Networking and discovery

Bonjour Discovery

OpenClaw uses Bonjour (mDNS / DNS-SD) as a **LAN-only convenience** to discover an active Gateway (WebSocket endpoint). It is best-effort and does **not** replace SSH or Tailnet-based connectivity.

Wide-area Bonjour (Unicast DNS-SD) over Tailscale

If the node and gateway are on different networks, multicast mDNS won't cross the boundary. You can keep the same discovery UX by switching to **unicast DNS-SD** ("Wide-Area Bonjour") over Tailscale.

High-level steps:

1. Run a DNS server on the gateway host (reachable over Tailnet).
2. Publish DNS-SD records for `_openclaw-gw._tcp` under a dedicated zone (example: `openclaw.internal.`).
3. Configure Tailscale **split DNS** so your chosen domain resolves via that DNS server for clients (including iOS).

OpenClaw supports any discovery domain; `openclaw.internal.` is just an example. iOS/Android nodes browse both `local.` and your configured wide-area domain.

Gateway config (recommended)

```
{
  gateway: { bind: "tailnet" }, // tailnet-only (recommended)
  discovery: { wideArea: { enabled: true } }, // enables wide-area DNS-SD publish
}
```

One-time DNS server setup (gateway host)

```
openclaw dns setup --apply
```

This installs CoreDNS and configures it to:

- listen on port 53 only on the gateway's Tailscale interfaces
- serve your chosen domain (example: `openclaw.internal.`) from `~/.openclaw/dns/<domain>.db`

Validate from a tailnet-connected machine:

```
dns-sd -B _openclaw-gw._tcp openclaw.internal.
dig @<TAILNET_IPV4> -p 53 _openclaw-gw._tcp.openclaw.internal PTR +short
```

Tailscale DNS settings

In the Tailscale admin console:

- Add a nameserver pointing at the gateway's tailnet IP (UDP/TCP 53).
- Add split DNS so your discovery domain uses that nameserver.

Once clients accept tailnet DNS, iOS nodes can browse `_openclaw-gw._tcp` in your discovery domain without multicast.

Gateway listener security (recommended)

The Gateway WS port (default `18789`) binds to loopback by default. For LAN/tailnet access, bind explicitly and keep auth enabled.

For tailnet-only setups: ›

- Set `gateway.bind: "tailnet"` in `~/.openclaw/openclaw.json`.
- Restart the Gateway (or restart the macOS menubar app).

What advertises

Only the Gateway advertises `_openclaw-gw._tcp`.

Service types

- `_openclaw-gw._tcp` — gateway transport beacon (used by macOS/iOS/Android nodes).

TXT keys (non-secret hints)

The Gateway advertises small non-secret hints to make UI flows convenient:

- `role=gateway`
- `displayName=<friendly name>`
- `lanHost=<hostname>.local`
- `gatewayPort=<port>` (Gateway WS + HTTP)
- `gatewayTls=1` (only when TLS is enabled)
- `gatewayTlsSha256=<sha256>` (only when TLS is enabled and fingerprint is available)
- `canvasPort=<port>` (only when the canvas host is enabled; currently the same as `gatewayPort`)
- `sshPort=<port>` (defaults to 22 when not overridden)

- `transport=gateway`
- `cliPath=<path>` (optional; absolute path to a runnable `openclaw` entrypoint)
- `tailnetDns=<magicdns>` (optional hint when Tailnet is available)

Security notes:

- Bonjour/mDNS TXT records are **unauthenticated**. Clients must not treat TXT as authoritative routing.
- Clients should route using the resolved service endpoint (SRV + A/AAAA). Treat `lanHost`, `tailnetDns`, `gatewayPort`, and `gatewayTlsSha256` as hints only.
- TLS pinning must never allow an advertised `gatewayTlsSha256` to override a previously stored pin.
- iOS/Android nodes should treat discovery-based direct connects as **TLS-only** and require explicit user confirmation before trusting a first-time fingerprint.

Debugging on macOS

Useful built-in tools:

- Browse instances:

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

- Resolve one instance (replace `<instance>`):

```
dns-sd -L "<instance>" _openclaw-gw._tcp local.
```

If browsing works but resolving fails, you're usually hitting a LAN policy or mDNS resolver issue.

Debugging in Gateway logs

The Gateway writes a rolling log file (printed on startup as `gateway log file: ...`). Look for `bonjour:` lines, especially:

- `bonjour: advertise failed ...`
- `bonjour: ... name conflict resolved` / `hostname conflict resolved`
- `bonjour: watchdog detected non-announced service ...`

Debugging on iOS node

The iOS node uses `NWBrowser` to discover `_openclaw-gw._tcp` .

To capture logs:

- Settings → Gateway → Advanced → **Discovery Debug Logs**
- Settings → Gateway → Advanced → **Discovery Logs** → reproduce → **Copy**

The log includes browser state transitions and result-set changes.

Common failure modes

- **Bonjour doesn’t cross networks**: use Tailnet or SSH.
- **Multicast blocked**: some Wi-Fi networks disable mDNS.
- **Sleep / interface churn**: macOS may temporarily drop mDNS results; retry.
- **Browse works but resolve fails**: keep machine names simple (avoid emojis or punctuation), then restart the Gateway. The service instance name derives from the host name, so overly complex names can confuse some resolvers.

Escaped instance names (`\032`)

Bonjour/DNS-SD often escapes bytes in service instance names as decimal `\DDD` sequences (e.g. spaces become `\032`).

- This is normal at the protocol level.
- UIs should decode for display (iOS uses `BonjourEscapes.decode`).

Disabling / configuration

- `OPENCLAW_DISABLE_BONJOUR=1` disables advertising (legacy: `OPENCLAW_DISABLE_BONJOUR`).
- `gateway.bind` in `~/.openclaw/openclaw.json` controls the Gateway bind mode.
- `OPENCLAW_SSH_PORT` overrides the SSH port advertised in TXT (legacy: `OPENCLAW_SSH_PORT`).
- `OPENCLAW_TAILNET_DNS` publishes a MagicDNS hint in TXT (legacy: `OPENCLAW_TAILNET_DNS`).
- `OPENCLAW_CLI_PATH` overrides the advertised CLI path (legacy: `OPENCLAW_CLI_PATH`).

Related docs

- Discovery policy and transport selection: **Discovery**
- Node pairing + approvals: **Gateway pairing**

‹ Discovery and Transports Remote Access ›