P2P Pairing Skills — CC sender + peer responder (MC #102988)

ALAI Company Mesh — P2P Pairing Skills (CC sender + peer responder)

Built: MC #102988 (responder side), 2026-06-05. Extended by MC #102990 (bidirectional), MC #102993 (timeout guard), MC #102996 (autonomous file-mesh loop), and MC #103009 (native-channel decision). Evidence: /tmp/alai/p2p-pairing-evidence/mesh-msg-122e962e-c969-41f1-8f1f-8af6d035e3ca-response.md

> **2026-06-05 decision (MC #103009):** For **in-session orchestrator→worker** work, use native `Agent(run_in_background:true)` + `SendMessage`. It is instant, harness-delivered, auto-wakes the worker, avoids polling/TTL expiry, and avoids CEO relay. The file-mesh skills documented here remain for **cross-machine** or deliberately separate terminal sessions only. Evidence: `/Users/makinja/system/evidence/mc103009-durable-p2p-messaging-decision-20260605.md` and `/tmp/alai/p2p-pairing-evidence/mc103009-sliceB-worker-1.md`.

What this is

These skills let two separate Claude Code sessions pair-program / cross-verify over the ALAI Company Mesh (SQLite-backed message bus at ~/system/databases/company-mesh.db). One session SENDS prompts; the peer session WATCHES and RESPONDS. Use this mesh mode only when native in-session `Agent` / `SendMessage` is not applicable.

Side	Skill	Role
CC agent (this orchestrator)	`p2p-pair` (~/.claude/skills/p2p-pair/SKILL.md)	SENDER — `company-mesh.js send`, await, materialize evidence

Side	Skill	Role
Peer agent (2nd terminal / pi)	`p2p-pair-responder` (~/.claude/skills/p2p-pair-responder/SKILL.md)	RECEIVER — drain inbox, respond, mark processed

Both registered in skill-registry.db at level 3.

Transport (shared, do not reinvent)

- `node ~/system/tools/company-mesh.js send|status|await|respond|list` — message primitives.
- Daemon `com.alai.company-mesh-pi-responder` (company-mesh-pi-responder.js): polls the DB every 10s, writes a trigger file to `/tmp/alai/pi-mesh-inbox/<message_id>.json` when a message targets the peer agent. It NOTIFIES only — it does not execute prompts.
- Helper `~/system/tools/run-p2p-mesh-drain.sh` — single-pass inbox lister (john-bash-block auto-allow pattern).
- Policy `~/system/lib/p2p-pair-policy.js`; context hook `~/.claude/hooks/p2p-pair-context-injector.py`.

How to pair (operator flow)

1. CEO opens a SECOND terminal with a peer Claude Code session.
2. Peer session: invoke `p2p-pair-responder` ("p2p watch" / "enter watch mode"). It drains the inbox on entry, then loops.
3. This (sender) session: invoke `p2p-pair` ("pair with pi" / "mesh send") to send a prompt with an explicit end-state (PASS/PARTIAL/BLOCKED/ANSWERED/DECLINED).
4. Daemon writes the trigger file within ~10s; the watching peer detects it, does the work, responds via `company-mesh.js respond` with evidence, and moves the trigger to `/tmp/alai/pi-mesh-inbox/processed/`.
5. Sender's `await` returns the peer's end-state + evidence_paths.

Hard contract (post-2026-05-31 incident)

- A handshake ANSWERED does NOT mean follow-on prompts will be handled — the peer MUST be in continuous-watch mode. On 2026-05-31, 6 prompts (#102638–643) expired because the peer was not watching.
- Responder drains the WHOLE inbox on entry (mass-drain) and keeps looping until empty; explicit exit "p2p exit watch".

- Do not mass-dispatch from the sender unless the peer is confirmed in watch mode.

Verification (MC #102988 round-trip)

Real mesh round-trip: send mesh-msg-122e962e (thread mesh-thr-f8f00656) → daemon trigger file written → responder steps executed → respond end_state=ANSWERED with evidence → thread status=answered, turn_count=1. Inbox drained (0 unprocessed, 3 processed). NOTE: a genuine two-live-session test requires CEO to open a real peer terminal; all primitives verified against the live mesh DB.

Revision #2
Created 2026-06-05 06:49:27 UTC by John
Updated 2026-06-05 13:52:37 UTC by John