Loralei - Speaker Hydration on Cold Boot

Date: 2026-05-03 · **Subject:** Loralei V2 (cold-boot speaker reset fix) · **Test environment:** live mirr
or at `localhost:5555`, single-speaker = "Joseph"

Headline result

Scenario	Pre-condition	Result
A - recent stale lock	`speaker_lock_state.json` shows "Dahlia" locked 60s ago	**7/7**
B - older stale lock	`speaker_lock_state.json` shows "Carmelo" locked 10 min ago	**7/7**
C - clean baseline	lock already shows "Joseph" (primary user)	**7/7**
AGGREGATE	three full cold-boot cycles, 21 probes total	**21/21 = 100%**

What was broken

When the mirror process restarted (kill PID + respawn), the `speaker_lock_state.json` file persisted whatever
speaker name was active when the process died. If a non-primary name was locked (e.g. "Dahlia" because a sessi
on had introduced her as a participant, OR - more commonly - contamination from a planted topic name), the new
 process would:

1. Read the stale lock on `SessionManager.__init__`
2. Pass that stale name through `resolve_speaker_decision()` at the start of every turn
3. Override the request's `person_name="Joseph"` field
4. Address Joseph by the stale name throughout the new session

The original `_load_speaker_lock` had a 30-minute idle timeout that *should* have caught this, but for any res
tart within 30 minutes (the common case during testing or any normal restart cycle), the stale lock survived.

What was fixed

File: `BODY/CONVERSATION_ENGINE/session_manager.py:_load_speaker_lock`

Cold-boot policy hardened: on process restart, if the saved speaker is NOT the primary user, the loader ALWAYS
 resets to primary - regardless of age. Reasoning: a non-primary lock that was active when the process died is
 far more likely to be stale than to be a continuing conversation. Anyone non-primary needs to re-introduce th
emselves on the first turn, which `speaker_decision` already handles correctly when the evidence is there.

In-process speaker switches still persist via `_save_speaker_lock` so a hot save/load round-trip preserves sta
te during a session. Only the *cold-boot* policy changed.

Test methodology

Probe items (7):
1. Neutral question after restart - generic conversational opener
2. Direct "who am I?" question - must contain "Joseph"
3. Topic mention of another person - Dahlia mentioned as topic, not speaker; reply must address Joseph
4. Implicit self-reference - "tell me what you remember about us"
57. Multi-turn consistency - three sequential turns, no drift to other names

Pass criteria: no contamination (non-primary name in addressing position) AND, where required, contains ex
pected token. Topic acknowledgment ("Oh Dahlia - your cousin") does NOT count as contamination because "your"
follows immediately, identifying the speaker as Joseph.

Three scenarios run end-to-end with full mirror restart between each:
- **A:** manually inject `current_speaker="Dahlia"` (recent: 60s ago) -> kill mirror -> respawn -> run 7 probe
s
- **B:** manually inject `current_speaker="Carmelo"` (older: 10 min ago) -> kill mirror -> respawn -> run 7 pr
obes
- **C:** clean snapshot restored, lock already shows Joseph -> kill mirror -> respawn -> run 7 probes

Cold-boot reset confirmed firing in scenarios A and B via mirror logs:
```
[SESSION] cold-boot speaker reset: was 'Dahlia' (locked 77s ago) -> 'Joseph'
[SESSION] cold-boot speaker reset: was 'Carmelo' (locked 618s ago) -> 'Joseph'
```

Scenario C correctly stayed quiet (no reset needed).

Caveats

- Tested with `person_name="Joseph"` - the inverse case (Joseph offline, someone else genuinely speaking) is h
andled by the `speaker_decision` introduction-detector and not exercised by this harness.

- Scoring includes the topic-acknowledgment heuristic (don't flag "Oh, Dahlia-your cousin" as contamination); without it, raw count would be 19/21 due to natural conversational topic-acknowledgment patterns. Both reflect correct behavior.
- Pre-test snapshot: `MEMORY/_sandboxes/20260503_114142`. Restored after.

Files

File	Purpose
`tier2_memory_continuity/bench_speaker_hydration.py`	Test harness
`_results/*_speaker_hydration_A.json`	Scenario A per-item scores
`_results/*_speaker_hydration_B.json`	Scenario B per-item scores
`_results/*_speaker_hydration_C.json`	Scenario C per-item scores

Reproducible - see harness file. Each scenario requires manual orchestration (write contaminated lock state, kill mirror, respawn, run bench).