



[Platforms overview](#)

## macOS App

The macOS app is the **menu-bar companion** for OpenClaw. It owns permissions, manages/attaches to the Gateway locally (launchd or manual), and exposes macOS capabilities to the agent as a node.

### What it does

Shows native notifications and status in the menu bar.

Owns TCC prompts (Notifications, Accessibility, Screen Recording, Microphone, Speech Recognition, Automation/AppleScript).

Runs or connects to the Gateway (local or remote).

Expose macOS-only tools (Canvas, Camera, Screen Recording, `system.run`).

Starts the local node host service in **remote** mode (launchd), and stops it in **local** mode.

Optionally hosts **PeekabooBridge** for UI automation.

Installs the global CLI (`openclaw`) via npm/pnpm on request (but not recommended for the Gateway runtime).

### Local vs remote mode

**Local** (default): the app attaches to a running local Gateway if present; otherwise it enables the launchd service via `openclaw gateway install`.



**Remote:** the app connects to a Gateway over SSH/Tailscale and never starts a local process. The app starts the local **node host service** so the remote Gateway can reach this Mac. The app does not spawn the Gateway as a 'child process.

## Launchd control

The app manages a per-user LaunchAgent labeled `bot.molt.gateway` (or `bot.molt.<profile>` when using `--profile / OPENCLAW_PROFILE`; legacy `com.openclaw.*` still unloads).

```
launchctl kickstart -k gui/$UID/bot.molt.gateway  
launchctl bootout gui/$UID/bot.molt.gateway
```

Replace the label with `bot.molt.<profile>` when running a named profile.

If the LaunchAgent isn't installed, enable it from the app or run `openclaw gateway install`.

## Node capabilities (mac)

The macOS app presents itself as a node. Common commands:

Canvas: `canvas.present`, `canvas.navigate`, `canvas.eval`,  
`canvas.snapshot`, `canvas.a2ui.*`

Camera: `camera.snap`, `camera.clip`

Screen: `screen.record`

System: `system.run`, `system.notify`

The node reports a `permissions` map so agents can decide what's allowed.

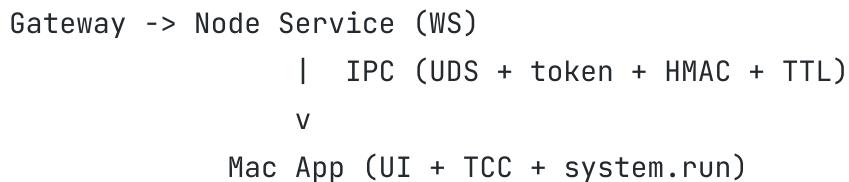
Node service + app IPC:



When the headless node host service is running (remote mode), it connects to the Gateway WS as a node.

`system.run` executes in the macOS app (UI/TCC context) over a local Unix socket; prompts + output stay in-app.

Diagram (SCI):



## Exec approvals (system.run)

`system.run` is controlled by **Exec approvals** in the macOS app (Settings → Exec approvals). Security + ask + allowlist are stored locally on the Mac in:

```
~/.openclaw/exec-approvals.json
```

Example:



```
"version": 1,  
"defaults": {  
    "security": "deny",  
    "ask": "on-miss"  
},  
"agents": {  
    "main": {  
        "security": "allowlist",  
        "ask": "on-miss",  
        "allowlist": [{ "pattern": "/opt/homebrew/bin/rg" }]  
    }  
}
```

## Notes:

`allowlist` entries are glob patterns for resolved binary paths.

Choosing “Always Allow” in the prompt adds that command to the allowlist.

`system.run` environment overrides are filtered (drops `PATH`, `DYLD_*`, `LD_*`, `NODE_OPTIONS`, `PYTHON*`, `PERL*`, `RUBYOPT`) and then merged with the app’s environment.

## Deep links

The app registers the `openclaw://` URL scheme for local actions.

### `openclaw://agent`

Triggers a Gateway agent request.

```
open 'openclaw://agent?message=Hello%20from%20deep%20link'
```

### Query parameters:



message (required)  
sessionKey (optional)  
thinking (optional)  
deliver / to / channel (optional)  
timeoutSeconds (optional)  
key (optional unattended mode key)

### Safety:

Without key , the app prompts for confirmation.

Without key , the app enforces a short message limit for the confirmation prompt and ignores deliver / to / channel .

With a valid key , the run is unattended (intended for personal automations).

## Onboarding flow (typical)

1. Install and launch **OpenClaw.app**.
2. Complete the permissions checklist (TCC prompts).
3. Ensure **Local** mode is active and the Gateway is running.
4. Install the CLI if you want terminal access.

## Build & dev workflow (native)

```
cd apps/macos && swift build  
swift run OpenClaw (or Xcode)  
Package app: scripts/package-mac-app.sh
```

## Debug gateway connectivity (macOS CLI)

Use the debug CLI to exercise the same Gateway WebSocket handshake and discovery logic that the macOS app uses, without launching the app.

```
cd apps/macos  
swift run openclaw-mac connect --json  
swift run openclaw-mac discover --timeout 3000 --json
```

Connect options:

```
--url <ws://host:port> : override config  
--mode <local|remote> : resolve from config (default: config or local)  
--probe : force a fresh health probe  
--timeout <ms> : request timeout (default: 15000 )  
--json : structured output for diffing
```

Discovery options:

```
--include-local : include gateways that would be filtered as "local"  
--timeout <ms> : overall discovery window (default: 2000 )  
--json : structured output for diffing
```

Tip: compare against `openclaw gateway discover --json` to see whether the macOS app's discovery pipeline (NWBrowser + tailnet DNS-SD fallback) differs from the Node CLI's `dns-sd` based discovery.

## Remote connection plumbing (SSH tunnels)

When the macOS app runs in `Remote` mode, it opens an SSH tunnel so local UI components can talk to a remote Gateway as if it were on localhost.

## Control tunnel (Gateway WebSocket port)



**Purpose:** health checks, status, Web Chat, config, and other control-plane calls.

**Local port:** the Gateway port (default 18789), always stable.

**Remote port:** the same Gateway port on the remote host.

**Behavior:** no random local port; the app reuses an existing healthy tunnel or restarts it if needed.

**SSH shape:** ssh -N -L <local>:127.0.0.1:<remote> with BatchMode + ExitOnForwardFailure + keepalive options.

**IP reporting:** the SSH tunnel uses loopback, so the gateway will see the node IP as 127.0.0.1. Use **Direct (ws/wss)** transport if you want the real client IP to appear (see [macOS remote access](#)).

For setup steps, see [macOS remote access](#). For protocol details, see [Gateway protocol](#).

## Related docs

[Gateway runbook](#)

[Gateway \(macOS\)](#)

[macOS permissions](#)

[Canvas](#)

< Platforms

Linux App >

Powered by [mintlify](#)