Configuration and operations

Logging

For a user-facing overview (CLI + Control UI + config), see **/logging**.

OpenClaw has two log “surfaces”:

- **Console output** (what you see in the terminal / Debug UI).
- **File logs** (JSON lines) written by the gateway logger.

File-based logger

- Default rolling log file is under `/tmp/openclaw/` (one file per day): `openclaw-YYYY-MM-DD.log`
  - Date uses the gateway host’s local timezone.
- The log file path and level can be configured via `~/.openclaw/openclaw.json` :
  - `logging.file`
  - `logging.level`

The file format is one JSON object per line.

The Control UI Logs tab tails this file via the gateway (`logs.tail`). CLI can do the same:

```
openclaw logs --follow
```

Verbose vs. log levels

File logs are controlled exclusively by `logging.level`.

`--verbose` only affects **console verbosity** (and WS log style); it does **not** raise the file log level.

To capture verbose-only details in file logs, set `logging.level` to `debug` or `trace`.

Console capture

The CLI captures `console.log/info/warn/error/debug/trace` and writes them to file logs, while still printing to stdout/stderr.

You can tune console verbosity independently via:

`logging.consoleLevel` (default `info`)

`logging.consoleStyle` (`pretty` | `compact` | `json`)

Tool summary redaction

Verbose tool summaries (e.g. `🛠 Exec: ...`) can mask sensitive tokens before they hit the console stream. This is **tools-only** and does not alter file logs.

`logging.redactSensitive`: `off` | `tools` (default: `tools`)

`logging.redactPatterns`: array of regex strings (overrides defaults)

Use raw regex strings (auto `gi`), or `/pattern/flags` if you need custom flags.

Matches are masked by keeping the first 6 + last 4 chars (length ⩾ 18), otherwise `***`.

Defaults cover common key assignments, CLI flags, JSON fields, bearer headers, PEM blocks, and popular token prefixes.

Gateway WebSocket logs

The gateway prints WebSocket protocol logs in two modes:

- **Normal mode (no `--verbose`)**: only “interesting” RPC results are printed:
  - errors (`ok=false`)
  - slow calls (default threshold: `>= 50ms`)
  - parse errors
- **Verbose mode (`--verbose`)**: prints all WS request/response traffic.

WS log style

`openclaw gateway` supports a per-gateway style switch:

- `--ws-log auto` (default): normal mode is optimized; verbose mode uses compact output
- `--ws-log compact` : compact output (paired request/response) when verbose
- `--ws-log full` : full per-frame output when verbose
- `--compact` : alias for `--ws-log compact`

Examples:

```
# optimized (only errors/slow)
openclaw gateway

# show all WS traffic (paired)
openclaw gateway --verbose --ws-log compact

# show all WS traffic (full meta)
openclaw gateway --verbose --ws-log full
```

Console formatting (subsystem logging)

The console formatter is **TTY-aware** and prints consistent, prefixed lines. Subsystem loggers keep output grouped and scannable.

Behavior:

- **Subsystem prefixes** on every line (e.g. `[gateway]`, `[canvas]`, `[tailscale]`)
- **Subsystem colors** (stable per subsystem) plus level coloring
- **Color when output is a TTY or the environment looks like a rich terminal** (`TERM` / `COLORTERM` / `TERM_PROGRAM`), respects `NO_COLOR`
- **Shortened subsystem prefixes**: drops leading `gateway/` + `channels/`, keeps last 2 segments (e.g. `whatsapp/outbound`)
- **Sub-loggers by subsystem** (auto prefix + structured field `{ subsystem }`)
- `logRaw()` for QR/UX output (no prefix, no formatting)
- **Console styles** (e.g. `pretty | compact | json`)
- **Console log level** separate from file log level (file keeps full detail when `logging.level` is set to `debug` / `trace`)
- **WhatsApp message bodies** are logged at `debug` (use `--verbose` to see them)

This keeps existing file logs stable while making interactive output scannable.

‹ Doctor Gateway Lock ›