macOS companion app

Voice Wake

Modes

Wake-word mode (default): always-on Speech recognizer waits for trigger tokens (`swabbleTriggerWords`). On match it starts capture, shows the overlay with partial text, and auto-sends after silence.

Push-to-talk (Right Option hold): hold the right Option key to capture immediately—no trigger needed. The overlay appears while held; releasing finalizes and forwards after a short delay so you can tweak text.

Runtime behavior (wake-word)

Speech recognizer lives in `VoiceWakeRuntime` .

Trigger only fires when there's a **meaningful pause** between the wake word and the next word (~0.55s gap). The overlay/chime can start on the pause even before the command begins.

Silence windows: 2.0s when speech is flowing, 5.0s if only the trigger was heard.

Hard stop: 120s to prevent runaway sessions.

Debounce between sessions: 350ms.

Overlay is driven via `VoiceWakeOverlayController` with committed/volatile coloring.

After send, recognizer restarts cleanly to listen for the next trigger.

›

Lifecycle invariants

If Voice Wake is enabled and permissions are granted, the wake-word recognizer should be listening (except during an explicit push-to-talk capture).

Overlay visibility (including manual dismiss via the X button) must never prevent the recognizer from resuming.

Sticky overlay failure mode (previous)

Previously, if the overlay got stuck visible and you manually closed it, Voice Wake could appear “dead” because the runtime’s restart attempt could be blocked by overlay visibility and no subsequent restart was scheduled.

Hardening:

Wake runtime restart is no longer blocked by overlay visibility.

Overlay dismiss completion triggers a `VoiceWakeRuntime.refresh(...)` via `VoiceSessionCoordinator`, so manual X-dismiss always resumes listening.

Push-to-talk specifics

Hotkey detection uses a global `.flagsChanged` monitor for **right Option** (`keyCode 61` + `.option`). We only observe events (no swallowing).

Capture pipeline lives in `VoicePushToTalk`: starts Speech immediately, streams partials to the overlay, and calls `VoiceWakeForwarder` on release.

When push-to-talk starts we pause the wake-word runtime to avoid dueling audio taps; it restarts automatically after release.

Permissions: requires Microphone + Speech; seeing events needs Accessibility/Input Monitoring approval.

External keyboards: some may not expose right Option as expected—offer a fallback shortcut if users report misses.

User-facing settings

Voice Wake toggle: enables wake-word runtime.

Hold Cmd+Fn to talk: enables the push-to-talk monitor. Disabled on macOS < 26.

Language & mic pickers, live level meter, trigger-word table, tester (local-only; does not forward).

Mic picker preserves the last selection if a device disconnects, shows a disconnected hint, and temporarily falls back to the system default until it returns.

Sounds: chimes on trigger detect and on send; defaults to the macOS “Glass” system sound. You can pick any `NSSound` -loadable file (e.g. MP3/WAV/AIFF) for each event or choose **No Sound**.

Forwarding behavior

When Voice Wake is enabled, transcripts are forwarded to the active gateway/agent (the same local vs remote mode used by the rest of the mac app).

Replies are delivered to the **last-used main provider** (WhatsApp/Telegram/Discord/WebChat). If delivery fails, the error is logged and the run is still visible via WebChat/session logs.

Forwarding payload

`VoiceWakeForwarder.prefixedTranscript(_:)` prepends the machine hint before sending. Shared between wake-word and push-to-talk paths.

Quick verification

Toggle push-to-talk on, hold Cmd+Fn, speak, release: overlay should show partials then send.

While holding, menu-bar ears should stay enlarged (uses `triggerVoiceEars(ttl:nil)`); they drop after release.

‹ Menu Bar Voice Overlay ›