Platforms overview

Android App

Support snapshot

Role: companion node app (Android does not host the Gateway).

Gateway required: yes (run it on macOS, Linux, or Windows via WSL2).

Install: Getting Started + Pairing.

Gateway: Runbook + Configuration.

Protocols: Gateway protocol (nodes + control plane).

System control

System control (launchd/systemd) lives on the Gateway host. See Gateway.

Connection Runbook

Android node app ⇄ (mDNS/NSD + WebSocket) ⇄ **Gateway**

Android connects directly to the Gateway WebSocket (default `ws://<host>:18789`) and uses Gateway-owned pairing.

Prerequisites

- You can run the Gateway on the “master” machine.
- Android device/emulator can reach the gateway WebSocket:
  - Same LAN with mDNS/NSD, **or**
  - Same Tailscale tailnet using Wide-Area Bonjour / unicast DNS-SD (see below), **or**
  - Manual gateway host/port (fallback)
- You can run the CLI (`openclaw`) on the gateway machine (or via SSH).

1) Start the Gateway

```
openclaw gateway --port 18789 --verbose
```

Confirm in logs you see something like:

- `listening on ws://0.0.0.0:18789`

For tailnet-only setups (recommended for Vienna ⇄ London), bind the gateway to the tailnet IP:

- Set `gateway.bind: "tailnet"` in `~/.openclaw/openclaw.json` on the gateway host.
- Restart the Gateway / macOS menubar app.

2) Verify discovery (optional)

From the gateway machine:

```
dns-sd -B _openclaw-gw._tcp local.
```

More debugging notes: .

Tailnet (Vienna ⇄ London) discovery via unicast DNS-SD

Android NSD/mDNS discovery won't cross networks. If your Android node and the gateway are on different networks but connected via Tailscale, use Wide-Area Bonjour / unicast DNS-SD instead:

1. Set up a DNS-SD zone (example `openclaw.internal.`) on the gateway host and publish `_openclaw-gw._tcp` records.
2. Configure Tailscale split DNS for your chosen domain pointing at that DNS server.

Details and example CoreDNS config: Bonjour.

3) Connect from Android

In the Android app:

- The app keeps its gateway connection alive via a **foreground service** (persistent notification).
- Open **Settings**.
- Under **Discovered Gateways**, select your gateway and hit **Connect**.
- If mDNS is blocked, use **Advanced → Manual Gateway** (host + port) and **Connect (Manual)**.

After the first successful pairing, Android auto-reconnects on launch:

- Manual endpoint (if enabled), otherwise
- The last discovered gateway (best-effort).

4) Approve pairing (CLI)

On the gateway machine:

```
openclaw nodes pending
openclaw nodes approve <requestId>
```

Pairing details: .

5) Verify the node is connected

Via nodes status:

```
openclaw nodes status
```

Via Gateway:

```
openclaw gateway call node.list --params "{}"
```

6) Chat + history

The Android node's Chat sheet uses the gateway's **primary session key** (`main`), so history and replies are shared with WebChat and other clients:

- History: `chat.history`
- Send: `chat.send`
- Push updates (best-effort): `chat.subscribe` → `event:"chat"`

7) Canvas + camera

Gateway Canvas Host (recommended for web content)

If you want the node to show real HTML/CSS/JS that the agent can edit on disk, point the node at the Gateway canvas host.

Note: nodes load canvas from the Gateway HTTP server (same port as `gateway.port` , default `18789`).

1. Create `~/.openclaw/workspace/canvas/index.html` on the gateway host.
2. Navigate the node to it (LAN):

```
openclaw nodes invoke --node "<Android Node>" --command canvas.navigat
```

Tailnet (optional): if both devices are on Tailscale, use a MagicDNS name or tailnet IP instead of `.local` , e.g. `http://<gateway-magicdns>:18789/__openclaw__/canvas/` .

This server injects a live-reload client into HTML and reloads on file changes. The A2UI host lives at `http://<gateway-host>:18789/__openclaw__/a2ui/` .

Canvas commands (foreground only):

- `canvas.eval` , `canvas.snapshot` , `canvas.navigate` (use `{"url":""}` or `{"url":"/"}` to return to the default scaffold). `canvas.snapshot` returns `{ format, base64 }` (default `format="jpeg"`).
- A2UI: `canvas.a2ui.push` , `canvas.a2ui.reset` (`canvas.a2ui.pushJSONL` legacy alias)

Camera commands (foreground only; permission-gated):

- `camera.snap` (jpg)
- `camera.clip` (mp4)

See for parameters and CLI helpers.

‹ Windows (WSL2) iOS App ›