macOS companion app

Voice Overlay

Audience: macOS app contributors. Goal: keep the voice overlay predictable when wake-word and push-to-talk overlap.

Current intent

If the overlay is already visible from wake-word and the user presses the hotkey, the hotkey session *adopts* the existing text instead of resetting it. The overlay stays up while the hotkey is held. When the user releases: send if there is trimmed text, otherwise dismiss.

Wake-word alone still auto-sends on silence; push-to-talk sends immediately on release.

Implemented (Dec 9, 2025)

Overlay sessions now carry a token per capture (wake-word or push-to-talk). Partial/final/send/dismiss/level updates are dropped when the token doesn't match, avoiding stale callbacks.

Push-to-talk adopts any visible overlay text as a prefix (so pressing the hotkey while the wake overlay is up keeps the text and appends new speech). It waits up to 1.5s for a final transcript before falling back to the current text.

Chime/overlay logging is emitted at `info` in categories `voicewake.overlay` , `voicewake.ptt` , and `voicewake.chime` (session start, partial, final, send, dismiss, chime reason).

Next steps

1. **VoiceSessionCoordinator (actor)**

   Owns exactly one `VoiceSession` at a time.

   API (token-based): `beginWakeCapture` , `beginPushToTalk` , `updatePartial` , `endCapture` , `cancel` , `applyCooldown` .

   Drops callbacks that carry stale tokens (prevents old recognizers from reopening the overlay).

2. **VoiceSession (model)**

   Fields: `token` , `source` (wakeWord|pushToTalk), committed/volatile text, chime flags, timers (auto-send, idle), `overlayMode` (display|editing|sending), cooldown deadline.

3. **Overlay binding**

   `VoiceSessionPublisher` (`ObservableObject`) mirrors the active session into SwiftUI.

   `VoiceWakeOverlayView` renders only via the publisher; it never mutates global singletons directly.

   Overlay user actions (`sendNow` , `dismiss` , `edit`) call back into the coordinator with the session token.

4. **Unified send path**

   On `endCapture` : if trimmed text is empty → dismiss; else `performSend(session:)` (plays send chime once, forwards, dismisses).

   Push-to-talk: no delay; wake-word: optional delay for auto-send.

   Apply a short cooldown to the wake runtime after push-to-talk finishes so wake-word doesn’t immediately retrigger.

5. **Logging**

   Coordinator emits `.info` logs in subsystem `bot.molt` , categories `voicewake.overlay` and `voicewake.chime` .

   Key events: `session_started` , `adopted_by_push_to_talk` , `partial` , `finalized` , `send` , `dismiss` , `cancel` , `cooldown` .

Debugging checklist

Stream logs while reproducing a sticky overlay:

```
sudo log stream --predicate 'subsystem == "bot.molt" AND category
```

Verify only one active session token; stale callbacks should be dropped by the coordinator.

Ensure push-to-talk release always calls `endCapture` with the active token; if text is empty, expect `dismiss` without chime or send.

Migration steps (suggested)

1. Add `VoiceSessionCoordinator` , `VoiceSession` , and `VoiceSessionPublisher` .
2. Refactor `VoiceWakeRuntime` to create/update/end sessions instead of touching `VoiceWakeOverlayController` directly.
3. Refactor `VoicePushToTalk` to adopt existing sessions and call `endCapture` on release; apply runtime cooldown.
4. Wire `VoiceWakeOverlayController` to the publisher; remove direct calls from runtime/PTT.
5. Add integration tests for session adoption, cooldown, and empty-text dismissal.

‹ Voice Wake WebChat ›