

vosk-cli-dictation — Manuel utilisateur

Installer, configurer et utiliser la dictée vocale hors ligne

Version v0.5.4-beta

2026-09-18

Table des matières

Démarrage express	3
Chapitre 1 — Démarrage	4
Prérequis	4
Plusieurs façons d'installer	4
Récupérer un modèle de langue	5
Premier démarrage	5
Une première dictée	6
Dicter la ponctuation	6
Désinstaller et mettre à jour	6
Chapitre 2 — Le moteur de reconnaissance vocale	8
Pour qui est cet outil	8
Ce que fait Vosk	8
Modèle acoustique et modèle de langue	9
Rien ne quitte la machine	9
Chapitre 3 — Les trois états : ÉCOUTE, DICTÉE, VEILLE	10
Vue d'ensemble	10
DICTÉE — dictée en cours	10
ÉCOUTE — en attente	10
Comment démarrer et arrêter une dictée	10
VEILLE — consommation quasi nulle, volontaire	11
Le seuil de confiance gouverne le déclenchement	11
Quand utiliser le mode veille	11
Le déroulé d'une session complète	11
Chapitre 4 — Configuration et personnalisation	13
Où se trouve le fichier de configuration	13

Choisir la langue du gabarit de configuration	13
Personnaliser les raccourcis clavier	13
Choisir comment la dictée démarre et s'arrête	14
Ponctuation : deux modes de saisie, par langue	14
Régler le seuil de confiance du déclenchement	14
Ajouter du vocabulaire et corriger des confusions	15
Où sont cherchés les modèles de langue	15
Chapitre 5 — Intégration système : Wayland et X11	16
Comment le texte arrive dans l'application	16
Le microphone	16
Les outils système utilisés, et leur choix automatique	16
Activer ydotool (solution de repli)	17
Raccourcis clavier : X11 et Wayland ne sont pas égaux	17
Que se passe-t-il si aucun outil d'injection n'est disponible	18
Chapitre 6 — Dépannage	19
Symptôme → cause probable → action	19
Où chercher un message d'erreur complet	20
Si le problème persiste	21

Démarrage express

Pour qui veut dicter dans les deux minutes, sans lire le reste tout de suite.

`vosk-cli-dictation` vise le matériel contraint et la dictée hors ligne, pas la précision maximale sur une station de travail puissante (Chapitre 2). Les raccourcis clavier ci-dessous supposent une session X11 ; sous Wayland, ils passent par le bureau (Chapitre 5).

1. Installation. Paquet natif de la distribution :

```
# Debian 12+, Ubuntu 24.04+, Mint 22+ ...
sudo apt install ./vosk-cli-dictation-<version>_all.deb

# Fedora
sudo dnf install ./vosk-cli-dictation-<version>.fc42.noarch.rpm

# openSUSE (Leap)
sudo zypper install ./vosk-cli-dictation-<version>.leap156.noarch.rpm

# Arch Linux
sudo pacman -U ./vosk-cli-dictation-<version>-any.pkg.tar.zst
```

2. Récupérer un modèle de langue, si aucun n'est déjà présent (le paquet ne les embarque pas — trop volumineux) :

```
vosk-cli-dictation --download-model fr
```

Le petit modèle de la langue (ici `fr` ; aussi `en`, `de`, `es`) est téléchargé, vérifié par sa somme SHA256 et installé dans `~/.local/share/vosk-models/`. Au tout premier lancement sans aucun modèle, l'outil propose ce même téléchargement de façon interactive. On peut aussi récupérer un modèle à la main sur <https://alphacephei.com/vosk/models> et le décompresser dans ce répertoire.

3. Dictée.

```
vosk-cli-dictation
```

Dire la phrase de réveil (« active dictée ») pour démarrer la dictée, parler, puis dire la phrase d'arrêt (« stop dictée ») pour la terminer et copier le texte dans le presse-papiers.

C'est tout pour commencer. La suite explique pourquoi et comment aller plus loin — personnaliser les raccourcis, comprendre le mode veille, dépanner un blocage.

Chapitre 1 — Démarrage

Prérequis

- **Un système Linux** avec une session graphique X11 ou Wayland. Sans serveur d’affichage, il n’y a pas de fenêtre active où écrire le texte reconnu.
- **Python 3** et un environnement virtuel (python3-venv).
- **Un outil d’injection de texte** : wtype ou ydotool sous Wayland, xdotool sous X11. Au moins un doit être présent, sinon l’outil refuse de démarrer (Chapitre 5).
- **Un microphone reconnu par le système** comme périphérique d’entrée par défaut (Chapitre 5).
- **De l’espace disque pour le modèle** : de 50 Mo (modèles «small») à plus d’un gigaoctet. En fonctionnement, une session utilise de l’ordre de 300 Mo de RAM (Chapitre 2).

Le paquet Debian déclare et installe lui-même toutes ses dépendances, système et Python. En mode clone Git, `pip install -r requirements.txt` installe les bibliothèques Python; l’outil d’injection (wtype, xdotool ou ydotool) est alors à installer séparément.

Plusieurs façons d’installer

vosk-cli-dictation propose plusieurs canaux, sans que l’un remplace l’autre.

Le paquet natif de la distribution (recommandé pour un usage courant)

```
# Debian 12+, Ubuntu 24.04+, Mint 22+ ...
sudo apt install ./vosk-cli-dictation-<version>_all.deb

# Fedora
sudo dnf install ./vosk-cli-dictation-<version>.fc42.noarch.rpm

# openSUSE (Leap)
sudo zypper install ./vosk-cli-dictation-<version>.leap156.noarch.rpm

# Arch Linux
sudo pacman -U ./vosk-cli-dictation-<version>-any.pkg.tar.zst
```

Quel que soit le format, le paquet installe une commande unique, `vosk-cli-dictation`, dans `/usr/bin` /. Il crée aussi un environnement Python isolé et y installe le moteur de reconnaissance vocale (Vosk) — cette étape a besoin du réseau au moment de l’installation, une seule fois. Rien n’est requis en tant que superutilisateur après coup.

L’installation elle-même ne modifie jamais les fichiers personnels : aucune configuration existante n’est écrasée, aucun répertoire du compte utilisateur n’est créé ni touché à ce stade. Ce n’est qu’au premier lancement, et à la récupération d’un modèle de langue (ci-dessous), que des fichiers apparaissent dans le répertoire personnel — jamais avant, jamais sans une action explicite de l’utilisateur.

Le clone Git (pour suivre le développement ou contribuer)

```
git clone https://github.com/RonanDavalan/vosk-cli-dictation.git
```

```
cd vosk-cli-dictation
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

Dans ce mode, l'outil se lance avec `python3 src/main.py` plutôt que par la commande `vosk-cli-dictation` du paquet, et sa configuration vit dans `config/config.yaml`, à côté du code — jamais dans le répertoire personnel. Les deux modes ne se mélangent pas : un clone Git ignore la configuration du paquet, et réciproquement.

Récupérer un modèle de langue

Aucun des deux canaux n'embarque de modèle de reconnaissance vocale — ils pèsent de 50 Mo à plus d'un gigaoctet, trop pour être livrés avec l'outil.

Le plus simple, en installation par paquet, est de laisser l'outil s'en charger :

```
vosk-cli-dictation --download-model fr
```

Cette commande récupère le petit modèle de la langue demandée (fr, ou en, de, es), vérifie sa somme SHA256, l'installe dans `~/.local/share/vosk-models/`, puis s'arrête. Au tout premier lancement sans aucun modèle, la même proposition est faite de façon interactive : répondre oui télécharge le petit modèle de la langue de l'interface.

Sinon, ou pour un modèle plus complet que le petit modèle par défaut, télécharger à la main sur <https://alphacephei.com/vosk/models>, puis décompresser :

- Installation par paquet : dans `~/.local/share/vosk-models/`.
- Clone Git : dans `vosk-model/`, à la racine du dépôt cloné.

Sans modèle présent au bon endroit, l'outil affiche au lancement les emplacements qu'il a cherchés et l'adresse de téléchargement — jamais un échec silencieux.

Premier démarrage

```
vosk-cli-dictation
```

(en mode clone Git, la même commande est `python3 src/main.py`, depuis la racine du dépôt et l'environnement virtuel activé)

Trois choses s'affichent : la vérification des dépendances externes (voir Chapitre 5), le modèle chargé, puis un rappel des commandes disponibles — phrases de réveil et d'arrêt, raccourcis clavier, mots de ponctuation. Ce rappel est régénéré à chaque lancement à partir de la configuration active : il reflète toujours ce qui est réellement utilisable, jamais une liste figée dans la documentation qui pourrait devenir fausse après une personnalisation.

Une première dictée

Deux façons équivalentes de démarrer une dictée, au choix :

- **Par la voix** : prononcer la phrase de réveil de la langue active (« active dictée »). La phrase complète est nécessaire : un seul mot (« active ») ne déclenche rien — le réveil n’est retenu que sur la phrase entière, avec une confiance suffisante.
- **Par le clavier** : appuyer sur **Alt+H** — ce raccourci bascule entre ÉCOUTE (l’état de départ, celui où l’on est juste après le lancement) et DICTÉE (voir Chapitre 3 pour les états) ; il sert aussi bien à démarrer qu’à mettre en pause. Il ne copie jamais rien dans le presse-papiers : voir plus bas pour terminer une dictée. Sous une session **Wayland**, ces raccourcis ne sont captés que pour les fenêtres passant par XWayland ; pour une fenêtre Wayland native, il faut les passer par le bureau (Chapitre 5).

Parler ensuite normalement. Le texte reconnu s’écrit directement dans l’application qui a le focus — un terminal, un éditeur, un navigateur, n’importe quoi.

Pour terminer et récupérer le texte dans le presse-papiers : dire la phrase d’arrêt (« stop dictée ») ou appuyer sur **Alt+S**. Une phrase d’arrêt entendue sous le seuil de confiance est absorbée — elle n’est ni tapée dans le document, ni déclenchée, et un indice s’affiche : la répéter, ou utiliser **Alt+S**.

Un pilotage plus fiable, sous X11 comme sous Wayland, consiste à lier les sous-commandes (`--toggle`, `--finalize`, `--sleep-toggle`) à des raccourcis du bureau (Chapitre 5) plutôt que d’utiliser les raccourcis intégrés.

Dictier la ponctuation

Les mots de ponctuation sont dictés comme le reste du texte — « virgule », « point d’exclamation », « nouvelle ligne », etc. La liste complète dépend de la langue active et s’affiche au lancement (voir ci-dessus) ; elle est aussi personnalisable, et la façon de déclencher une commande de ponctuation peut changer selon le mode de saisie de la langue (Chapitre 4).

Désinstaller et mettre à jour

Paquet natif. Le gestionnaire de la distribution retire la commande et l’environnement Python isolé (`/usr/lib/vosk-cli-dictation/venv`) : `sudo apt remove vosk-cli-dictation` (Debian 12+ et dérivées ; `sudo apt purge vosk-cli-dictation` efface en plus le modèle de configuration d’usine), `sudo dnf remove vosk-cli-dictation` (Fedora), `sudo zypper remove vosk-cli-dictation` (openSUSE), `sudo pacman -R vosk-cli-dictation` (Arch Linux). Le fichier de configuration personnel (`~/.config/vosk-cli-dictation/config.yaml`) et les modèles de langue (`~/.local/share/vosk-models/`) ne sont jamais touchés par la désinstallation — les supprimer à la main si on le souhaite.

Clone Git. Supprimer le répertoire cloné : il contient tout (code, environnement virtuel `venv/`, éventuels modèles dans `vosk-model/`). Rien n’est écrit ailleurs dans ce mode.

Mise à jour. Avec un paquet natif, installer le fichier plus récent avec la même commande que pour la première installation (`apt install`, `dnf install`, `zypper install`, `pacman -U`) : il remplace la version

en place sans toucher à la configuration personnelle. Avec le clone Git, faire `git pull`, puis relancer `pip install -r requirements.txt` dans l'environnement virtuel activé pour répercuter d'éventuelles nouvelles dépendances.

Chapitre 2 — Le moteur de reconnaissance vocale

Pour qui est cet outil

Avant d'entrer dans les principes techniques, une clarification honnête : vosk-cli-dictation n'a pas vocation à rivaliser avec les moteurs de dictée les plus précis du marché sur un ordinateur de bureau récent. Ce n'est pas une faiblesse cachée — c'est un choix d'architecture assumé, qui rend l'outil excellent sur un terrain précis :

- **Matériel contraint** — Raspberry Pi, ancien portable, machine sans GPU. Le moteur n'a jamais besoin d'accélération matérielle. En régime, une session utilise de l'ordre de **300 Mo de RAM** (209 à 383 Mo selon la langue et la machine — modèles `small` mesurés : `vosk-model-small-fr-0.22`, `-en-us-0.15`, `-de-0.15`, `-es-0.42`) — modeste face à un navigateur, mais pas négligeable sur une machine qui en a peu.
- **Réactivité en direct** — le texte apparaît au fur et à mesure de la parole, sans attendre la fin d'un segment. C'est une propriété de l'architecture du moteur (voir plus bas), pas un réglage qu'on pourrait perdre en changeant un paramètre. Le décodage ne prend pas de retard sur la parole : le traitement d'un bloc audio demande moins de temps que la durée du bloc lui-même (facteur temps réel RTF < 1,0), y compris sur le cas le plus lourd : la dictée en français sur un Raspberry Pi 4 (RTF 0,73) ou sur un AMD A4-6210 (RTF 0,79). Le français est la langue la plus coûteuse à décoder ; sur le même matériel modeste, l'anglais, l'allemand et l'espagnol tournent à RTF 0,20–0,33.
- **Rien ne sort de la machine** — aucune donnée audio ni texte n'est jamais transmise à un service distant, pendant la reconnaissance comme à tout autre moment.
- **Une chaîne de dépendances minimale** — pas de bibliothèque d'apprentissage profond lourde à installer, pas de pilote de carte graphique requis.

En contrepartie, la compréhension du sens reste limitée : deux mots qui se prononcent pareil mais s'écrivent différemment (« ce »/« se », « sauna »/« sonna ») ne sont pas toujours distingués correctement à partir du seul son. Le Chapitre 4 explique comment corriger ces cas un par un, une fois repérés à l'usage — ce n'est pas automatique.

Si l'objectif est de rédiger de longs textes avec une précision maximale sur une machine de bureau puissante, un autre outil sera mieux adapté. Si l'objectif est de dicter sur du matériel modeste, en direct, sans jamais dépendre du réseau, vosk-cli-dictation est construit précisément pour ça.

Ce que fait Vosk

Vosk est le moteur de reconnaissance vocale utilisé par l'outil — une bibliothèque indépendante, pas un composant maison. Il s'appuie sur une architecture plus ancienne que les moteurs à base de transformeurs (type Whisper) : un modèle acoustique associé à un modèle de langage statistique, dans la lignée du logiciel Kaldi.

Cette architecture a une conséquence directe et utile : le décodage se fait au fil de l'eau, au rythme où le son arrive du microphone, avec des résultats affichés progressivement — pas par blocs de plusieurs secondes qu'il faudrait attendre. C'est ce qui donne la sensation de dictée « en direct ».

Modèle acoustique et modèle de langue

Chaque modèle téléchargé (Chapitre 1) contient deux parties, invisibles à l’usage mais utiles à comprendre :

- **Le modèle acoustique** reconnaît les sons de la parole et les associe à des unités de langage (phonèmes).
- **Le modèle de langue** évalue quelles suites de mots sont statistiquement plausibles dans la langue concernée, pour arbitrer entre plusieurs lectures possibles d’un même son.

Les modèles proposés par le projet Vosk existent en plusieurs tailles. `vosk-cli-dictation` utilise par défaut les modèles compacts (« small », de l’ordre de 50 Mo sur disque) — le bon compromis pour du matériel contraint. Rien n’empêche de déclarer un modèle plus volumineux dans la configuration (Chapitre 4) si la machine cible le permet, en échange d’une empreinte mémoire plus importante.

Rien ne quitte la machine

Aucune étape de la reconnaissance vocale ne sollicite le réseau. Le seul moment où une connexion est nécessaire est l’installation initiale du paquet (téléchargement du moteur Python) et le téléchargement manuel d’un modèle de langue — jamais pendant l’usage courant.

Chapitre 3 — Les trois états : ÉCOUTE, DICTÉE, VEILLE

`vosk-cli-dictation` a trois états distincts, chacun avec un coût CPU différent. Les confondre est la source la plus fréquente de surprise à l’usage (« pourquoi le mot de réveil ne fonctionne plus ? ») — ce chapitre les distingue clairement.

ÉCOUTE, DICTÉE et VEILLE sont les libellés que le programme affiche à l’écran. Ils restent en français quelle que soit la langue de dictée : ils identifient un état interne, pas un texte à traduire.

Vue d’ensemble

État	Comment y entrer	Micro actif	Décodage	Réveil
DICTÉE	Alt+H (ou phrase de réveil), depuis ÉCOUTE	Oui	Oui	— (dictée déjà en cours)
ÉCOUTE	Alt+H , depuis DICTÉE	Oui	Oui	Oui
VEILLE	Alt+W	Non	Non	Non

DICTÉE — dictée en cours

Ce que vous dites est reconnu et tapé dans l’application active. Pour finir et copier le résultat dans le presse-papiers : dire la phrase d’arrêt (« stop dictée »), ou appuyer sur **Alt+S**.

ÉCOUTE — en attente

C’est l’état par défaut entre deux dictées. Le microphone reste actif et le moteur continue d’écouter, pour une seule raison : pouvoir entendre la phrase de réveil à tout moment. Le coût CPU de ce mode n’est donc pas nul — c’est le prix de la réactivité vocale. Pour le réduire, le décodage se fait dans cet état avec une grammaire restreinte à la seule phrase de réveil : le moteur ne cherche rien d’autre pendant l’attente, ce qui allège nettement le travail du processeur.

Le réveil n’est pas un simple mot : il faut la **phrase complète**, et le moteur ne la retient que si la confiance de reconnaissance est suffisante. Un mot isolé (« active ») n’est jamais pris pour un réveil — c’est une protection contre les déclenchements intempestifs en pleine conversation.

Comment démarrer et arrêter une dictée

Trois façons de piloter le passage ÉCOUTE → DICTÉE → ÉCOUTE, réglables via `interaction.mode` dans la configuration (Chapitre 4) :

- **toggle** (par défaut, inchangé) : celle décrite ci-dessus — un geste démarre (**Alt+H** ou phrase de réveil), un geste distinct arrête (**Alt+S** ou phrase d’arrêt).

- **maintenue** (pousser-pour-parler) : la dictée dure exactement le temps où la touche configurée sous `hotkeys.push_to_talk` (une touche simple, pas une combinaison) reste enfoncée — on la relâche, la session s'arrête et se copie seule.
- **silence** : une pression (ou la phrase de réveil) démarre ; la dictée s'arrête et se copie seule après un silence de `interaction.silence_timeout_ms` (700 ms par défaut, réglable par langue), dès qu'au moins un mot a été capté. C'est le seul des trois modes qui laisse les mains complètement libres pendant toute la phrase.

Dans les trois modes, la phrase d'arrêt vocale et **Alt+S** / `--finalize` restent des filets de sécurité valides — aucun mode ne retire un moyen d'arrêt existant.

VEILLE — consommation quasi nulle, volontaire

Un troisième état, activé explicitement par **Alt+W**, jamais par la voix (par construction : le micro est coupé dans cet état, aucun mot ne peut donc être entendu). Capture audio et décodage vocal sont tous les deux suspendus — le coût CPU retombe à **0,0 %** et le microphone est relâché.

Ce que ça coûte : la phrase de réveil ne fonctionne plus tant que la veille est active. C'est un choix assumé, pas un défaut caché — l'outil l'affiche explicitement au moment du basculement. Seul **Alt+W** reste actif pendant la veille (il ne dépend pas du micro) pour en sortir.

Basculer en veille pendant une dictée en cours l'arrête proprement d'abord, comme si **Alt+H** avait été pressé — aucun texte n'est perdu au milieu d'une phrase.

Le seuil de confiance gouverne le déclenchement

Réveil comme arrêt ne sont retenus que si la confiance moyenne des mots de la phrase atteint le seuil configuré (0,55 par défaut, réglable par langue — Chapitre 4). Conséquences concrètes :

- Une phrase d'arrêt entendue sous le seuil est **absorbée** : elle n'est ni tapée dans le document, ni déclenchée — un indice s'affiche, et il suffit de la répéter (ou d'utiliser **Alt+S**). Ce mécanisme protège le document : une phrase ordinaire ressemblant à « stop dictée » ne s'y retrouve jamais.
- Une phrase de réveil ou d'arrêt dite **deux fois de suite** est tapée littéralement dans le texte, au lieu de déclencher — c'est voulu : la répétition immédiate est rare en commande vocale, et ce comportement évite qu'une redite soit perdue.

Quand utiliser le mode veille

Utile chaque fois que le micro doit rester coupé pour une durée notable sans quitter le programme — une réunion, un appel, un moment où la consommation CPU compte (portable sur batterie, matériel contraint). Sur une machine où le CPU importe peu, rester en ÉCOUTE en permanence est parfaitement raisonnable : c'est un choix, pas une obligation d'optimiser.

Le déroulé d'une session complète

1. **Lancer** l'outil dans un terminal visible. Il démarre en ÉCOUTE.

2. **Ouvrir l'application cible** (éditeur, navigateur, terminal) et lui donner le focus clavier.
3. **Passer en DICTÉE** : dire la phrase de réveil, ou appuyer sur **Alt+H**.
4. **Parler** normalement. Le texte s'écrit au fil de l'eau dans l'application, et les mots de ponctuation (« virgule », « point d'exclamation », « nouvelle ligne »...) sont interprétés en passant (Chapitre 4).
5. **Faire une pause** si besoin : **Alt+H** repasse en ÉCOUTE sans rien copier, le texte déjà écrit reste dans l'application. **Alt+H** de nouveau pour reprendre.
6. **Terminer** : dire la phrase d'arrêt, ou **Alt+S**, ou lier `--finalize` à un raccourci du bureau (Chapitre 5). Le texte complet de la session est alors placé dans le presse-papiers.
7. **Coller** (Ctrl+V) où on veut — le presse-papiers d'origine est restauré juste après.
8. **Mettre en veille** entre deux usages : **Alt+W** (micro relâché, 0 % CPU). **Alt+W** de nouveau pour réveiller.
9. **Quitter** : Ctrl+C dans le terminal de lancement.

Chapitre 4 — Configuration et personnalisation

Tout le comportement de vosk-cli-dictation passe par un unique fichier YAML, `config.yaml`. Ce chapitre explique où il se trouve et ce qu'on peut y changer.

Où se trouve le fichier de configuration

L'emplacement dépend du canal d'installation (Chapitre 1) et suit un ordre de priorité précis — le premier emplacement qui existe l'emporte, les suivants ne sont jamais consultés :

1. La variable d'environnement `VOSK_CONFIG`, si définie — utilisée telle quelle, jamais créée automatiquement.
2. `config/config.yaml` du dépôt, en mode clone Git.
3. `~/.config/vosk-cli-dictation/config.yaml`, en installation par paquet — créé automatiquement au premier lancement, à partir d'un modèle d'usine, avec un message indiquant où le fichier a été créé.
4. Le modèle d'usine lui-même, en lecture seule.

En pratique : avec le paquet Debian, modifier `~/.config/vosk-cli-dictation/config.yaml` — c'est un fichier personnel, jamais écrasé par une mise à jour du paquet.

Choisir la langue du gabarit de configuration

Le paquet Debian embarque quatre gabarits localisés — `config.en.yaml`, `config.fr.yaml`, `config.de.yaml`, `config.es.yaml`. Ils vivent sous `/usr/share/vosk-cli-dictation/`.

Au premier lancement, celui qui correspond à la locale système (`LANGUAGE` puis `LANG`, repli sur l'anglais si aucun des quatre ne correspond) est copié vers `~/.config/vosk-cli-dictation/config.yaml`.

Pour passer à un autre gabarit après ce premier lancement, les repérer avec `dpkg -L vosk-cli-dictation | grep config` puis remplacer son fichier personnel par celui voulu :

```
cp /usr/share/vosk-cli-dictation/config.de.yaml ~/.config/vosk-cli-dictation/config.yaml
```

En mode clone Git, les quatre gabarits sont directement dans `config/` du dépôt.

Personnaliser les raccourcis clavier

Section `hotkeys` : du fichier. Chaque raccourci se définit par une liste de touches `pynput_keys` et un nom affiché `display_name` :

```
hotkeys:
  toggle_recording:
    pynput_keys: ["alt_l", "h"]
    display_name: "[Alt+H]"
```

Trois raccourcis sont configurables ici — `toggle_recording` (Chapitre 1), `finalize_session` et `sleep_toggle` (Chapitre 3). `Ctrl+C`, qui quitte le programme, n'en fait pas partie : c'est le signal d'interruption

standard du système, toujours actif, jamais reconfigurable depuis ce fichier.

Sous Wayland, ces raccourcis n'atteignent que les fenêtres passant par XWayland (voir Chapitre 5) — pour les fenêtres Wayland natives, c'est le bureau qui porte les raccourcis, via les sous-commandes de contrôle.

Une quatrième entrée, `push_to_talk`, ne compte qu'une touche simple (pas une combinaison) : elle sert au mode pousser-pour-parler (Chapitre 3), sans effet dans les deux autres modes.

Choisir comment la dictée démarre et s'arrête

Section interaction: du fichier (Chapitre 3 pour la présentation des trois modes) :

```
interaction:
  mode: toggle          # toggle | maintenue | silence
  silence_timeout_ms: 700
```

`silence_timeout_ms` se surcharge par langue exactement comme le seuil de confiance ci-dessous, sous `language_settings.<lang>.silence_timeout_ms` — utile si un modèle décode plus lentement sur une machine donnée et déclenche un délai trop court.

Ponctuation : deux modes de saisie, par langue

Chaque langue définit, sous `language_settings.<lang>.punctuation`, sa liste de mots de ponctuation et son mode de saisie :

- **mode: inline** (par défaut) — les mots de ponctuation sont reconnus et remplacés directement au fil de la dictée : dire « virgule » insère une virgule.
- **mode: explicit** — une commande de ponctuation n'est appliquée que si elle est précédée immédiatement du mot déclencheur de la langue : il faut dire « ponctuation virgule » pour obtenir une virgule, alors que « virgule » seul s'écrit en toutes lettres. Utile quand la langue ou le vocabulaire rend les faux déclenchements fréquents.

La liste réelle des mots (ponctuation simple, double, sauts de ligne, caractères spéciaux — tiret, parenthèses, guillemets...) est affichée par le programme au lancement ; elle est définie sous les sections `simple`, `double`, `layout` et `special`. La typographie appliquée au texte (l'espace fine française avant ? ! : ; » et les guillemets français « ... »...) est réglée par le bloc `typography` de la langue.

Les commandes d'usine du français incluent les homophones du mot « tiret » : dire « tirer », « tirés », « tirait » ou « tirez » insère un tiret (–). « trait » n'en fait volontairement pas partie — « un trait de génie » doit rester inchangé. Ces alias sont modifiables comme toute commande de ponctuation.

Régler le seuil de confiance du déclenchement

Le réveil et l'arrêt ne sont retenus que si la confiance moyenne des mots de la phrase atteint le seuil `recognition.confidence_threshold` (0,55 par défaut, Chapitre 3). Il est surchargeable par langue :

```
language_settings:
```

```
fr:
  confidence_threshold: 0.55
```

Un seuil plus haut rend le déclenchement plus strict (moins de faux positifs, mais une phrase d'arrêt plus souvent absorbée); un seuil plus bas fait l'inverse.

Ajouter du vocabulaire et corriger des confusions

Deux mécanismes distincts, sous `language_settings.<lang>` :

- **custom_vocabulary** — corrections sensibles à la casse, appliquées après tout le reste. Utile pour des sigles ou noms propres que le moteur écrit systématiquement en minuscules :

```
custom_vocabulary:
  "ia": "IA"
```

- **recognition_aliases** — corrige une confusion phonétique récurrente du moteur (deux mots qui se prononcent presque pareil, Chapitre 2). Par exemple, si le moteur reconnaît systématiquement « sauna » à la place de « sonna » :

```
recognition_aliases:
  "sauna": "sonna"
```

Ces deux réglages se découvrent à l'usage, pas à l'avance — dicter, repérer une confusion récurrente, l'ajouter.

Où sont cherchés les modèles de langue

Comme pour la configuration, la recherche suit un ordre précis, le premier répertoire trouvé sur le disque l'emportant :

1. Un chemin absolu — malgré son nom, la clé `path_relative` peut aussi contenir un chemin absolu complet; dans ce cas, il prime sur tout le reste ci-dessous.
2. `~/.local/share/vosk-models/` (ou `$XDG_DATA_HOME/vosk-models/` si cette variable est définie).
3. `/usr/share/vosk-models/` — modèles installés au niveau système.
4. `vosk-model/` à la racine du projet, en mode clone Git uniquement.

Si aucun modèle n'est trouvé, le message affiché au démarrage liste précisément ces emplacements et l'adresse de téléchargement, et rappelle la commande `vosk-cli-dictation --download-model` — jamais un échec sans explication. Cette commande installe le petit modèle dans l'emplacement 2 ci-dessus (`~/.local/share/vosk-models/`), le seul où un modèle récupéré sans droits d'administrateur est ensuite retrouvé.

Chapitre 5 — Intégration système : Wayland et X11

`vosk-cli-dictation` n'a pas d'interface graphique propre — il écrit le texte reconnu directement dans l'application qui a le focus clavier, quelle qu'elle soit. Ce chapitre explique comment, et ce qui change selon le système d'affichage utilisé.

Comment le texte arrive dans l'application

Contrairement à ce qu'on pourrait imaginer, le texte n'est **pas tapé caractère par caractère**. Le mécanisme réel : le texte reconnu est placé dans le presse-papiers, puis un raccourci « coller » (`Ctrl+V`) est simulé. C'est plus fiable et plus rapide qu'une simulation de frappe complète, en particulier sur de longs passages.

Le presse-papiers d'origine est restauré juste après ce collage (`Ctrl+V`) — dicter n'écrase donc pas ce qu'on y avait copié auparavant plus d'un instant. Sous Wayland avec une application GTK, cette restauration peut arriver avant que la fenêtre ait fini de lire le presse-papiers, qui colle alors l'ancien contenu ; le réglage `clipboard.defer_restore: true` diffère la restauration à la fin de la session de dictée et supprime ce risque.

Le microphone

`vosk-cli-dictation` capte le son sur le **périphérique d'entrée par défaut du système** (PulseAudio ou PipeWire). Il n'y a pas de réglage de périphérique dans `config.yaml` : pour changer de micro, on définit le bon périphérique comme entrée par défaut — dans les réglages son du bureau, avec `pavucontrol`, ou en ligne de commande (`pactl set-default-source <nom>`, la liste étant donnée par `pactl list sources short` ou `arecord -l`).

`vosk-cli-dictation --diagnose` ouvre un flux de test sur ce périphérique et signale si l'ouverture échoue — micro absent, coupé, ou déjà utilisé par une autre application.

Les outils système utilisés, et leur choix automatique

L'injection demande **deux** outils : un simulateur de frappe et un outil de presse-papiers. Ils sont choisis automatiquement selon le type de session, sans configuration à faire :

- **Sous Wayland** : `wtype` (frappe) et `wl-clipboard` (presse-papiers), qui parlent le protocole Wayland nativement. Si `wtype` est absent, la frappe passe par `ydotool` — mais `wl-clipboard` reste nécessaire.
- **Sous X11** : `xdotool` (frappe) et `xclip` (presse-papiers).

`ydotool` est le dernier recours pour la frappe (il fonctionne dans les deux mondes via `/dev/uinput`, mais dépend de la disposition clavier pour ses codes de touches). Au moins un simulateur de frappe doit être présent — sinon l'outil refuse de démarrer et l'indique clairement. Le paquet Debian dépend de `wl-clipboard` et recommande `xclip` ; en clone Git, les installer à la main. Sans outil de presse-papiers, le texte dicté n'atteint jamais le presse-papiers : rien ne s'écrit dans l'application et un `Ctrl+V` recolle le dernier contenu copié à la main. La commande `vosk-cli-dictation --diagnose` vérifie l'environnement (type de session, outils, modèles, micro, presse-papiers, socket de contrôle) et affiche ce qui manque.

Activer ydotool (solution de repli)

Cette section ne concerne que les systèmes qui se rabattent sur ydotool, faute de wtype (Wayland) ou de xdotool (X11). Quand wtype ou xdotool est disponible, rien de ce qui suit n'est nécessaire — en particulier, il est inutile de rejoindre le groupe input. Le paquet Debian applique la même règle : son message d'installation ne propose l'ajout au groupe input que si l'on recourt au repli ydotool, et précise que xdotool et wtype n'en ont pas besoin.

Trois conditions, à vérifier dans l'ordre si le texte ne s'écrit pas :

1. **Le paquet ydotool est installé.** Sur Debian, il vit dans trixie-backports, pas dans la branche stable — activer ce dépôt si nécessaire avant l'installation.
2. **Le service ydotoold tourne.** ydotool a besoin de ce démon en arrière-plan pour fonctionner.
3. **L'utilisateur appartient au groupe input.** C'est ce groupe qui donne accès à /dev/uinput, sans quoi ydotool échoue silencieusement pour cet utilisateur :

```
sudo usermod -aG input $USER
```

Se déconnecter puis se reconnecter est nécessaire pour que ce changement de groupe prenne effet — un redémarrage du terminal seul ne suffit pas.

Raccourcis clavier : X11 et Wayland ne sont pas égaux

Les raccourcis intégrés **Alt+H**, **Alt+S**, **Alt+W** sont écoutés par l'outil lui-même via la bibliothèque pynput, qui observe le serveur X. Sous Wayland, elle n'entend que les fenêtres passant par **XWayland** (beaucoup d'applications Electron et anciennes), jamais une fenêtre **Wayland native** : pour ces dernières, ce sont les raccourcis de l'environnement de bureau qui doivent piloter l'outil.

Chaque environnement de bureau peut lier une combinaison de touches à une commande. Côté vosk-cli-dictation, les sous-commandes de contrôle à utiliser sont :

Action voulue	Sous-commande
Démarrer / mettre en pause la dictée	<code>vosk-cli-dictation --toggle</code>
Terminer la dictée et copier le texte	<code>vosk-cli-dictation --finalize</code>
Entrer / sortir de la veille	<code>vosk-cli-dictation --sleep-toggle</code>
Connaître l'état courant	<code>vosk-cli-dictation --status</code>
Annuler la dictée en cours sans rien taper	<code>vosk-cli-dictation --cancel</code>
Quitter la session proprement	<code>vosk-cli-dictation --stop</code>

Exemple sous KDE Plasma : dans *Paramètres système* → *Raccourcis* → *Raccourcis personnalisés*, créer une nouvelle entrée « Commande/URL » dont la commande est `vosk-cli-dictation --finalize`, et lui attribuer une combinaison (par exemple **Meta+S**). La sous-commande se connecte à la session en cours, exécute l'action et affiche l'état résultant (OK DICTÉE, OK ÉCOUTE, OK VEILLE).

Sous X11 également, ces sous-commandes sont le moyen le plus fiable : un raccourci pynput n'est pas consommé — la combinaison atteint aussi la fenêtre ciblée, et pressée dans un terminal elle y insère une séquence parasite (par exemple `^[s` pour `Alt+S`). Lier `--finalize` (ou `--toggle`, `--sleep-toggle`) à un raccourci du bureau donne un arrêt propre depuis n'importe quelle fenêtre.

Que se passe-t-il si aucun outil d'injection n'est disponible

L'outil ne démarre pas et l'indique explicitement, avec une suggestion d'installation. Il n'y a pas de mode dégradé « sans injection » : sans `wtype`/`xdotool`/`ydotool`, il n'existe aucun moyen fiable d'écrire le texte reconnu dans une autre application.

Ce contrôle au démarrage ne porte que sur le simulateur de frappe. Un outil de presse-papiers manquant (`wl-clipboard` sous Wayland, `xclip` sous X11) ne bloque pas le démarrage mais casse l'injection : le texte dicté n'apparaît que dans le terminal et chaque `Ctrl+V` recolle le dernier contenu copié à la main. `vosk-cli-dictation --diagnose` le signale.

Chapitre 6 — Dépannage

Symptôme → cause probable → action

Symptôme	Cause probable	Action
Le programme refuse de démarrer, message sur les outils d'injection	Aucun outil d'injection (<code>wtype/xdotool/ymdotool</code>) n'est installé ou fonctionnel	Installer l'un d'eux — voir Chapitre 5
« No speech recognition model could be loaded » au démarrage	Aucun modèle trouvé aux emplacements attendus	Le message liste les emplacements cherchés et l'URL de téléchargement — voir Chapitre 1
Rien n'est reconnu, le micro semble muet	Mauvais périphérique d'entrée par défaut, ou micro coupé au niveau système	Vérifier / changer l'entrée par défaut (réglages son, <code>pavucontrol</code>); <code>vosk-cli-dictation</code> <code>--diagnose</code> — voir Chapitre 5
<code>--diagnose</code> signale l'échec d'ouverture du flux audio	Périphérique occupé par une autre application, ou droits d'accès au micro insuffisants	Fermer l'autre application; vérifier les permissions du périphérique — voir Chapitre 5
Rien ne s'écrit dans l'application, mais aucune erreur affichée	L'outil s'est rabattu sur <code>ymdotool</code> , mais l'utilisateur n'est pas dans le groupe <code>input</code>	Vérifier l'appartenance au groupe, se reconnecter — voir Chapitre 5
L'application recolle toujours le dernier contenu copié à la main; le texte dicté n'apparaît que dans le terminal	Aucun outil de presse-papiers installé (<code>wl-clipboard</code> sous Wayland, <code>xclip</code> sous X11); le texte dicté n'atteint jamais le presse-papiers	L'installer. Le paquet Debian $\geq 0.5.1$ dépend de <code>wl-clipboard</code> et recommande <code>xclip</code> ; en clone Git, les installer à la main — voir Chapitre 5
Sous Wayland avec une application GTK, la fenêtre colle par instant l'ancien contenu du presse-papiers	La restauration du presse-papiers arrive avant que la fenêtre ait fini de lire la sélection	<code>clipboard.defer_restore: true</code> dans la config — la restauration est alors différée à la fin de la session — voir Chapitre 5
Aucun son de notification	<code>paplay</code> absent, ou aucun fichier son configuré	Avertissement bénin, sans impact sur la reconnaissance — installer <code>pulseaudio-utils</code> si le son est souhaité
Un mot isolé (« active ») peut-il déclencher la dictée par erreur ?	Non : le réveil exige la phrase complète	Le réveil ne se fait que sur « active dictée » entier — rien à corriger
La phrase de réveil ne démarre pas la dictée	Mode veille actif (<code>Alt+W</code>), ou phrase incomplète, ou confiance sous le seuil	Sortir de veille avec <code>Alt+W</code> ; prononcer la phrase complète; répéter si besoin — voir Chapitre 3

Symptôme	Cause probable	Action
« stop dictée » n'arrête pas la dictée, et rien n'est tapé	Phrase d'arrêt sous le seuil de confiance — absorbée par sécurité	La répéter, ou utiliser Alt+S / --finalize — voir Chapitre 3
La phrase de réveil ou d'arrêt s'écrit dans le texte au lieu de déclencher	Phrase dite deux fois de suite — comportement voulu	La dire une seule fois, après un court silence — voir Chapitre 3
Alt+H / Alt+S / Alt+W ne répondent pas	Fenêtre Wayland native au premier plan — pynput ne voit que les fenêtres X11 et XWayland	Lier --toggle, --finalize, --sleep-toggle aux raccourcis du bureau — voir Chapitre 5
Alt+S insère ^[s dans le terminal	Comportement connu : le raccourci n'est pas consommé par l'outil	Préférer un raccourci du bureau lié à --finalize — voir Chapitre 5
Alt+H ne copie rien dans le presse-papiers	Comportement normal — Alt+H met en pause, il ne finalise pas	Utiliser Alt+S ou la phrase d'arrêt pour copier — voir Chapitre 1
Un mot est systématiquement mal reconnu	Confusion phonétique du moteur (mots proches à l'oreille)	Ajouter une correction dans recognition_aliases — voir Chapitre 4
L'outil termine la session de lui-même : « le moteur de reconnaissance ne se rétablit pas »	Le décodeur a renvoyé des trames illisibles plusieurs fois de suite, la reconstruction du moteur n'a rien changé	Relancer l'outil. Si cela se répète, joindre à un rapport le fichier écrit sous ~/.local/state/vosk-cli-dictation/diagnostics/ (voir ci-dessous)
La configuration modifiée ne semble jamais prise en compte	Le fichier édité n'est pas celui réellement utilisé (plusieurs emplacements possibles)	Vérifier l'ordre de résolution — voir Chapitre 4

Où chercher un message d'erreur complet

vosk-cli-dictation affiche ses erreurs directement dans le terminal où il a été lancé. Relancer depuis un terminal visible (pas en arrière-plan) est la première chose à faire pour diagnostiquer un problème non listé ci-dessus. En cas de doute sur l'environnement, `vosk-cli-dictation --diagnose` vérifie tout (outils, modèles, micro, presse-papiers, socket de contrôle) et affiche l'état de chaque point.

Un seul cas écrit un fichier : quand le décodeur renvoie une sortie illisible, la chaîne brute est enregistrée une fois par session sous `~/.local/state/vosk-cli-dictation/diagnostics/`. Ce fichier est utile à joindre à un rapport de problème de reconnaissance.

Si le problème persiste

Le dépôt du projet a une page de suivi des problèmes publique : <https://github.com/RonanDavalan/vosk-cli-dictation/issues>