WhispSkrid — User manual

Install and use offline push-to-talk dictation

Version v1.0.1
2026-09-18

Contents

Chapter 1 — Installation

In two minutes

For those who want to dictate immediately, without reading the rest.

1. Install the native package for your distribution.

```
# Debian 12 et 13, Ubuntu 22.04 et 24.04 et leurs dérivées
sudo apt install ./whispskrid_1.0.1_all.deb

# Fedora 42
sudo dnf install ./whispskrid-1.0.1-1.fc42.noarch.rpm

# openSUSE Leap 15.6
sudo zypper install ./whispskrid-1.0.1-1.leap156.noarch.rpm

# Arch Linux
sudo pacman -U whispskrid-1.0.1-1-any.pkg.tar.zst
```

2. Retrieve a Whisper model, if one is not present (the package does not include one — it's too large).

```
whispskrid --download-model base
```

The `base` model is downloaded, verified, and then stored in `~/.local/share/whispskrid/whisper-models/`. When the tool is launched for the first time without any models, it prompts the user to download the model interactively.

3. Launch the tool.

```
whispskrid
```

It displays a "ready" banner and waits. Hold down the **right Shift** key, speak, then release: the transcribed text will appear in the active window.

That's all for the introduction. The following sections explain how to install from source, change the model or language, and control the tool using the desktop keyboard (Chapter 2).

Prerequisites

- **A Linux system** with an X11 or Wayland graphical session. Without a display server, there is no active window to write the transcribed text.
- **Python 3** and a virtual environment (`python3-venv`).
- **A text injection tool**: `ydotool` or `xdotool` for typing, `wl-clipboard` under Wayland or `xclip` under X11 for the clipboard. If a functional typing engine is not available, the tool will still start, but will operate in a degraded mode (the transcribed text remains displayed in the terminal, without injection - Chapter 3).

- **A microphone recognized by the system** as the default input device (Chapter 3).
- **Disk space for the model**: approximately 145 MB for the model `base`, more for larger models (Chapter 2). During operation, a session keeps the model in memory - approximately 150 to 300 MB of RAM for `base`.

The native package declares and installs all of its dependencies, both system and Python, regardless of the distribution. In Git clone mode, `.venv/bin/pip install -e .` installs the Python libraries; the injection tool (`ydotool` or `xdotool`, plus `wl-clipboard` or `xclip`) then needs to be installed separately.

Two ways to install

WhispSkrid offers two channels, and neither replaces the other.

The native package (recommended for general use)

```
# Debian 12 et 13, Ubuntu 22.04 et 24.04 et leurs dérivées
sudo apt install ./whispskrid_1.0.1_all.deb

# Fedora 42
sudo dnf install ./whispskrid-1.0.1-1.fc42.noarch.rpm

# openSUSE Leap 15.6
sudo zypper install ./whispskrid-1.0.1-1.leap156.noarch.rpm

# Arch Linux
sudo pacman -U whispskrid-1.0.1-1-any.pkg.tar.zst
```

The package installs a single command, `whispskrid`, in `/usr/bin/`. It also creates an isolated Python environment and installs the transcription engine (`faster-whisper`) within it—this step requires network access during installation, only once. No superuser privileges are required afterward.

Each package was installed, checked by `whispskrid --diagnose` and then uninstalled cleanly within a container for each of the versions above (Debian 12 and 13, Ubuntu 22.04 and 24.04, Fedora 42, openSUSE Leap 15.6, Arch Linux). The Debian package was also validated in real-world usage, including speech recognition, on several physical machines, one of which was running Debian 13; the other packages have not yet been tested in a real graphical session. The other systems are not covered—not by principle, simply the scope that was tested.

The installation itself never modifies personal files: no existing configurations are overwritten, and no user account directories are created or touched at this stage. It's only upon the first launch, and when recovering a template, that files appear in the personal directory—never before, and never without an explicit action from the user.

Clone the Git repository (to track development or contribute)

```
git clone https://github.com/RonanDavalan/whispskrid.git
cd whispskrid
python3 -m venv .venv
```

```
.venv/bin/pip install -e .
.venv/bin/whispskrid --diagnose
```

In this mode, the tool is launched via `.venv/bin/whispskrid` instead of the `whispskrid` command from the package, and its configuration is stored in `config/config.yaml`, alongside the code—never in the user's home directory. The two modes do not mix: a Git clone ignores the package configuration, and vice versa.

Retrieve a model

Neither of the channels includes a transcription model – they weigh approximately 75 MB (`tiny`) to several gigabytes (`large-v3`), which is too large to be included with the tool.

The simplest method, when installing via a package, is to let the tool handle it.

```
whispskrid --download-model base
```

This command retrieves the requested model—`base`, by default, or one of `tiny`, `small`, `medium`, `large-v3` (and their variants `.en`, specialized English)—verifies that it loads, installs it in `~/.local/share/whispskrid/whisper-models/`, and then stops. On the very first launch without any model, the same suggestion is made interactively.

The model and the language are two distinct choices: a Whisper model is multilingual, and it's the `-l` (or `default_language` in the configuration, Chapter 2) that sets the recognition language. Without it, Whisper detects the language automatically.

If no model is found upon launching, the tool displays the locations searched and the steps to follow—never a silent failure.

First start

```
whispskrid
```

(In clone Git mode, the same command is `.venv/bin/whispskrid`, starting from the root of the repository.)

Upon launching, the tool loads the configuration, loads the model once and for all, opens the microphone, and then displays a "ready" banner and waits. This process remains in the foreground: this is the persistent session. It keeps the model in memory so that each dictation starts without a loading delay (Chapter 2). Only one session can run at a time.

To exit: **Ctrl+C** in the launch terminal, or `whispskrid --stop` from another terminal.

A first dictation

WhispSkrid works with a "press-to-talk" function: you hold down a button while you're speaking, and release it when you're finished.

1. Give keyboard focus to the target application (editor, browser, terminal).
2. **Hold down the right Shift key**. A short sound confirms the start of the recording.

3. **Speak** normally as long as the key is held down.
4. **Release**. The captured audio is sent for transcription; the resulting text is inserted into the active window.

There is no wake-up phrase, no waiting state, and no timer: the capture lasts exactly as long as the button is held down. Each release triggers an independent injection.

Under a **Wayland** session, this key is only captured for windows that are running through XWayland. For a native Wayland window, you need to bind a control command to a desktop shortcut (Chapter 2).

Uninstall and update

Native package. `sudo apt remove whispskrid` (Debian), `sudo dnf remove whispskrid` (Fedora), `sudo zypper remove whispskrid` (openSUSE) ou `sudo pacman -R whispskrid` (Arch) removes the command and the isolated Python environment `/usr/lib/whispskrid/venv`. On Debian, `sudo apt purge whispskrid` it also removes the factory configuration template. The personal configuration file `~/.config/whispskrid/config.yaml` and the templates `~/.local/share/whispskrid/whisper-models/` are never affected by the uninstallation on any distribution; they can be deleted manually if desired.

Clone Git. Delete the cloned directory: it contains everything (code, virtual environment `venv/`, any models in `whisper-models/`). Nothing is written elsewhere in this mode.

Update. With the package, install the most recent file using the same command as for installation (`apt install`, `dnf install`, `zypper install` ou `pacman -U`): it replaces the version in place without affecting your personal configuration. With the Git clone, run `git pull`, and then run `.venv/bin/pip install -e .` again to reflect any new dependencies.

Chapter 2 — Using

For whom is this tool?

WhispSkrid is an online and offline command-line dictation tool built around Whisper. It aims for a simple and predictable usage: you hold down a key, you speak, you release, and the text appears in the active window. That's it — no wake-up phrases, no voice commands, no waiting states to manage.

The transcription is powered by `faster-whisper`, a Whisper implementation based on CTranslate2, optimized for running on the CPU. The default model is `base`, a good balance of speed and accuracy for most machines; it can be changed at any time (see below). All speech recognition is local: no audio or text leaves the machine, at any time.

The interaction model: strict "press-to-talk" mode

A dictation is a self-contained episode, with no memory of one episode to the next.

1. **Press and Hold**: Press and hold the "press-to-talk" button. The audio recording starts; a short beep confirms it.
2. **Speaking**: The audio is recorded as long as the button is held. There is no automatic cutoff, timer, or sentence-end detection.
3. **Release**: The recording stops. The complete audio buffer is transcribed in a single pass.
4. **Injection**: The transcribed text, after a slight post-processing, is injected into the active window (Chapter 3 for the mechanism).

There is no "accumulated" text between two dictation sessions, so there is no command to "copy the session": each session produces its own injection.

The duration safeguard

Automatic stop: `capture.max_seconds` (default 300 seconds). If the key is held down for longer, the capture stops automatically, what has been recorded is transcribed and injected normally, and a warning is logged. This is not a comfort timer – it is a protection against a stuck key or an audio buffer that inflates endlessly.

The mute function is not enabled

Automatic speech detection (VAD) is not yet available. The keys `vad.enabled` (default `false`) and `vad.silence_ms` are already present in the configuration; `vad.enabled: true` displays a "not yet implemented" warning, and the tool continues in strict push-to-talk mode.

Two ways to initiate a dictation

The local key

A hotkey listener (`pynput`) monitors the keyboard. The default key is the **right Shift** (`shift_r`), with no default global hotkey under common desktops, making it easy to hold. It can be changed through `hotkeys.push_to_talk` in the configuration. In this mode, the hold is native: the capture lasts exactly from the press to the release.

`pynput` observes server X. Under Wayland, it only captures windows that go through XWayland, never a native Wayland window—it's a best-effort convenience. To never start this listener, set `hotkeys.pynput_enabled: false` and control it through the following method.

The control subcommands

Universal pathway, independent of the session type: these sub-commands are linked to the global desktop shortcuts. Since a desktop shortcut does not transmit the concept of "key held / key released," this pathway works in a toggle mode:

Action	Subcommand
Start recording	`whispskrid --dictate`
Stop capturing and inject	`whispskrid --dictate-stop`
Start or stop according to the status	`whispskrid --toggle`
Discard the current capture without injecting	`whispskrid --cancel`
Know the current status	`whispskrid --status`
End the current session	`whispskrid --stop`

Example under KDE Plasma: In *System Settings* → *Shortcuts* → *Custom Shortcuts*, create an entry with a "Command/URL" field where the command is `whispskrid --toggle`, and assign a key combination to it. A first press starts the capture, a second press stops it and injects the result.

The two pathways control the same internal state. A capture started by a key press can be canceled by `--cancel`, and vice versa.

Persistent session and client

`whispskrid` Launched without any control arguments, it starts a **persistent** session in the foreground: it loads the configuration and the model, opens the microphone, starts the key listener and the control socket, and then waits.

`whispskrid` is launched with a control argument (`--dictate`, `--status`, `--stop`…), and acts as a **client**: it connects to the current session via a Unix socket (at `$XDG_RUNTIME_DIR`), sends the command, displays the response (`OK` or `ERR`), and exits. If there is no current session, it indicates this and exits with an error.

Only one persistent session can run at a time: if the socket is already responding, a new instance will refuse to start.

Configuration

All the behavior is defined through a single YAML file, `config.yaml`.

Where is the file?

The location follows a priority order – the first one found takes precedence, and the others are never consulted.

1. The environment variable `WHISPSKRID_CONFIG`, if defined, is used as is and is never created automatically.
2. `config/config.yaml` from the repository, in Git clone mode.
3. `~/.config/whispskrid/config.yaml`, during package installation — created automatically on the first launch from a factory template, with a message indicating where the file was created.
4. The factory template itself is read-only.

In practice, whichever native package is installed, we modify `~/.config/whispskrid/config.yaml` - a personal file that is never overwritten by a package update.

Change model

```
models:
  default: "base"
  device: "auto"
  compute_type: "auto"
```

`models.default` is a short name resolved in the models folder, or an absolute path. `device` can be `cpu`, `cuda`, or `auto`.

`compute_type` accepts four values: `int8`, `int8_float16`, `float16`, and `auto`. With `auto`, the choice is made automatically: `int8` on the processor, `float16` on the graphics card.

`--model NOM` on the command line overrides `models.default` for a session. A larger model transcribes more accurately, at the cost of a larger memory footprint and higher latency.

Force the language

```
default_language: ""
```

As you can see, the interface uses the system's language, and the recognition automatically detects the language. `-l fr` (or `--lang fr`) on the command line forces the language of the interface and the recognition for a single session, without modifying the file.

The push-to-talk button

```
hotkeys:
  pynput_enabled: true
  push_to_talk: ["shift_r"]
  min_hold_ms: 250
  mode: "hold"
```

A touch of functionality is also suitable (`["f4"]`, protected from desk shortcuts). Listing multiple keys creates a **combination**: all must be held down together to start dictation, and in `hold` mode, releasing any one of them stops and injects. `min_hold_ms` cancels without transcribing a brief press — prevents accidental activation that would start a recording on silence, which Whisper might misinterpret; has no effect outside `hold`. `mode: "toggle"` changes the semantics: a press starts recording, the next press stops and injects, releasing it does nothing. `mode: "armed"` (listening for spoken phrases) requires models that

are not provided: attempting to use it will fail as long as they are missing.

The duration safeguard and post-processing

```
capture:
  max_seconds: 300

post_processing:
  trim: true
  capitalize_sentence_start: true
```

`trim` It trims the leading and trailing whitespace of the transcribed text. `capitalize_sentence_start` It forces a capital letter on the first character inserted. Whisper itself punctuates and capitalizes sentences. Post-processing is limited to these two settings; there is no substitution table, no language-specific rule, and no insertion of thin spaces.

Copying and pasting under Wayland

```
clipboard:
  defer_restore: false
```

In `true`, the clipboard is only restored at the end of the session, instead of after each dictation. This is useful with certain GTK applications under Wayland (Chapter 3).

Manage models

The models reside in a managed folder:

- Installed via package: `/usr/share/whispskrid/whisper-models/` (read-only), supplemented by `~/.local/share/whispskrid/whisper-models/` for user downloads;
- From source: `whisper-models/` at the root of the repository.

`whispskrid --download-model [NOM]` Downloads a model to the user folder and checks that it loads. Order of magnitude of the disk footprint in quantization `int8` :

```
tiny        75 Mo
base       145 Mo
small      480 Mo
medium     1,5 Go
large-v3   3 Go
```

In RAM, at peak usage: for CPU `tiny` ~0.4 GB, `base` ~0.5–0.6 GB, `small` ~1.3 GB, `medium` ~3.5–3.7 GB, `large-v3` ~6.8 GB; for GPU (x86) `tiny` ~0.2 GB, `base` ~0.3 GB, `small` ~0.7 GB, `medium` ~1.7–2 GB, `large-v3` ~3.2–3.9 GB of VRAM. Ranges measured on five machines, four languages - values per model. In practice: `tiny` and `base` follow real-time everywhere, including on ARM; `small` lags on ARM (2–3 times slower); `medium` requires x86; `large-v3` requires x86 with ~8 GB of RAM and only works on GPU (on CPU, even x86 exceeds real-time).

The tool works offline once the model is present: the Hugging Face cache is only a last resort, not the normal way to use it.

Command-line summary

```
whispskrid [-l LANG] [--model NOM]
whispskrid --dictate | --dictate-stop | --toggle | --cancel
whispskrid --status | --stop
whispskrid --diagnose
whispskrid --download-model [NOM]
whispskrid --help | --version
```

Chapter 3 — Troubleshooting

How does the text get into the application?

WhispSkrid does not have a graphical interface; it writes the transcribed text directly into the application that currently has keyboard focus, regardless of which application it is.

The text is **not typed character by character**. The actual mechanism: the transcribed text is placed in the clipboard, and then a "paste" shortcut (`Ctrl+V`) is simulated. This is more reliable and faster than simulating a complete typing process, especially for long passages.

The original clipboard is restored immediately after pasting — dictation does not overwrite what was copied, so it is not permanent. Under Wayland with a GTK application, this restoration can occur before the window has finished reading the clipboard, which then pastes the old content. The setting `clipboard.defer_restore: true` delays the restoration until the end of the session and eliminates this risk.

The microphone

WhispSkrid captures audio from the **system's default input device** (PulseAudio or PipeWire). There is no device setting within `config.yaml` : to change the microphone, you set the desired device as the default input — in the desktop's sound settings, with `pavucontrol`, or via the command line (`pactl set-default-source <nom>`, the list being provided by `pactl list sources short` or `arecord -l`).

`whispskrid --diagnose` opens a test stream on this device and reports whether the opening fails—e.g., the microphone is missing, muted, or already in use by another application.

Injection tools and their automatic selection

The injection process requires **two** tools: a typing engine and a clipboard tool. The typing engine is selected only once at the start, through a direct query — and is never retested before each copy-paste operation.

- **First choice**: `ydotool` (strikes at the driver level via the daemon `ydotoold`, key codes depend on keyboard layout).
- **Otherwise**: `xdotool` (standard path, with window detection).
- **Otherwise**: degraded mode - the transcribed text remains displayed in the terminal, and typing is disabled for the session, rather than a silent failure for each dictation.

The clipboard behavior depends on the session type: `wl-clipboard` under Wayland (always required, even with `ydotool`), `xclip` under X11. Without a clipboard tool, the transcribed text never reaches the clipboard; nothing is written to the application, and a `Ctrl+V` re-pastes the last content copied manually.

Activate ydotool (fallback solution)

This section applies only to systems where the selected engine is `ydotool`. When `xdotool` handles the typing, nothing that follows is necessary—in particular, it is unnecessary to join the group `input`.

Three conditions to check, in order, if the text is not displaying correctly:

1. **The package `ydotool` is installed**. On Debian, it resides in `trixie-backports`, not in the stable branch. Enable this repository if necessary.

2. **The service `ydotoold` is running**. `ydotool` needs this daemon in the background.

3. **The user belongs to the group `input`**. It is this group that provides access to `/dev/uinput`, otherwise `ydotool` will fail silently:

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

Disconnecting and then reconnecting is necessary for this group change to take effect.

Shortcuts: X11 and Wayland are not the same

The built-in push-to-talk button is listened to by the tool itself via `pynput`, which monitors the X server. Under Wayland, `pynput` only hears windows that are passing through **XWayland** (many Electron applications and older ones), but never a **native Wayland** window. For the latter, you need to bind a control subcommand (`--toggle`, `--dictate`, `--dictate-stop`) to a desktop environment shortcut (Chapter 2).

`--diagnose`

`whispskrid --diagnose` checks the environment and exits — exit code 0 if everything passes, non-zero otherwise. Each point is rendered on line `ok` / `!!` :

- **Environment** — session type (X11 / Wayland), injection backend retained.
- **Tools** — presence of `ydotool`, `ydotoold`, `xdotool`, `wl-copy` / `wl-paste` or `xclip`, `paplay` ; read-write access to `/dev/uinput`.
- **Backend** — import of `faster_whisper` and `ctranslate2` ; `device` and `compute_type` effective; CUDA available or not (informative).
- **Models** — resolved models directory; presence of the default model; test loading (memory allocation then release).
- **Audio input** — opening a capture stream in the expected format.
- **Clipboard** — read/write round trip.
- **Control socket** — path, executable directory accessible in write mode, session active or not.
- **Configuration** — path of the actually loaded `config.yaml`.

Symptom → probable cause → action

Symptom	Probable cause	Action
The transcribed text remains in the terminal; nothing is injected	No functional typing engines (neither `ydotool` nor `xdotool`): degraded mode	Install either `ydotool` or `xdotool`, then restart the session

Symptom	Probable cause	Action
"No model found" message at startup	No model found at the expected locations	`whispskrid --download-model base` — see Chapter 1
Nothing is being transcribed, the microphone seems to be muted	Bad default input device, or microphone muted at the system level	Check / change the default input (`pavucontrol`); `whispskrid --diagnose`
`--diagnose` indicates a failure to open the audio stream	Device in use by another application, or insufficient access rights	Close the other application; check the device permissions
Nothing is being written to the application, and no errors are being displayed	The tool collapsed onto `ydotool`, but the user is not in the group `input`	Join the group, reconnect - see above
The application always pastes the last content that was manually copied	No clipboard tools installed (`wl-clipboard` under Wayland, `xclip` under X11)	Install it
Under Wayland, with a GTK application, the window temporarily displays the previous content	The clipboard restoration happens before the window has finished reading the selection	`clipboard.defer_restore: true` in the configuration
The right Shift key is not working	Native Wayland window in the foreground — `pynput` only sees X11 and XWayland windows	Link `--toggle` / `--dictate` to a desktop shortcut - see Chapter 2
No sound at the beginning of the recording	`paplay` absent, or no sound file configured	Minor warning: install `pulseaudio-utils` if sound is desired
The recording stops automatically after five minutes	Guardrail `capture.max_seconds` reached	Desired behavior; adjust the value in the configuration if necessary — see Chapter 2
`vad.enabled: true` without effect	Silence detection is not implemented in this version	Reserved setting; the tool continues to operate in strict push-to-talk mode—see Chapter 2
The transcription is slow	The model is too large for the machine, or `compute_type` is unsuitable	Switch to `base` or `small`; leave `compute_type: auto` — see Chapter 2
The modified configuration doesn't seem to be taken into account	The edited file is not the one that was actually used	Check the order of operations — see Chapter 2
"A session is already in progress" at startup	A persistent session is already running (only one at a time)	`whispskrid --stop`, or use the existing session

Where to find an error message

WhispSkrid displays its errors directly in the terminal where the persistent session was started. Restarting from a visible terminal is the first thing to do to diagnose a problem not listed here. If in doubt, `whispskrid --diagnose` checks everything and displays the status of each point.

If the problem persists

The project repository has a public tracking page: `https://github.com/RonanDavalan/whispskrid/issues`