



≡ Media and devices > **Nodes**

Media and devices

Nodes

A **node** is a companion device (macOS/iOS/Android/headless) that connects to the Gateway **WebSocket** (same port as operators) with `role: "node"` and exposes a command surface (e.g. `canvas.*` , `camera.*` , `system.*`) via `node.invoke` . Protocol details: [Gateway protocol](#).

Legacy transport: [Bridge protocol](#) (TCP JSONL; deprecated/removed for current nodes).

macOS can also run in **node mode**: the menubar app connects to the Gateway's WS server and exposes its local canvas/camera commands as a node (so `openclaw nodes ...` works against this Mac).

Notes:

Nodes are **peripherals**, not gateways. They don't run the gateway service.


Telegram/WhatsApp/etc. messages land on the **gateway**, not on nodes.

Troubleshooting runbook: [/nodes/troubleshooting](#)

Pairing + status

WS nodes use device pairing. Nodes present a device identity during `connect` ; the Gateway creates a device pairing request for `role: node` . Approve via the devices CLI (or UI).

Quick CLI:



```

openclaw devices list
openclaw devices approve <requestId>
openclaw devices reject <requestId>
openclaw nodes status
openclaw nodes describe --node <idOrNameOrIp>

```

Notes:

`nodes status` marks a node as **paired** when its device pairing role includes `node` .

`node.pair.*` (CLI: `openclaw nodes pending/approve/reject`) is a separate gateway-owned node pairing store; it does **not** gate the WS `connect` handshake.

Remote node host (system.run)

Use a **node host** when your Gateway runs on one machine and you want commands to execute on another. The model still talks to the **gateway**; the gateway forwards `exec` calls to the **node host** when `host=node` is selected.

What runs where


Gateway host: receives messages, runs the model, routes tool calls.

Node host: executes `system.run` / `system.which` on the node machine.

Approvals: enforced on the node host via `~/.openclaw/exec-approvals.json` .

Start a node host (foreground)

On the node machine:



```
openclaw node run --host <gateway-host> --port 18789 --display-name "E
```

>

Remote gateway via SSH tunnel (loopback bind)

If the Gateway binds to loopback (`gateway.bind=loopback` , default in local mode), remote node hosts cannot connect directly. Create an SSH tunnel and point the node host at the local end of the tunnel.

Example (node host → gateway host):

```
# Terminal A (keep running): forward local 18790 -> gateway 127.0.0.1:
ssh -N -L 18790:127.0.0.1:18789 user@gateway-host

# Terminal B: export the gateway token and connect through the tunnel
export OPENCLAW_GATEWAY_TOKEN="<gateway-token>"
openclaw node run --host 127.0.0.1 --port 18790 --display-name "Build Node"
```

Notes:

The token is `gateway.auth.token` from the gateway config
(`~/.openclaw/openclaw.json` on the gateway host).

`openclaw node run` reads `OPENCLAW_GATEWAY_TOKEN` for auth.

Start a node host (service)

```
openclaw node install --host <gateway-host> --port 18789 --display-name "Build Node"
openclaw node restart
```

Pair + name

On the gateway host:

```

openclaw nodes pending
openclaw nodes approve <requestId>
openclaw nodes list
>

```

Naming options:

```

--display-name on openclaw node run / openclaw node install (persists
in ~/.openclaw/node.json on the node).

```

```

openclaw nodes rename --node <id|name|ip> --name "Build Node" (gateway
override).

```

Allowlist the commands

Exec approvals are **per node host**. Add allowlist entries from the gateway:

```

openclaw approvals allowlist add --node <id|name|ip> "/usr/bin/uname"
openclaw approvals allowlist add --node <id|name|ip> "/usr/bin/sw_vers"

```

Approvals live on the node host at `~/.openclaw/exec-approvals.json`.

Point exec at the node


Configure defaults (gateway config):

```

openclaw config set tools.exec.host node
openclaw config set tools.exec.security allowlist
openclaw config set tools.exec.node "<id-or-name>"

```

Or per session:

```
 exec host=node security=allowlist node=<id-or-name>
```

Once set, any `exec` call with `host=node` runs on the node host (subject to the node allowlist/approvals).

Related:

Invoking commands

Low-level (raw RPC):

```
openclaw nodes invoke --node <idOrNameOrIp> --command canvas.eval --p
```

Higher-level helpers exist for the common “give the agent a MEDIA attachment” workflows.

Screenshots (canvas snapshots)

If the node is showing the Canvas (WebView), `canvas.snapshot` returns `{ format, base64 }`.

CLI helper (writes to a temp file and prints `MEDIA:<path>`):

```
openclaw nodes canvas snapshot --node <idOrNameOrIp> --format png
openclaw nodes canvas snapshot --node <idOrNameOrIp> --format jpg --max-width 1200
```

Canvas controls



```
openclaw nodes canvas present --node <idOrNameOrIp> --target https://example.com
openclaw nodes canvas hide --node <idOrNameOrIp>
openclaw nodes canvas navigate https://example.com --node <idOrNameOrIp>
openclaw nodes canvas eval --node <idOrNameOrIp> --js "document.title"
```

Notes:

`canvas present` accepts URLs or local file paths (`--target`), plus optional `--x/--y/--width/--height` for positioning.

`canvas eval` accepts inline JS (`--js`) or a positional arg.

A2UI (Canvas)

```
openclaw nodes canvas a2ui push --node <idOrNameOrIp> --text "Hello"
openclaw nodes canvas a2ui push --node <idOrNameOrIp> --jsonl ./payload.jsonl
openclaw nodes canvas a2ui reset --node <idOrNameOrIp>
```

Notes:

Only A2UI v0.8 JSONL is supported (v0.9/createSurface is rejected).

Photos + videos (node camera)

Photos (`jpg`):

```
openclaw nodes camera list --node <idOrNameOrIp>
openclaw nodes camera snap --node <idOrNameOrIp> # default: both facing
openclaw nodes camera snap --node <idOrNameOrIp> --facing front
```

Video clips (mp4):



```
openclaw nodes camera clip --node <idOrNameOrIp> --duration 10s
openclaw nodes camera clip --node <idOrNameOrIp> --duration 3000 --no-audio
```

Notes:

The node must be **foregrounded** for `canvas.*` and `camera.*` (background calls return `NODE_BACKGROUND_UNAVAILABLE`).

Clip duration is clamped (currently `<= 60s`) to avoid oversized base64 payloads.

Android will prompt for `CAMERA / RECORD_AUDIO` permissions when possible; denied permissions fail with `*_PERMISSION_REQUIRED` .

Screen recordings (nodes)

Nodes expose `screen.record` (mp4). Example:

```
openclaw nodes screen record --node <idOrNameOrIp> --duration 10s --fps 10
openclaw nodes screen record --node <idOrNameOrIp> --duration 10s --fps 10 --no-audio
```

Notes:

`screen.record` requires the node app to be foregrounded.

Android will show the system screen-capture prompt before recording.

Screen recordings are clamped to `<= 60s` .

`--no-audio` disables microphone capture (supported on iOS/Android; macOS uses system capture audio).

Use `--screen <index>` to select a display when multiple screens are available.

Location (nodes)



Nodes expose `location.get` when Location is enabled in settings.

>

CLI helper:

```
openclaw nodes location get --node <idOrNameOrIp>
openclaw nodes location get --node <idOrNameOrIp> --accuracy precise --max-age 150
```

Notes:

Location is **off by default**.

“Always” requires system permission; background fetch is best-effort.

The response includes lat/lon, accuracy (meters), and timestamp.

SMS (Android nodes)

Android nodes can expose `sms.send` when the user grants **SMS** permission and the device supports telephony.

Low-level invoke:


```
openclaw nodes invoke --node <idOrNameOrIp> --command sms.send --param
```

Notes:

The permission prompt must be accepted on the Android device before the capability is advertised.

Wi-Fi-only devices without telephony will not advertise `sms.send`.

System commands (node host / mac node)

The macOS node exposes `system.run` , `system.notify` , and  `system.execApprovals.get/set` . The headless node host exposes `system.run` , `system.which` , and `system.execApprovals.get/set` .

Examples:

```
openclaw nodes run --node <idOrNameOrIp> -- echo "Hello from mac node"
openclaw nodes notify --node <idOrNameOrIp> --title "Ping" --body "Gateway ready"
```

Notes:

`system.run` returns stdout/stderr/exit code in the payload.

`system.notify` respects notification permission state on the macOS app.

`system.run` supports `--cwd` , `--env KEY=VAL` , `--command-timeout` , and `--needs-screen-recording` .

`system.notify` supports `--priority <passive|active|timeSensitive>` and `--delivery <system|overlay|auto>` .

Node hosts ignore `PATH` overrides. If you need extra `PATH` entries, configure the node host service environment (or install tools in standard locations) instead of passing `PATH` via `--env` .

On macOS node mode, `system.run` is gated by exec approvals in the macOS app (Settings → Exec approvals). Ask/allowlist/full behave the same as the headless node host; denied prompts return `SYSTEM_RUN_DENIED` .

On headless node host, `system.run` is gated by exec approvals (`~/.openclaw/exec-approvals.json`).

Exec node binding

When multiple nodes are available, you can bind exec to a specific node. This sets the default node for `exec host=node` (and can be

overridden per agent).



Global default:

>

```
openclaw config set tools.exec.node "node-id-or-name"
```

Per-agent override:

```
openclaw config get agents.list  
openclaw config set agents.list[0].tools.exec.node "node-id-or-name"
```

Unset to allow any node:

```
openclaw config unset tools.exec.node  
openclaw config unset agents.list[0].tools.exec.node
```

Permissions map

Nodes may include a `permissions` map in `node.list` / `node.describe`, keyed by permission name (e.g. `screenRecording`, `accessibility`) with boolean values (`true` = granted).

Headless node host (cross-platform)

OpenClaw can run a **headless node host** (no UI) that connects to the Gateway WebSocket and exposes `system.run` / `system.which`. This is useful on Linux/Windows or for running a minimal node alongside a server.

Start it:

```
openclaw node run --host <gateway-host> --port 18789
```

Notes:



Pairing is still required (the Gateway will show a node approval prompt).

The node host stores its node id, token, display name, and gateway connection info in `~/.openclaw/node.json`.

Exec approvals are enforced locally via `~/.openclaw/exec-approvals.json` (see [Exec approvals](#)).

On macOS, the headless node host prefers the companion app exec host when reachable and falls back to local execution if the app is unavailable. Set `OPENCLAW_NODE_EXEC_HOST=app` to require the app, or `OPENCLAW_NODE_EXEC_FALLBACK=0` to disable fallback.

Add `--tls` / `--tls-fingerprint` when the Gateway WS uses TLS.

Mac node mode

The macOS menubar app connects to the Gateway WS server as a node (so `openclaw nodes ...` works against this Mac).

In remote mode, the app opens an SSH tunnel for the Gateway port and connects to `localhost`.

[Auth Monitoring](#)

[Node Troubleshooting](#)

Powered by [mintlify](#)