, et pas toujours trouvable

Sans `--titre`, le script prend **la plus longue des huit premières lignes**, entre 30 et 120 caractères, en écartant dates, heures, signatures (`Par X`), `Mis à jour`, `Lire aussi`.

Trois versions ont été nécessaires :

Règle essayée	Résultat obtenu
première ligne substantielle	« Éditos & Analyses » — la rubrique
la plus longue des premières	« Le 01/09/2026 à 06h45 Mis à jour... » — la date
+ filtre, plancher à 20 caractères	« trottinettes spatiales » — une bribe de phrase
+ plancher à 30 caractères	correct, ou repli sur la date

Il faut capturer depuis le HAUT de l'article. Une fois qu'on a défilé loin, l'en-tête sort de l'arbre et aucun titre n'est trouvable : la note prend alors la date (`Capture Android du 03-09-2026 a 07h18`). C'est délibéré — mieux vaut une date qu'un titre faux. Sinon, `--titre "..."`.

⚠ 4. `cmd clipboard` n'existe pas

Sur les images « Google Play », `adb shell cmd clipboard` répond `No shell command implementation`. D'où le passage par `clip.exe` côté Windows, en UTF-16LE.

⚠ 5. Ne pas faire défiler la tablette par `adb` à l'aveugle

Sept `input swipe` enchaînés pour « remonter en haut » ont fait **sortir l'application Les Échos de l'article** et atterrir sur un écran de recherche. Les gestes longs déclenchent des navigations imprévues.

Laisser l'utilisateur positionner la vue lui-même.

⚠ 6. Vérifier Joplin sur TOUTES les pages

`GET /folders` est **paginé**. Avec 106 carnets, une vérification limitée à la première page conclut à tort que le carnet n'existe pas. Le script, lui, suit `has_more` correctement.

```
# faux : ne lit que la page 1
Invoke-RestMethod "http://127.0.0.1:41184/folders?token=$j"
```

Installer sur un autre PC

1. **Python 3.12** — `winget install --id Python.Python.3.12 --scope user`
2. **platform-tools** — via le SDK Manager d'Android Studio, ou l'archive seule de Google
3. Adapter `ADB_DEFAULT` en tête du script, ou passer `--adb`
4. Pour Obsidian : adapter `COFFRE_DEFAULT`, ou passer `--coffre`
5. Pour Joplin : activer le Web Clipper et déposer le jeton **hors de Syncthing**
6. Recopier les deux `.cmd` du bureau en corrigeant les chemins en tête de fichier

Le script lui-même n'a **rien à installer** : stdlib seule.

Limites

- **Le texte dessiné en image n'apparaît pas** : infographies, PDF rastérisés, certains lecteurs. Il faudrait de l'OCR — `tesseract` est déjà disponible dans StirlingPDF sur le VPS si le besoin se présente.
- Une application déclarant ses vues `importantForAccessibility="no"` reste muette.
- Les fenêtres `FLAG_SECURE` (applications bancaires, vidéo protégée) ne sont pas exportées.

Vérifié le 2026-09-03

- capture d'un article des Échos : **6 563 caractères**, texte exact, ordre correct
- note Joplin créée dans un carnet auto-créé, contenu vérifié côté serveur
- note Obsidian déposée avec l'en-tête YAML conforme au coffre
- 4 tests unitaires sur le choix du titre

Si ça ne marche pas

Le journal `D:\\Android\\capture_texte.log` est le point d'entrée : les lanceurs y écrivent la sortie d'erreur et en affichent les dernières lignes en cas d'échec.

Une exécution saine ressemble à ceci :

```
===== 03/09/2026  7:42:55 - Joplin =====
  ecran 1 : 29 nouveaux fragments
29 fragments -> C:\\Users\\julie\\AppData\\Local\\Temp\\derniere_capture.txt
Joplin : note creee dans « Captures Android » – e4e99c3f...
```

Échec transitoire observé le 2026-09-03 à 07:23. La capture avait réussi — fichier de secours de 6,7 Ko écrit, article complet — mais aucune note n'avait été créée. Rejoué à l'identique quelques minutes plus tard : succès, sans qu'aucune modification n'ait été faite entre-temps.

Cause non établie. Deux hypothèses restent ouvertes : Joplin indisponible ou en cours de synchronisation à cet instant, ou un problème d'encodage sur la sortie d'erreur sous console `.cmd`. Une piste a été **écartée par la mesure** : le chemin Python codé en dur dans les `.cmd` pointe sous `%LOCALAPPDATA%`, mais l'installation y est bien **réelle** — le chemin conteneur `...\\Packages\\Claude_pzs8srxjxfjfc\\LocalCache\\Local\\Programs\\Python\\...` **n'existe pas**. C'est d'ailleurs le test à retenir pour trancher ce type de doute (page 282, section 6).

C'est pour cette raison que les lanceurs journalisent : si le cas se reproduit, le message sera conservé.

En cas d'échec, le texte n'est jamais perdu : il reste dans %TEMP%\derniere_capture.txt.

Contrôles rapides

```
# la tablette repond-elle ?
& 'D:\Android\Sdk\platform-tools\adb.exe' devices

# le service Web Clipper est-il actif ?
Invoke-WebRequest 'http://127.0.0.1:41184/ping' -UseBasicParsing # -> JoplinClipperServer

# le jeton est-il en place ?
Test-Path 'D:\Android\joplin_token.txt'
```

Voir aussi

- Page **282** — l'émulateur Android, son installation et ses pièges
- [Jux-scripts/Android-Texte/README.md](#) — documentation technique du script

Revision #3

Created 2026-09-03 05:25:46 UTC by Julien

Updated 2026-09-03 05:44:32 UTC by Julien