Configuration and operations

Configuration Reference

Complete field-by-field reference for ~/.openclaw/openclaw.json

Every field available in `~/.openclaw/openclaw.json` . For a task-oriented overview, see Configuration.

Config format is **JSON5** (comments + trailing commas allowed). All fields are optional — OpenClaw uses safe defaults when omitted.

Channels

Each channel starts automatically when its config section exists (unless `enabled: false`).

DM and group access

All channels support DM policies and group policies:

DM policy	Behavior
`pairing` (default)	Unknown senders get a one-time pairing code; owner must approve
`allowlist`	Only senders in `allowFrom` (or paired allow store)
`open`	Allow all inbound DMs (requires `allowFrom: ["*"]`)
`disabled`	Ignore all inbound DMs

Group policy	Behavior
`allowlist` (default)	Only groups matching the configured allowlist
`open`	Bypass group allowlists (mention-gating still applies)
`disabled`	Block all group/room messages

`channels.defaults.groupPolicy` sets the default when a provider's `groupPolicy` is unset. Pairing codes expire after 1 hour. Pending DM pairing requests are capped at **3 per channel**. Slack/Discord have a special fallback: if their provider section is missing entirely, runtime group policy can resolve to `open` (with a startup warning).

WhatsApp

WhatsApp runs through the gateway's web channel (Baileys Web). It starts automatically when a linked session exists.

```
{
  channels: {
    whatsapp: {
      dmPolicy: "pairing", // pairing | allowlist | open | disabled
      allowFrom: ["+15555550123", "+447700900123"],
      textChunkLimit: 4000,
      chunkMode: "length", // length | newline
      mediaMaxMb: 50,
      sendReadReceipts: true, // blue ticks (false in self-chat mode)
      groups: {
        "*": { requireMention: true },
      },
      groupPolicy: "allowlist",
      groupAllowFrom: ["+15551234567"],
    },
  },
  web: {
    enabled: true,
    heartbeatSeconds: 60,
    reconnect: {
      initialMs: 2000,
      maxMs: 120000,
      factor: 1.4,
      jitter: 0.2,
      maxAttempts: 0,
    },
  },
}
```

Multi-account WhatsApp

Telegram

```
  channels: {
    telegram: {
      enabled: true,
      botToken: "your-bot-token",
      dmPolicy: "pairing",
      allowFrom: ["tg:123456789"],
      groups: {
        "*": { requireMention: true },
        "-1001234567890": {
          allowFrom: ["@admin"],
          systemPrompt: "Keep answers brief.",
          topics: {
            "99": {
              requireMention: false,
              skills: ["search"],
              systemPrompt: "Stay on topic.",
            },
          },
        },
      },
      customCommands: [
        { command: "backup", description: "Git backup" },
        { command: "generate", description: "Create an image" },
      ],
      historyLimit: 50,
      replyToMode: "first", // off | first | all
      linkPreview: true,
      streamMode: "partial", // off | partial | block
      draftChunk: {
        minChars: 200,
        maxChars: 800,
        breakPreference: "paragraph", // paragraph | newline | sentence
      },
      actions: { reactions: true, sendMessage: true },
      reactionNotifications: "own", // off | own | all
      mediaMaxMb: 5,
      retry: {
        attempts: 3,
        minDelayMs: 400,
        maxDelayMs: 30000,
        jitter: 0.1,
      },
      network: { autoSelectFamily: false },
      proxy: "socks5://localhost:9050",
      webhookUrl: "https://example.com/telegram-webhook",
      webhookSecret: "secret",
      webhookPath: "/telegram-webhook",
    },
  },
}
```

Bot token: `channels.telegram.botToken` or `channels.telegram.tokenFile` , with `TELEGRAM_BOT_TOKEN` as fallback for the default account.

`configWrites: false` blocks Telegram-initiated config writes (supergroup ID migrations, `/config set|unset`).

Telegram stream previews use `sendMessage` + `editMessageText` (works in direct and group chats).

Retry policy: see **Retry policy**.

Discord

```
{
  channels: {
    discord: {
      enabled: true,
      token: "your-bot-token",
      mediaMaxMb: 8,
      allowBots: false,
      actions: {
        reactions: true,
        stickers: true,
        polls: true,
        permissions: true,
        messages: true,
        threads: true,
        pins: true,
        search: true,
        memberInfo: true,
        roleInfo: true,
        roles: false,
        channelInfo: true,
        voiceStatus: true,
        events: true,
        moderation: false,
      },
      replyToMode: "off", // off | first | all
      dmPolicy: "pairing",
      allowFrom: ["1234567890", "steipete"],
      dm: { enabled: true, groupEnabled: false, groupChannels: ["openclaw-dm"] },
      guilds: {
        "123456789012345678": {
          slug: "friends-of-openclaw",
          requireMention: false,
          reactionNotifications: "own",
          users: ["987654321098765432"],
          channels: {
            general: { allow: true },
            help: {
              allow: true,
              requireMention: true,
              users: ["987654321098765432"],
              skills: ["docs"],
              systemPrompt: "Short answers only.",
            },
          },
        },
      },
      historyLimit: 20,
      textChunkLimit: 2000,
      chunkMode: "length", // length | newline
      maxLinesPerMessage: 17,
      ui: {
        components: {
          accentColor: "#5865F2",
        },
      },
      retry: {
        attempts: 3,
        minDelayMs: 500,
        maxDelayMs: 30000,
        jitter: 0.1,
      },
    },
  },
}
```

Token: `channels.discord.token` , with `DISCORD_BOT_TOKEN` as fallback for the default account.

Use `user:<id>` (DM) or `channel:<id>` (guild channel) for delivery targets; bare numeric IDs are rejected.

Guild slugs are lowercase with spaces replaced by `-` ; channel keys use the slugged name (no `#`). Prefer guild IDs.

Bot-authored messages are ignored by default. `allowBots: true` enables them (own messages still filtered).

`maxLinesPerMessage` (default 17) splits tall messages even when under 2000 chars.

`channels.discord.ui.components.accentColor` sets the accent color for Discord components v2 containers.

Reaction notification modes: › `off` (none), `own` (bot's messages, default), `all` (all messages), `allowlist` (from `guilds.<id>.users` on all messages).

Google Chat

```
{
  channels: {
    googlechat: {
      enabled: true,
      serviceAccountFile: "/path/to/service-account.json",
      audienceType: "app-url", // app-url | project-number
      audience: "https://gateway.example.com/googlechat",
      webhookPath: "/googlechat",
      botUser: "users/1234567890",
      dm: {
        enabled: true,
        policy: "pairing",
        allowFrom: ["users/1234567890"],
      },
      groupPolicy: "allowlist",
      groups: {
        "spaces/AAAA": { allow: true, requireMention: true },
      },
      actions: { reactions: true },
      typingIndicator: "message",
      mediaMaxMb: 20,
    },
  },
}
```

Service account JSON: inline (`serviceAccount`) or file-based (`serviceAccountFile`).

Env fallbacks: `GOOGLE_CHAT_SERVICE_ACCOUNT` or `GOOGLE_CHAT_SERVICE_ACCOUNT_FILE` .

Use `spaces/<spaceId>` or `users/<userId|email>` for delivery targets.

Slack

```
{
  channels: {
    slack: {
      enabled: true,
      botToken: "xoxb-...",
      appToken: "xapp-...",
      dmPolicy: "pairing",
      allowFrom: ["U123", "U456", "*"],
      dm: { enabled: true, groupEnabled: false, groupChannels: ["G123"] },
      channels: {
        C123: { allow: true, requireMention: true, allowBots: false },
        "#general": {
          allow: true,
          requireMention: true,
          allowBots: false,
          users: ["U123"],
          skills: ["docs"],
          systemPrompt: "Short answers only.",
        },
      },
      historyLimit: 50,
      allowBots: false,
      reactionNotifications: "own",
      reactionAllowlist: ["U123"],
      replyToMode: "off", // off | first | all
      thread: {
        historyScope: "thread", // thread | channel
        inheritParent: false,
      },
      actions: {
        reactions: true,
        messages: true,
        pins: true,
        memberInfo: true,
        emojiList: true,
      },
      slashCommand: {
        enabled: true,
        name: "openclaw",
        sessionPrefix: "slack:slash",
        ephemeral: true,
      },
      textChunkLimit: 4000,
      chunkMode: "length",
      mediaMaxMb: 20,
    },
  },
}
```

- **Socket mode** requires both `botToken` and `appToken` (`SLACK_BOT_TOKEN` + `SLACK_APP_TOKEN` for default account env fallback).
- **HTTP mode** requires `botToken` plus `signingSecret` (at root or per-account).
- `configWrites: false` blocks Slack-initiated config writes.
- Use `user:<id>` (DM) or `channel:<id>` for delivery targets.

Reaction notification modes: `off` , `own` (default), `all` , `allowlist` (from `reactionAllowlist`).

Thread session isolation: `thread.historyScope` is per-thread (default) or shared across channel. `thread.inheritParent` copies parent channel transcript to new threads.

Action group	Default	Notes
reactions	enabled	React + list reactions
messages	enabled	Read/send/edit/delete
pins	enabled	Pin/unpin/list
memberInfo	enabled	Member info
emojiList	enabled	Custom emoji list

Mattermost

Mattermost ships as a plugin: `openclaw plugins install @openclaw/mattermost` .

```
{
  channels: {
    mattermost: {
      enabled: true,
      botToken: "mm-token",
      baseUrl: "https://chat.example.com",
      dmPolicy: "pairing",
      chatmode: "oncall", // oncall | onmessage | onchar
      oncharPrefixes: [">", "!"],
      textChunkLimit: 4000,
      chunkMode: "length",
    },
  },
}
```

Chat modes: `oncall` (respond on @-mention, default), `onmessage` (every message), `onchar` (messages starting with trigger prefix).

Signal

```
{
  channels: {
    signal: {
      reactionNotifications: "own", // off | own | all | allowlist
      reactionAllowlist: ["+15551234567", "uuid:123e4567-e89b-12d3-a456-42661417
      historyLimit: 50,
    },
  },
}
```

Reaction notification modes: `off` , `own` (default), `all` , `allowlist` (from `reactionAllowlist`).

iMessage

OpenClaw spawns `imsg rpc` (JSON-RPC over stdio). No daemon or port required.

```
{
  channels: {
    imessage: {
      enabled: true,
      cliPath: "imsg",
      dbPath: "~/Library/Messages/chat.db",
      remoteHost: "user@gateway-host",
      dmPolicy: "pairing",
      allowFrom: ["+15555550123", "user@example.com", "chat_id:123"],
      historyLimit: 50,
      includeAttachments: false,
      mediaMaxMb: 16,
      service: "auto",
      region: "US",
    },
  },
}
```

- Requires Full Disk Access to the Messages DB.
- Prefer `chat_id:<id>` targets. Use `imsg chats --limit 20` to list chats.
- `cliPath` can point to an SSH wrapper; set `remoteHost` for SCP attachment fetching.

iMessage SSH wrapper example

Multi-account (all channels)

Run multiple accounts per channel (each with its own `accountId`):

```
{
  channels: {
    telegram: {
      accounts: {
        default: {
          name: "Primary bot",
          botToken: "123456:ABC...",
        },
        alerts: {
          name: "Alerts bot",
          botToken: "987654:XYZ...",
        },
      },
    },
  },
}
```

- `default` is used when `accountId` is omitted (CLI + routing).
- Env tokens only apply to the **default** account.
- Base channel settings apply to all accounts unless overridden per account.
- Use `bindings[].match.accountId` to route each account to a different agent.

Group chat mention gating

Group messages default to **require mention** (metadata mention or regex patterns). Applies to WhatsApp, Telegram, Discord, Google Chat, and iMessage group chats.

Mention types:

- **Metadata mentions**: Native platform @-mentions. Ignored in WhatsApp self-chat mode.

Text patterns: Regex patterns in `agents.list[].groupChat.mentionPatterns` . Always checked.

Mention gating is enforced only when detection is possible (native mentions or at least one pattern).

```
{
  messages: {
    groupChat: { historyLimit: 50 },
  },
  agents: {
    list: [{ id: "main", groupChat: { mentionPatterns: ["@openclaw", "openclaw"]
  },
}
```

`messages.groupChat.historyLimit` sets the global default. Channels can override with `channels.<channel>.historyLimit` (or per-account). Set `0` to disable.

DM history limits

```
{
  channels: {
    telegram: {
      dmHistoryLimit: 30,
      dms: {
        "123456789": { historyLimit: 50 },
      },
    },
  },
}
```

Resolution: per-DM override → provider default → no limit (all retained).

Supported: `telegram` , `whatsapp` , `discord` , `slack` , `signal` , `imessage` , `msteams` .

Self-chat mode

Include your own number in `allowFrom` to enable self-chat mode (ignores native @-mentions, only responds to text patterns):

```
{
  channels: {
    whatsapp: {
      allowFrom: ["+15555550123"],
      groups: { "*": { requireMention: true } },
    },
  },
  agents: {
    list: [
      {
        id: "main",
        groupChat: { mentionPatterns: ["reisponde", "@openclaw"] },
      },
    ],
  },
}
```

Commands (chat command handling)

```
{
  commands: {
    native: "auto", // register native commands when supported
    text: true, // parse /commands in chat messages
    bash: false, // allow ! (alias: /bash)
    bashForegroundMs: 2000,
    config: false, // allow /config
    debug: false, // allow /debug
    restart: false, // allow /restart + gateway restart tool
    allowFrom: {
      "*": ["user1"],
      discord: ["user:123"],
    },
    useAccessGroups: true,
  },
}
```

Command details

Agent defaults

agents.defaults.workspace

Default: ~/.openclaw/workspace .

```
{
  agents: { defaults: { workspace: "~/.openclaw/workspace" } },
}
```

agents.defaults.repoRoot

Optional repository root shown in the system prompt's Runtime line. If unset, OpenClaw auto-detects by walking upward from the workspace.

```
{
  agents: { defaults: { repoRoot: "~/Projects/openclaw" } },
}
```

`agents.defaults.skipBootstrap`

Disables automatic creation of workspace bootstrap files (`AGENTS.md` , `SOUL.md` , `TOOLS.md` , `IDENTITY.md` , `USER.md` , `HEARTBEAT.md` , `BOOTSTRAP.md`).

```
{
  agents: { defaults: { skipBootstrap: true } },
}
```

`agents.defaults.bootstrapMaxChars`

Max characters per workspace bootstrap file before truncation. Default: `20000` .

```
{
  agents: { defaults: { bootstrapMaxChars: 20000 } },
}
```

`agents.defaults.bootstrapTotalMaxChars`

Max total characters injected across all workspace bootstrap files. Default: `150000` .

```
{
  agents: { defaults: { bootstrapTotalMaxChars: 150000 } },
}
```

`agents.defaults.imageMaxDimensionPx`

Max pixel size for the longest image side in transcript/tool image blocks before provider calls. Default: `1200`.

Lower values usually reduce vision-token usage and request payload size for screenshot-heavy runs. Higher values preserve more visual detail.

```
{
  agents: { defaults: { imageMaxDimensionPx: 1200 } },
}
```

`agents.defaults.userTimezone`

Timezone for system prompt context (not message timestamps). Falls back to host timezone.

```
{
  agents: { defaults: { userTimezone: "America/Chicago" } },
}
```

`agents.defaults.timeFormat`

Time format in system prompt. Default: `auto` (OS preference).

```
{
  agents: { defaults: { timeFormat: "auto" } }, // auto | 12 | 24
}
```

`agents.defaults.model`

```
{
  agents: {
    defaults: {
      models: {
        "anthropic/claude-opus-4-6": { alias: "opus" },
        "minimax/MiniMax-M2.1": { alias: "minimax" },
      },
      model: {
        primary: "anthropic/claude-opus-4-6",
        fallbacks: ["minimax/MiniMax-M2.1"],
      },
      imageModel: {
        primary: "openrouter/qwen/qwen-2.5-vl-72b-instruct:free",
        fallbacks: ["openrouter/google/gemini-2.0-flash-vision:free"],
      },
      thinkingDefault: "low",
      verboseDefault: "off",
      elevatedDefault: "on",
      timeoutSeconds: 600,
      mediaMaxMb: 5,
      contextTokens: 200000,
      maxConcurrent: 3,
    },
  },
}
```

`model.primary` : format `provider/model` (e.g. `anthropic/claude-opus-4-6`). If you omit the provider, OpenClaw assumes `anthropic` (deprecated).

`models` : the configured model catalog and allowlist for `/model` . Each entry can include `alias` (shortcut) and `params` (provider-specific: `temperature` , `maxTokens`).

`imageModel` : only used if the primary model lacks image input.

`maxConcurrent` : max parallel agent runs across sessions (each session still serialized). Default: 1.

Built-in alias shorthands (only apply when the model is in `agents.defaults.models`):

Alias	Model
`opus`	`anthropic/claude-opus-4-6`
`sonnet`	`anthropic/claude-sonnet-4-5`
`gpt`	`openai/gpt-5.2`
`gpt-mini`	`openai/gpt-5-mini`
`gemini`	`google/gemini-3-pro-preview`
`gemini-flash`	`google/gemini-3-flash-preview`

Your configured aliases always win over defaults.

Z.AI GLM-4.x models automatically enable thinking mode unless you set `--thinking off` or define `agents.defaults.models["zai/<model>"].params.thinking` yourself. Z.AI models enable `tool_stream` by default for tool call streaming. Set `agents.defaults.models["zai/<model>"].params.tool_stream` to `false` to disable it.

`agents.defaults.cliBackends`

Optional CLI backends for text-only fallback runs (no tool calls). Useful as a backup when API providers fail.

```
{
  agents: {
    defaults: {
      cliBackends: {
        "claude-cli": {
          command: "/opt/homebrew/bin/claude",
        },
        "my-cli": {
          command: "my-cli",
          args: ["--json"],
          output: "json",
          modelArg: "--model",
          sessionArg: "--session",
          sessionMode: "existing",
          systemPromptArg: "--system",
          systemPromptWhen: "first",
          imageArg: "--image",
          imageMode: "repeat",
        },
      },
    },
  },
}
```

CLI backends are text-first; tools are always disabled.

Sessions supported when `sessionArg` is set.

Image pass-through supported when `imageArg` accepts file paths.

agents.defaults.heartbeat

Periodic heartbeat runs.

```
{
  agents: {
    defaults: {
      heartbeat: {
        every: "30m", // 0m disables
        model: "openai/gpt-5.2-mini",
        includeReasoning: false,
        session: "main",
        to: "+15555550123",
        target: "last", // last | whatsapp | telegram | discord | ... | none
        prompt: "Read HEARTBEAT.md if it exists...",
        ackMaxChars: 300,
        suppressToolErrorWarnings: false,
      },
    },
  },
}
```

`every`: duration string (ms/s/m/h). Default: `30m`.

`suppressToolErrorWarnings`: when true, suppresses tool error warning payloads during heartbeat runs.

Per-agent: set `agents.list[].heartbeat`. When any agent defines `heartbeat`, **only those agents** run heartbeats.

Heartbeats run full agent turns — shorter intervals burn more tokens.

agents.defaults.compaction

```
{
  agents: {
    defaults: {
      compaction: {
        mode: "safeguard", // default | safeguard
        reserveTokensFloor: 24000,
        memoryFlush: {
          enabled: true,
          softThresholdTokens: 6000,
          systemPrompt: "Session nearing compaction. Store durable memories now."
          prompt: "Write any lasting notes to memory/YYYY-MM-DD.md; reply with NO
        },
      },
    },
  },
}
```

- `mode` : `default` or `safeguard` (chunked summarization for long histories). See .
- `memoryFlush` : silent agentic turn before auto-compaction to store durable memories. Skipped when workspace is read-only.

agents.defaults.contextPruning

Prunes **old tool results** from in-memory context before sending to the LLM. Does **not** modify session history on disk.

```
{
  agents: {
    defaults: {
      contextPruning: {
        mode: "cache-ttl", // off | cache-ttl
        ttl: "1h", // duration (ms/s/m/h), default unit: minutes
        keepLastAssistants: 3,
        softTrimRatio: 0.3,
        hardClearRatio: 0.5,
        minPrunableToolChars: 50000,
        softTrim: { maxChars: 4000, headChars: 1500, tailChars: 1500 },
        hardClear: { enabled: true, placeholder: "[Old tool result content cleare
        tools: { deny: ["browser", "canvas"] },
      },
    },
  },
}
```

cache-ttl mode behavior

See for behavior details.

Block streaming

```
{
  agents: {
    defaults: {
      blockStreamingDefault: "off", // on | off
      blockStreamingBreak: "text_end", // text_end | message_end
      blockStreamingChunk: { minChars: 800, maxChars: 1200 },
      blockStreamingCoalesce: { idleMs: 1000 },
      humanDelay: { mode: "natural" }, // off | natural | custom (use minMs/maxMs
    },
  },
}
```

Non-Telegram channels require explicit `*.blockStreaming: true` to enable block replies.

Channel overrides: `channels.<channel>.blockStreamingCoalesce` (and per-account variants). Signal/Slack/Discord/Google Chat default `minChars: 1500`.

`humanDelay`: randomized pause between block replies. `natural` = 800–2500ms. Per-agent override: `agents.list[].humanDelay`.

See **Streaming** for behavior + chunking details.

Typing indicators

```
{
  agents: {
    defaults: {
      typingMode: "instant", // never | instant | thinking | message
      typingIntervalSeconds: 6,
    },
  },
}
```

Defaults: `instant` for direct chats/mentions, `message` for unmentioned group chats.

Per-session overrides: `session.typingMode`, `session.typingIntervalSeconds`.

See .

`agents.defaults.sandbox`

Optional **Docker sandboxing** for the embedded agent. See for the full guide.

```
{
  agents: {
    defaults: {
      sandbox: {
        mode: "non-main", // off | non-main | all
        scope: "agent", // session | agent | shared
        workspaceAccess: "none", // none | ro | rw
        workspaceRoot: "~/.openclaw/sandboxes",
        docker: {
          image: "openclaw-sandbox:bookworm-slim",
          containerPrefix: "openclaw-sbx-",
          workdir: "/workspace",
          readOnlyRoot: true,
          tmpfs: ["/tmp", "/var/tmp", "/run"],
          network: "none",
          user: "1000:1000",
          capDrop: ["ALL"],
          env: { LANG: "C.UTF-8" },
          setupCommand: "apt-get update && apt-get install -y git curl jq",
          pidsLimit: 256,
          memory: "1g",
          memorySwap: "2g",
          cpus: 1,
          ulimits: {
            nofile: { soft: 1024, hard: 2048 },
            nproc: 256,
          },
          seccompProfile: "/path/to/seccomp.json",
          apparmorProfile: "openclaw-sandbox",
          dns: ["1.1.1.1", "8.8.8.8"],
          extraHosts: ["internal.service:10.0.0.5"],
          binds: ["/home/user/source:/source:rw"],
        },
        browser: {
          enabled: false,
          image: "openclaw-sandbox-browser:bookworm-slim",
          cdpPort: 9222,
          vncPort: 5900,
          noVncPort: 6080,
          headless: false,
          enableNoVnc: true,
          allowHostControl: false,
          autoStart: true,
          autoStartTimeoutMs: 12000,
        },
        prune: {
          idleHours: 24,
          maxAgeDays: 7,
        },
      },
    },
  },
  tools: {
    sandbox: {
      tools: {
        allow: [
          "exec",
          "process",
          "read",
          "write",
          "edit",
          "apply_patch",
          "sessions_list",
          "sessions_history",
          "sessions_send",
          "sessions_spawn",
          "session_status",
        ],
        deny: ["browser", "canvas", "nodes", "cron", "discord", "gateway"],
      },
    },
  },
}
```

Sandbox details

Build images:

```
scripts/sandbox-setup.sh           # main sandbox image
scripts/sandbox-browser-setup.sh   # optional browser image
```

`agents.list` (per-agent overrides)

```
{
  agents: {
    list: [
      {
        id: "main",
        default: true,
        name: "Main Agent",
        workspace: "~/.openclaw/workspace",
        agentDir: "~/.openclaw/agents/main/agent",
        model: "anthropic/claude-opus-4-6", // or { primary, fallbacks }
        identity: {
          name: "Samantha",
          theme: "helpful sloth",
          emoji: "🦥",
          avatar: "avatars/samantha.png",
        },
        groupChat: { mentionPatterns: ["@openclaw"] },
        sandbox: { mode: "off" },
        subagents: { allowAgents: ["*"] },
        tools: {
          profile: "coding",
          allow: ["browser"],
          deny: ["canvas"],
          elevated: { enabled: true },
        },
      },
    ],
  },
}
```

- **`id`**: stable agent id (required).

- `default`: when multiple are set, first wins (warning logged). If none set, first list entry is default.
- `model`: string form overrides `primary` only; object form `{ primary, fallbacks }` overrides both (`[]` disables global fallbacks). Cron jobs that only override `primary` still inherit default fallbacks unless you set `fallbacks: []`.
- `identity.avatar`: workspace-relative path, `http(s)` URL, or `data:` URI.
- `identity` derives defaults: `ackReaction` from `emoji`, `mentionPatterns` from `name` / `emoji`.
- `subagents.allowAgents`: allowlist of agent ids for `sessions_spawn` (`["*"]` = any; default: same agent only).

Multi-agent routing

Run multiple isolated agents inside one Gateway. See **Multi-Agent**.

```
{
  agents: {
    list: [
      { id: "home", default: true, workspace: "~/.openclaw/workspace-home" },
      { id: "work", workspace: "~/.openclaw/workspace-work" },
    ],
  },
  bindings: [
    { agentId: "home", match: { channel: "whatsapp", accountId: "personal" } },
    { agentId: "work", match: { channel: "whatsapp", accountId: "biz" } },
  ],
}
```

Binding match fields

- `match.channel` (required)
- `match.accountId` (optional; `*` = any account; omitted = default account)
- `match.peer` (optional; `{ kind: direct|group|channel, id }`)
- `match.guildId` / `match.teamId` (optional; channel-specific)

Deterministic match order:

1. `match.peer`
2. `match.guildId`
3. `match.teamId`
4. `match.accountId` (exact, no peer/guild/team)
5. `match.accountId: "*"` (channel-wide)
6. Default agent

Within each tier, the first matching `bindings` entry wins.

Per-agent access profiles

Full access (no sandbox)

Read-only tools + workspace

No filesystem access (messaging only)

See **Multi-Agent Sandbox & Tools** for precedence details.

Session

```
{
  session: {
    scope: "per-sender",
    dmScope: "main", // main | per-peer | per-channel-peer | per-account-channel-
    identityLinks: {
      alice: ["telegram:123456789", "discord:987654321012345678"],
    },
    reset: {
      mode: "daily", // daily | idle
      atHour: 4,
      idleMinutes: 60,
    },
    resetByType: {
      thread: { mode: "daily", atHour: 4 },
      direct: { mode: "idle", idleMinutes: 240 },
      group: { mode: "idle", idleMinutes: 120 },
    },
    resetTriggers: ["/new", "/reset"],
    store: "~/.openclaw/agents/{agentId}/sessions/sessions.json",
    maintenance: {
      mode: "warn", // warn | enforce
      pruneAfter: "30d",
      maxEntries: 500,
      rotateBytes: "10mb",
    },
    mainKey: "main", // legacy (runtime always uses "main")
    agentToAgent: { maxPingPongTurns: 5 },
    sendPolicy: {
      rules: [{ action: "deny", match: { channel: "discord", chatType: "group" } 
      default: "allow",
    },
  },
}
```

Session field details

Messages

```
{
  messages: {
    responsePrefix: "🦞", // or "auto"
    ackReaction: "👀",
    ackReactionScope: "group-mentions", // group-mentions | group-all | direct |
    removeAckAfterReply: false,
    queue: {
      mode: "collect", // steer | followup | collect | steer-backlog | steer+back
      debounceMs: 1000,
      cap: 20,
      drop: "summarize", // old | new | summarize
      byChannel: {
        whatsapp: "collect",
        telegram: "collect",
      },
    },
    inbound: {
      debounceMs: 2000, // 0 disables
      byChannel: {
        whatsapp: 5000,
        slack: 1500,
      },
    },
  },
}
```

Response prefix

Per-channel/account overrides: `channels.<channel>.responsePrefix` , `channels.<channel>.accounts.<id>.responsePrefix` .

Resolution (most specific wins): account → channel → global. `""` disables and stops cascade. `"auto"` derives `[{identity.name}]` .

Template variables:

Variable	Description	Example
`{model}`	Short model name	`claude-opus-4-6`
`{modelFull}`	Full model identifier	`anthropic/claude-opus-4-6`
`{provider}`	Provider name	`anthropic`
`{thinkingLevel}`	Current thinking level	`high` , `low` , `off`
`{identity.name}`	Agent identity name	(same as `"auto"`)

Variables are case-insensitive. `{think}` is an alias for `{thinkingLevel}` .

Ack reaction

- Defaults to active agent's `identity.emoji` , otherwise `"👀"` . Set `""` to disable.
- Per-channel overrides: `channels.<channel>.ackReaction` , `channels.<channel>.accounts.<id>.ackReaction` .
- Resolution order: account → channel → `messages.ackReaction` → identity fallback.
- Scope: `group-mentions` (default), `group-all` , `direct` , `all` .
- `removeAckAfterReply` : removes ack after reply (Slack/Discord/Telegram/Google Chat only).

Inbound debounce

Batches rapid text-only messages from the same sender into a single agent turn. Media/attachments flush immediately. Control commands bypass debouncing.

TTS (text-to-speech)

```
{
  messages: {
    tts: {
      auto: "always", // off | always | inbound | tagged
      mode: "final", // final | all
      provider: "elevenlabs",
      summaryModel: "openai/gpt-4.1-mini",
      modelOverrides: { enabled: true },
      maxTextLength: 4000,
      timeoutMs: 30000,
      prefsPath: "~/.openclaw/settings/tts.json",
      elevenlabs: {
        apiKey: "elevenlabs_api_key",
        baseUrl: "https://api.elevenlabs.io",
        voiceId: "voice_id",
        modelId: "eleven_multilingual_v2",
        seed: 42,
        applyTextNormalization: "auto",
        languageCode: "en",
        voiceSettings: {
          stability: 0.5,
          similarityBoost: 0.75,
          style: 0.0,
          useSpeakerBoost: true,
          speed: 1.0,
        },
      },
      openai: {
        apiKey: "openai_api_key",
        model: "gpt-4o-mini-tts",
        voice: "alloy",
      },
    },
  },
}
```

`auto` controls auto-TTS. `/tts off|always|inbound|tagged` overrides per session.

- `summaryModel` overrides `agents.defaults.model.primary` for auto-summary.
- API keys fall back to `ELEVENLABS_API_KEY` / `XI_API_KEY` and `OPENAI_API_KEY` .

Talk

Defaults for Talk mode (macOS/iOS/Android).

```
{
  talk: {
    voiceId: "elevenlabs_voice_id",
    voiceAliases: {
      Clawd: "EXAVITQu4vr4xnSDxMaL",
      Roger: "CwhRBWXzGAHq8TQ4Fs17",
    },
    modelId: "eleven_v3",
    outputFormat: "mp3_44100_128",
    apiKey: "elevenlabs_api_key",
    interruptOnSpeech: true,
  },
}
```

- Voice IDs fall back to `ELEVENLABS_VOICE_ID` or `SAG_VOICE_ID` .
- `apiKey` falls back to `ELEVENLABS_API_KEY` .
- `voiceAliases` lets Talk directives use friendly names.

Tools

Tool profiles

`tools.profile` sets a base allowlist before `tools.allow` / `tools.deny` :

Profile	Includes
minimal	session_status only
coding	group:fs , group:runtime , group:sessions , group:memory , image
messaging	group:messaging , sessions_list , sessions_history , sessions_send , session_status
full	No restriction (same as unset)

Tool groups

Group	Tools
group:runtime	exec , process (bash is accepted as an alias for exec)
group:fs	read , write , edit , apply_patch
group:sessions	sessions_list , sessions_history , sessions_send , sessions_spawn , session_status
group:memory	memory_search , memory_get
group:web	web_search , web_fetch
group:ui	browser , canvas
group:automation	cron , gateway
group:messaging	message
group:nodes	nodes
group:openclaw	All built-in tools (excludes provider plugins)

tools.allow / tools.deny

Global tool allow/deny policy (deny wins). Case-insensitive, supports * wildcards. Applied even when Docker sandbox is off.

```
{
  tools: { deny: ["browser", "canvas"] },
}
```

tools.byProvider

Further restrict tools for specific providers or models. Order: base profile → provider profile → allow/deny.

```
{
  tools: {
    profile: "coding",
    byProvider: {
      "google-antigravity": { profile: "minimal" },
      "openai/gpt-5.2": { allow: ["group:fs", "sessions_list"] },
    },
  },
}
```

tools.elevated

Controls elevated (host) exec access:

```
{
  tools: {
    elevated: {
      enabled: true,
      allowFrom: {
        whatsapp: ["+15555550123"],
        discord: ["steipete", "1234567890123"],
      },
    },
  },
}
```

Per-agent override (`agents.list[].tools.elevated`) can only further restrict.

`/elevated on|off|ask|full` stores state per session; inline directives apply to single message.

Elevated `exec` runs on the host, bypasses sandboxing.

tools.exec

```
{
  tools: {
    exec: {
      backgroundMs: 10000,
      timeoutSec: 1800,
      cleanupMs: 1800000,
      notifyOnExit: true,
      notifyOnExitEmptySuccess: false,
      applyPatch: {
        enabled: false,
        allowModels: ["gpt-5.2"],
      },
    },
  },
}
```

tools.loopDetection

Tool-loop safety checks are **disabled by default**. Set `enabled: true` to activate detection. Settings can be defined globally in `tools.loopDetection` and overridden per-agent at `agents.list[].tools.loopDetection` .

```
{
  tools: {
    loopDetection: {
      enabled: true,
      historySize: 30,
      warningThreshold: 10,
      criticalThreshold: 20,
      globalCircuitBreakerThreshold: 30,
      detectors: {
        genericRepeat: true,
        knownPollNoProgress: true,
        pingPong: true,
      },
    },
  },
}
```

`historySize` : max tool-call history retained for loop analysis.

`warningThreshold` : repeating no-progress pattern threshold for warnings.

`criticalThreshold` : higher repeating threshold for blocking critical loops.

`globalCircuitBreakerThreshold` : hard stop threshold for any no-progress run.

`detectors.genericRepeat` : warn on repeated same-tool/same-args calls.

`detectors.knownPollNoProgress` : warn/block on known poll tools (`process.poll` , `command_status` , etc.).

`detectors.pingPong` : warn/block on alternating no-progress pair patterns.

If `warningThreshold >= criticalThreshold` or `criticalThreshold >= globalCircuitBreakerThreshold` , validation fails.

`tools.web`

```
{
  tools: {
    web: {
      search: {
        enabled: true,
        apiKey: "brave_api_key", // or BRAVE_API_KEY env
        maxResults: 5,
        timeoutSeconds: 30,
        cacheTtlMinutes: 15,
      },
      fetch: {
        enabled: true,
        maxChars: 50000,
        maxCharsCap: 50000,
        timeoutSeconds: 30,
        cacheTtlMinutes: 15,
        userAgent: "custom-ua",
      },
    },
  },
}
```

`tools.media`

Configures inbound media understanding (image/audio/video):

```
{
  tools: {
    media: {
      concurrency: 2,
      audio: {
        enabled: true,
        maxBytes: 20971520,
        scope: {
          default: "deny",
          rules: [{ action: "allow", match: { chatType: "direct" } }],
        },
        models: [
          { provider: "openai", model: "gpt-4o-mini-transcribe" },
          { type: "cli", command: "whisper", args: ["--model", "base", "{{MediaPa
        ],
      },
      video: {
        enabled: true,
        maxBytes: 52428800,
        models: [{ provider: "google", model: "gemini-3-flash-preview" }],
      },
    },
  },
}
```

Media model entry fields

tools.agentToAgent

```
{
  tools: {
    agentToAgent: {
      enabled: false,
      allow: ["home", "work"],
    },
  },
}
```

tools.sessions

Controls which sessions can be targeted by the session tools (`sessions_list` , `sessions_history` , `sessions_send`).

Default: `tree` (current session + sessions spawned by it, such as subagents).

```
{
  tools: {
    sessions: {
      // "self" | "tree" | "agent" | "all"
      visibility: "tree",
    },
  },
}
```

Notes:

- `self` : only the current session key.
- `tree` : current session + sessions spawned by the current session (subagents).
- `agent` : any session belonging to the current agent id (can include other users if you run per-sender sessions under the same agent id).

`all` : any session. Cross-agent targeting still requires `tools.agentToAgent` .

Sandbox clamp: when the current session is sandboxed and `agents.defaults.sandbox.sessionToolsVisibility="spawned"` , visibility is forced to `tree` even if `tools.sessions.visibility="all"` .

`tools.subagents`

```
{
  agents: {
    defaults: {
      subagents: {
        model: "minimax/MiniMax-M2.1",
        maxConcurrent: 1,
        archiveAfterMinutes: 60,
      },
    },
  },
}
```

`model` : default model for spawned sub-agents. If omitted, sub-agents inherit the caller's model.

Per-subagent tool policy: `tools.subagents.tools.allow` / `tools.subagents.tools.deny` .

Custom providers and base URLs

OpenClaw uses the pi-coding-agent model catalog. Add custom providers via `models.providers` in config or `~/.openclaw/agents/<agentId>/agent/models.json` .

```
{
  models: {
    mode: "merge", // merge (default) | replace
    providers: {
      "custom-proxy": {
        baseUrl: "http://localhost:4000/v1",
        apiKey: "LITELLM_KEY",
        api: "openai-completions", // openai-completions | openai-responses | ant
        models: [
          {
            id: "llama-3.1-8b",
            name: "Llama 3.1 8B",
            reasoning: false,
            input: ["text"],
            cost: { input: 0, output: 0, cacheRead: 0, cacheWrite: 0 },
            contextWindow: 128000,
            maxTokens: 32000,
          },
        ],
      },
    },
  },
}
```

Use `authHeader: true` + `headers` for custom auth needs.

Override agent config root with `OPENCLAW_AGENT_DIR` (or `PI_CODING_AGENT_DIR`).

Provider examples

Cerebras (GLM 4.6 / 4.7)

OpenCode Zen

Z.AI (GLM-4.7)

Moonshot AI (Kimi)

Kimi Coding

Synthetic (Anthropic-compatible)

MiniMax M2.1 (direct)

Local models (LM Studio)

Skills

```
{
  skills: {
    allowBundled: ["gemini", "peekaboo"],
    load: {
      extraDirs: ["~/Projects/agent-scripts/skills"],
    },
    install: {
      preferBrew: true,
      nodeManager: "npm", // npm | pnpm | yarn
    },
    entries: {
      "nano-banana-pro": {
        apiKey: "GEMINI_KEY_HERE",
        env: { GEMINI_API_KEY: "GEMINI_KEY_HERE" },
      },
      peekaboo: { enabled: true },
      sag: { enabled: false },
    },
  },
}
```

`allowBundled` : optional allowlist for bundled skills only (managed/workspace skills unaffected).

`entries.<skillKey>.enabled: false` disables a skill even if bundled/installed.

`entries.<skillKey>.apiKey` : convenience for skills declaring a primary env var.

Plugins

```
{
  plugins: {
    enabled: true,
    allow: ["voice-call"],
    deny: [],
    load: {
      paths: ["~/Projects/oss/voice-call-extension"],
    },
    entries: {
      "voice-call": {
        enabled: true,
        config: { provider: "twilio" },
      },
    },
  },
}
```

Loaded from `~/.openclaw/extensions` , `<workspace>/.openclaw/extensions` , plus `plugins.load.paths` .

Config changes require a gateway restart.

`allow` : optional allowlist (only listed plugins load). `deny` wins.

See .

Browser

```
{
  browser: {
    enabled: true,
    evaluateEnabled: true,
    defaultProfile: "chrome",
    profiles: {
      openclaw: { cdpPort: 18800, color: "#FF4500" },
      work: { cdpPort: 18801, color: "#0066CC" },
      remote: { cdpUrl: "http://10.0.0.42:9222", color: "#00AA00" },
    },
    color: "#FF4500",
    // headless: false,
    // noSandbox: false,
    // executablePath: "/Applications/Brave Browser.app/Contents/MacOS/Brave Brows
    // attachOnly: false,
  },
}
```

`evaluateEnabled: false` disables `act:evaluate` and `wait --fn`.

Remote profiles are attach-only (start/stop/reset disabled).

Auto-detect order: default browser if Chromium-based → Chrome → Brave → Edge → Chromium → Chrome Canary.

Control service: loopback only (port derived from `gateway.port`, default `18791`).

UI

```
{
  ui: {
    seamColor: "#FF4500",
    assistant: {
      name: "OpenClaw",
      avatar: "CB", // emoji, short text, image URL, or data URI
    },
  },
}
```

`seamColor` : accent color for native app UI chrome (Talk Mode bubble tint, etc.).

`assistant` : Control UI identity override. Falls back to active agent identity.

Gateway

```
  gateway: {
    mode: "local", // local | remote
    port: 18789,
    bind: "loopback",
    auth: {
      mode: "token", // token | password | trusted-proxy
      token: "your-token",
      // password: "your-password", // or OPENCLAW_GATEWAY_PASSWORD
      // trustedProxy: { userHeader: "x-forwarded-user" }, // for mode=trusted-pr
      allowTailscale: true,
      rateLimit: {
        maxAttempts: 10,
        windowMs: 60000,
        lockoutMs: 300000,
        exemptLoopback: true,
      },
    },
    tailscale: {
      mode: "off", // off | serve | funnel
      resetOnExit: false,
    },
    controlUi: {
      enabled: true,
      basePath: "/openclaw",
      // root: "dist/control-ui",
      // allowInsecureAuth: false,
      // dangerouslyDisableDeviceAuth: false,
    },
    remote: {
      url: "ws://gateway.tailnet:18789",
      transport: "ssh", // ssh | direct
      token: "your-token",
      // password: "your-password",
    },
    trustedProxies: ["10.0.0.1"],
    tools: {
      // Additional /tools/invoke HTTP denies
      deny: ["browser"],
      // Remove tools from the default HTTP deny list
      allow: ["gateway"],
    },
  },
}
```

Gateway field details

OpenAI-compatible endpoints

- Chat Completions: disabled by default. Enable with `gateway.http.endpoints.chatCompletions.enabled: true` .
- Responses API: `gateway.http.endpoints.responses.enabled` .
- Responses URL-input hardening:
  - `gateway.http.endpoints.responses.maxUrlParts`
  - `gateway.http.endpoints.responses.files.urlAllowlist`
  - `gateway.http.endpoints.responses.images.urlAllowlist`

Multi-instance isolation

Run multiple gateways on one host with unique ports and state dirs:

```
OPENCLAW_CONFIG_PATH=~/.openclaw/a.json \
OPENCLAW_STATE_DIR=~/.openclaw-a \
openclaw gateway --port 19001
```

Convenience flags: `--dev` (uses `~/.openclaw-dev` + port `19001`), `--profile <name>` (uses `~/.openclaw-<name>`).

See .

Hooks

```
{
  hooks: {
    enabled: true,
    token: "shared-secret",
    path: "/hooks",
    maxBodyBytes: 262144,
    defaultSessionKey: "hook:ingress",
    allowRequestSessionKey: false,
    allowedSessionKeyPrefixes: ["hook:"],
    allowedAgentIds: ["hooks", "main"],
    presets: ["gmail"],
    transformsDir: "~/.openclaw/hooks/transforms",
    mappings: [
      {
        match: { path: "gmail" },
        action: "agent",
        agentId: "hooks",
        wakeMode: "now",
        name: "Gmail",
        sessionKey: "hook:gmail:{{messages[0].id}}",
        messageTemplate: "From: {{messages[0].from}}\nSubject: {{messages[0].subj
        deliver: true,
        channel: "last",
        model: "openai/gpt-5.2-mini",
      },
    ],
  },
}
```

Auth: `Authorization: Bearer <token>` or `x-openclaw-token: <token>` .

Endpoints:

```
POST /hooks/wake → { text, mode?: "now"|"next-heartbeat" }
```

`POST /hooks/agent` → `{ message, name?, agentId?, sessionKey?, wakeMode?, deliver?, channel?, to?, model?, thinking?, timeoutSeconds? }`

`sessionKey` from request payload is accepted only when `hooks.allowRequestSessionKey=true` (default: `false`).

`POST /hooks/<name>` → resolved via `hooks.mappings`

Mapping details

Gmail integration

```
{
  hooks: {
    gmail: {
      account: "openclaw@gmail.com",
      topic: "projects/<project-id>/topics/gog-gmail-watch",
      subscription: "gog-gmail-watch-push",
      pushToken: "shared-push-token",
      hookUrl: "http://127.0.0.1:18789/hooks/gmail",
      includeBody: true,
      maxBytes: 20000,
      renewEveryMinutes: 720,
      serve: { bind: "127.0.0.1", port: 8788, path: "/" },
      tailscale: { mode: "funnel", path: "/gmail-pubsub" },
      model: "openrouter/meta-llama/llama-3.3-70b-instruct:free",
      thinking: "off",
    },
  },
}
```

Gateway auto-starts `gog gmail watch serve` on boot when configured. Set `OPENCLAW_SKIP_GMAIL_WATCHER=1` to disable.

Don't run a separate `gog gmail watch serve` alongside the Gateway.

Canvas host

```
{
  canvasHost: {
    root: "~/.openclaw/workspace/canvas",
    liveReload: true,
    // enabled: false, // or OPENCLAW_SKIP_CANVAS_HOST=1
  },
}
```

Serves agent-editable HTML/CSS/JS and A2UI over HTTP under the Gateway port:

`http://<gateway-host>:<gateway.port>/__openclaw__/canvas/`

`http://<gateway-host>:<gateway.port>/__openclaw__/a2ui/`

Local-only: keep `gateway.bind: "loopback"` (default).

Non-loopback binds: canvas routes require Gateway auth (token/password/trusted-proxy), same as other Gateway HTTP surfaces.

Node WebViews typically don't send auth headers; after a node is paired and connected, the Gateway allows a private-IP fallback so the node can load canvas/A2UI without leaking secrets into URLs.

Injects live-reload client into served HTML.

Auto-creates starter `index.html` when empty.

Also serves A2UI at `/__openclaw__/a2ui/` .

Changes require a gateway restart.

Disable live reload for large directories or `EMFILE` errors.

Discovery

mDNS (Bonjour)

```
{
  discovery: {
    mdns: {
      mode: "minimal", // minimal | full | off
    },
  },
}
```

- `minimal` (default): omit `cliPath` + `sshPort` from TXT records.
- `full` : include `cliPath` + `sshPort` .

Hostname defaults to `openclaw` . Override with `OPENCLAW_MDNS_HOSTNAME` .

Wide-area (DNS-SD)

```
{
  discovery: {
    wideArea: { enabled: true },
  },
}
```

Writes a unicast DNS-SD zone under `~/.openclaw/dns/` . For cross-network discovery, pair with a DNS server (CoreDNS recommended) + Tailscale split DNS.

Setup: `openclaw dns setup --apply` .

Environment

`env` (inline env vars)

```
{
  env: {
    OPENROUTER_API_KEY: "sk-or-...",
    vars: {
      GROQ_API_KEY: "gsk-...",
    },
    shellEnv: {
      enabled: true,
      timeoutMs: 15000,
    },
  },
}
```

Inline env vars are only applied if the process env is missing the key.

`.env` files: CWD `.env` + `~/.openclaw/.env` (neither overrides existing vars).

`shellEnv` : imports missing expected keys from your login shell profile.

See for full precedence.

Env var substitution

Reference env vars in any config string with `${VAR_NAME}` :

```
{
  gateway: {
    auth: { token: "${OPENCLAW_GATEWAY_TOKEN}" },
  },
}
```

Only uppercase names matched: `[A-Z_][A-Z0-9_]*` .

Missing/empty vars throw an error at config load.

Escape with `$${VAR}` for a literal `${VAR}`.

Works with `$include`.

›

Auth storage

```
{
  auth: {
    profiles: {
      "anthropic:me@example.com": { provider: "anthropic", mode: "oauth", email: "
      "anthropic:work": { provider: "anthropic", mode: "api_key" },
    },
    order: {
      anthropic: ["anthropic:me@example.com", "anthropic:work"],
    },
  },
}
```

Per-agent auth profiles stored at `<agentDir>/auth-profiles.json`.

Legacy OAuth imports from `~/.openclaw/credentials/oauth.json`.

See .

Logging

```
{
  logging: {
    level: "info",
    file: "/tmp/openclaw/openclaw.log",
    consoleLevel: "info",
    consoleStyle: "pretty", // pretty | compact | json
    redactSensitive: "tools", // off | tools
    redactPatterns: ["\\bTOKEN\\b\\s*[=:]\\s*([\"']?)([^\\s\"']+)\\1"],
  },
}
```

Default log file: `/tmp/openclaw/openclaw-YYYY-MM-DD.log` .

Set `logging.file` for a stable path.

`consoleLevel` bumps to `debug` when `--verbose` .

Wizard

Metadata written by CLI wizards (`onboard` , `configure` , `doctor`):

```
{
  wizard: {
    lastRunAt: "2026-01-01T00:00:00.000Z",
    lastRunVersion: "2026.1.4",
    lastRunCommit: "abc1234",
    lastRunCommand: "configure",
    lastRunMode: "local",
  },
}
```

Identity

```
{
  agents: {
    list: [
      {
        id: "main",
        identity: {
          name: "Samantha",
          theme: "helpful sloth",
          emoji: "🦥",
          avatar: "avatars/samantha.png",
        },
      },
    ],
  },
}
```

Written by the macOS onboarding assistant. Derives defaults:

- `messages.ackReaction` from `identity.emoji` (falls back to 👀)
- `mentionPatterns` from `identity.name` / `identity.emoji`
- `avatar` accepts: workspace-relative path, `http(s)` URL, or `data:` URI

Bridge (legacy, removed)

Current builds no longer include the TCP bridge. Nodes connect over the Gateway WebSocket. `bridge.*` keys are no longer part of the config schema (validation fails until removed; `openclaw doctor --fix` can strip unknown keys).

Legacy bridge config (historical reference)

Cron

```
{
  cron: {
    enabled: true,
    maxConcurrentRuns: 2,
    webhook: "https://example.invalid/legacy", // deprecated fallback for stored 
    webhookToken: "replace-with-dedicated-token", // optional bearer token for ou
    sessionRetention: "24h", // duration string or false
  },
}
```

- `sessionRetention` : how long to keep completed cron sessions before pruning. Default: `24h` .
- `webhookToken` : bearer token used for cron webhook POST delivery (`delivery.mode = "webhook"`), if omitted no auth header is sent.
- `webhook` : deprecated legacy fallback webhook URL (http/https) used only for stored jobs that still have `notify: true` .

See .

Media model template variables

Template placeholders expanded in `tools.media.*.models[].args` :

Variable	Description
`{{Body}}`	Full inbound message body
`{{RawBody}}`	Raw body (no history/sender wrappers)
`{{BodyStripped}}`	Body with group mentions stripped
`{{From}}`	Sender identifier

Variable	Description
`{{To}}`	Destination identifier
`{{MessageSid}}`	Channel message id
`{{SessionId}}`	Current session UUID
`{{IsNewSession}}`	`"true"` when new session created
`{{MediaUrl}}`	Inbound media pseudo-URL
`{{MediaPath}}`	Local media path
`{{MediaType}}`	Media type (image/audio/document/…)
`{{Transcript}}`	Audio transcript
`{{Prompt}}`	Resolved media prompt for CLI entries
`{{MaxChars}}`	Resolved max output chars for CLI entries
`{{ChatType}}`	`"direct"` or `"group"`
`{{GroupSubject}}`	Group subject (best effort)
`{{GroupMembers}}`	Group members preview (best effort)
`{{SenderName}}`	Sender display name (best effort)
`{{SenderE164}}`	Sender phone number (best effort)
`{{Provider}}`	Provider hint (whatsapp, telegram, discord, etc.)

Config includes (`$include`)

Split config into multiple files:

```
// ~/.openclaw/openclaw.json
{
  gateway: { port: 18789 },
  agents: { $include: "./agents.json5" },
  broadcast: {
    $include: ["./clients/mueller.json5", "./clients/schmidt.json5"],
  },
}
```

Merge behavior:

- Single file: replaces the containing object.
- Array of files: deep-merged in order (later overrides earlier).
- Sibling keys: merged after includes (override included values).
- Nested includes: up to 10 levels deep.
- Paths: resolved relative to the including file, but must stay inside the top-level config directory (`dirname` of the main config file). Absolute/ `../` forms are allowed only when they still resolve inside that boundary.
- Errors: clear messages for missing files, parse errors, and circular includes.

Related: · ·

‹ Configuration Configuration Examples ›