Agent coordination

Multi-Agent Sandbox & Tools

Overview

Each agent in a multi-agent setup can now have its own:

- **Sandbox configuration** (`agents.list[].sandbox` overrides `agents.defaults.sandbox`)
- **Tool restrictions** (`tools.allow` / `tools.deny` , plus `agents.list[].tools`)

This allows you to run multiple agents with different security profiles:

- Personal assistant with full access
- Family/work agents with restricted tools
- Public-facing agents in sandboxes

`setupCommand` belongs under `sandbox.docker` (global or per-agent) and runs once when the container is created.

Auth is per-agent: each agent reads from its own `agentDir` auth store at:

```
~/.openclaw/agents/<agentId>/agent/auth-profiles.json
```

Credentials are **not** shared between agents. Never reuse `agentDir` across agents. If you want to share creds, copy `auth-profiles.json` into the other agent's `agentDir` .

For how sandboxing behaves at runtime, see [Sandboxing]. For debugging "why is this blocked?", see [Sandbox vs Tool Policy vs Elevated] and `openclaw sandbox explain` .

Configuration Examples

Example 1: Personal + Restricted Family Agent

```
{
  "agents": {
    "list": [
      {
        "id": "main",
        "default": true,
        "name": "Personal Assistant",
        "workspace": "~/.openclaw/workspace",
        "sandbox": { "mode": "off" }
      },
      {
        "id": "family",
        "name": "Family Bot",
        "workspace": "~/.openclaw/workspace-family",
        "sandbox": {
          "mode": "all",
          "scope": "agent"
        },
        "tools": {
          "allow": ["read"],
          "deny": ["exec", "write", "edit", "apply_patch", "process", "browser"]
        }
      }
    ]
  },
  "bindings": [
    {
      "agentId": "family",
      "match": {
        "provider": "whatsapp",
        "accountId": "*",
        "peer": {
          "kind": "group",
          "id": "120363424282127706@g.us"
        }
      }
    }
  ]
}
```

Result:

- `main` agent: Runs on host, full tool access
- `family` agent: Runs in Docker (one container per agent), only `read` tool

Example 2: Work Agent with Shared Sandbox

```
{
  "agents": {
    "list": [
      {
        "id": "personal",
        "workspace": "~/.openclaw/workspace-personal",
        "sandbox": { "mode": "off" }
      },
      {
        "id": "work",
        "workspace": "~/.openclaw/workspace-work",
        "sandbox": {
          "mode": "all",
          "scope": "shared",
          "workspaceRoot": "/tmp/work-sandboxes"
        },
        "tools": {
          "allow": ["read", "write", "apply_patch", "exec"],
          "deny": ["browser", "gateway", "discord"]
        }
      }
    ]
  }
}
```

Example 2b: Global coding profile + messaging-only agent

```
{
  "tools": { "profile": "coding" },
  "agents": {
    "list": [
      {
        "id": "support",
        "tools": { "profile": "messaging", "allow": ["slack"] }
      }
    ]
  }
}
```

Result:

- default agents get coding tools
- `support` agent is messaging-only (+ Slack tool)

Example 3: Different Sandbox Modes per Agent

```
{
  "agents": {
    "defaults": {
      "sandbox": {
        "mode": "non-main", // Global default
        "scope": "session"
      }
    },
    "list": [
      {
        "id": "main",
        "workspace": "~/.openclaw/workspace",
        "sandbox": {
          "mode": "off" // Override: main never sandboxed
        }
      },
      {
        "id": "public",
        "workspace": "~/.openclaw/workspace-public",
        "sandbox": {
          "mode": "all", // Override: public always sandboxed
          "scope": "agent"
        },
        "tools": {
          "allow": ["read"],
          "deny": ["exec", "write", "edit", "apply_patch"]
        }
      }
    ]
  }
}
```

Configuration Precedence

When both global (`agents.defaults.*`) and agent-specific (`agents.list[].*`) configs exist:

Sandbox Config

Agent-specific settings override global:

```
agents.list[].sandbox.mode > agents.defaults.sandbox.mode
agents.list[].sandbox.scope > agents.defaults.sandbox.scope
agents.list[].sandbox.workspaceRoot > agents.defaults.sandbox.workspaceRoot
agents.list[].sandbox.workspaceAccess > agents.defaults.sandbox.workspaceAccess
agents.list[].sandbox.docker.* > agents.defaults.sandbox.docker.*
agents.list[].sandbox.browser.* > agents.defaults.sandbox.browser.*
agents.list[].sandbox.prune.* > agents.defaults.sandbox.prune.*
```

Notes:

- `agents.list[].sandbox.{docker,browser,prune}.*` overrides `agents.defaults.sandbox.{docker,browser,prune}.*` for that agent (ignored when sandbox scope resolves to `"shared"`).

Tool Restrictions

The filtering order is:

1. **Tool profile** (`tools.profile` or `agents.list[].tools.profile`)
2. **Provider tool profile** (`tools.byProvider[provider].profile` or `agents.list[].tools.byProvider[provider].profile`)
3. **Global tool policy** (`tools.allow` / `tools.deny`)
4. **Provider tool policy** (`tools.byProvider[provider].allow/deny`)
5. **Agent-specific tool policy** (`agents.list[].tools.allow/deny`)
6. **Agent provider policy** (`agents.list[].tools.byProvider[provider].allow/deny`)
7. **Sandbox tool policy** (`tools.sandbox.tools` or `agents.list[].tools.sandbox.tools`)
8. **Subagent tool policy** (`tools.subagents.tools` , if applicable)

Each level can further restrict tools, but cannot grant back denied tools from earlier levels. If `agents.list[].tools.sandbox.tools` is set, it replaces `tools.sandbox.tools` for that agent. If `agents.list[].tools.profile` is set, it overrides `tools.profile` for that agent. Provider tool keys accept either `provider` (e.g. `google-antigravity`) or `provider/model` (e.g. `openai/gpt-5.2`).

Tool groups (shorthands)

Tool policies (global, agent, sandbox) support `group:*` entries that expand to multiple concrete tools:

- `group:runtime` : `exec` , `bash` , `process`
- `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:ui` : `browser` , `canvas`
- `group:automation` : `cron` , `gateway`
- `group:messaging` : `message`
- `group:nodes` : `nodes`
- `group:openclaw` : all built-in OpenClaw tools (excludes provider plugins)

Elevated Mode

`tools.elevated` is the global baseline (sender-based allowlist). `agents.list[].tools.elevated` can further restrict elevated for specific agents (both must allow).

Mitigation patterns:

- Deny `exec` for untrusted agents (`agents.list[].tools.deny: ["exec"]`)

Avoid allowlisting senders that route to restricted agents

Disable elevated globally (`tools.elevated.enabled: false`) if you only want sandboxed execution

Disable elevated per agent (`agents.list[].tools.elevated.enabled: false`) for sensitive profiles

Migration from Single Agent

Before (single agent):

```
{
  "agents": {
    "defaults": {
      "workspace": "~/.openclaw/workspace",
      "sandbox": {
        "mode": "non-main"
      }
    }
  },
  "tools": {
    "sandbox": {
      "tools": {
        "allow": ["read", "write", "apply_patch", "exec"],
        "deny": []
      }
    }
  }
}
```

After (multi-agent with different profiles):

```
{
  "agents": {
    "list": [
      {
        "id": "main",
        "default": true,
        "workspace": "~/.openclaw/workspace",
        "sandbox": { "mode": "off" }
      }
    ]
  }
}
```

Legacy `agent.*` configs are migrated by `openclaw doctor` ; prefer `agents.defaults` + `agents.list` going forward.

Tool Restriction Examples

Read-only Agent

```
{
  "tools": {
    "allow": ["read"],
    "deny": ["exec", "write", "edit", "apply_patch", "process"]
  }
}
```

Safe Execution Agent (no file modifications)

```
{
  "tools": {
    "allow": ["read", "exec", "process"],
    "deny": ["write", "edit", "apply_patch", "browser", "gateway"]
  }
}
```

Communication-only Agent

```
{
  "tools": {
    "sessions": { "visibility": "tree" },
    "allow": ["sessions_list", "sessions_send", "sessions_history", "session_stat
    "deny": ["exec", "write", "edit", "apply_patch", "read", "browser"]
  }
}
```

Common Pitfall: "non-main"

`agents.defaults.sandbox.mode: "non-main"` is based on `session.mainKey` (default `"main"`), not the agent id. Group/channel sessions always get their own keys, so they are treated as non-main and will be sandboxed. If you want an agent to never sandbox, set `agents.list[].sandbox.mode: "off"`.

Testing

After configuring multi-agent sandbox and tools:

1. **Check agent resolution:**

```
openclaw agents list --bindings
```

2. **Verify sandbox containers:**

```
docker ps --filter "name=openclaw-sbx-"
```

3. **Test tool restrictions:**

   Send a message requiring restricted tools

   Verify the agent cannot use denied tools

4. **Monitor logs:**

```
tail -f "${OPENCLAW_STATE_DIR:-$HOME/.openclaw}/logs/gateway.log"
```

Troubleshooting

Agent not sandboxed despite `mode: "all"`

Check if there's a global `agents.defaults.sandbox.mode` that overrides it

Agent-specific config takes precedence, so set `agents.list[].sandbox.mode: "all"`

Tools still available despite deny list

Check tool filtering order: global → agent → sandbox → subagent

Each level can only further restrict, not grant back

Verify with logs: `[tools] filtering tools for agent:${agentId}`

Container not isolated per agent

Set `scope: "agent"` in agent-specific sandbox config

Default is `"session"` which creates one container per session

See Also

- Multi-Agent Routing
- Sandbox Configuration
- Session Management

‹ Sub-Agents Slash Commands ›

Powered by mintlify