macOS companion app

Peekaboo Bridge

OpenClaw can host **PeekabooBridge** as a local, permission-aware UI automation broker. This lets the `peekaboo` CLI drive UI automation while reusing the macOS app's TCC permissions.

What this is (and isn't)

- **Host**: OpenClaw.app can act as a PeekabooBridge host.
- **Client**: use the `peekaboo` CLI (no separate `openclaw ui ...` surface).
- **UI**: visual overlays stay in Peekaboo.app; OpenClaw is a thin broker host.

Enable the bridge

In the macOS app:

- Settings → **Enable Peekaboo Bridge**

When enabled, OpenClaw starts a local UNIX socket server. If disabled, the host is stopped and `peekaboo` will fall back to other available hosts.

Client discovery order

Peekaboo clients typically try hosts in this order:

1. Peekaboo.app (full UX)
2. Claude.app (if installed)
3. OpenClaw.app (thin broker)

Use `peekaboo bridge status --verbose` to see which host is active and which socket path is in use. You can override with:

```
export PEEKABOO_BRIDGE_SOCKET=/path/to/bridge.sock
```

Security & permissions

- The bridge validates **caller code signatures**; an allowlist of TeamIDs is enforced (Peekaboo host TeamID + OpenClaw app TeamID).
- Requests time out after ~10 seconds.
- If required permissions are missing, the bridge returns a clear error message rather than launching System Settings.

Snapshot behavior (automation)

Snapshots are stored in memory and expire automatically after a short window. If you need longer retention, re-capture from the client.

Troubleshooting

- If `peekaboo` reports “bridge client is not authorized”, ensure the client is properly signed or run the host with `PEEKABOO_ALLOW_UNSIGNED_SOCKET_CLIENTS=1` in **debug** mode only.
- If no hosts are found, open one of the host apps (Peekaboo.app or OpenClaw.app) and confirm permissions are granted.

‹ Skills