Built-in tools

# Exec Tool

Run shell commands in the workspace. Supports foreground + background execution via `process`. If `process` is disallowed, `exec` runs synchronously and ignores `yieldMs` / `background`. Background sessions are scoped per agent; `process` only sees sessions from the same agent.

## Parameters

- `command` (required)
- `workdir` (defaults to cwd)
- `env` (key/value overrides)
- `yieldMs` (default 10000): auto-background after delay
- `background` (bool): background immediately
- `timeout` (seconds, default 1800): kill on expiry
- `pty` (bool): run in a pseudo-terminal when available (TTY-only CLIs, coding agents, terminal UIs)
- `host` ( `sandbox` | `gateway` | `node` ): where to execute
- `security` ( `deny` | `allowlist` | `full` ): enforcement mode for `gateway` / `node`
- `ask` ( `off` | `on-miss` | `always` ): approval prompts for `gateway` / `node`
- `node` (string): node id/name for `host=node`
- `elevated` (bool): request elevated mode (gateway host); `security=full` is only forced when elevated resolves to `full`

Notes:

- `host` defaults to `sandbox`.
- `elevated` is ignored when sandboxing is off (exec already runs on the host).
- `gateway` / `node` approvals are controlled by `~/.openclaw/exec-approvals.json`.
- `node` requires a paired node (companion app or headless node host).
- If multiple nodes are available, set `exec.node` or `tools.exec.node` to select one.
- On non-Windows hosts, exec uses `SHELL` when set; if `SHELL` is `fish`, it prefers `bash` (or `sh`) from `PATH` to avoid fish-incompatible scripts, then falls back to `SHELL` if neither exists.
- Host execution (`gateway` / `node`) rejects `env.PATH` and loader overrides (`LD_*` / `DYLD_*`) to prevent binary hijacking or injected code.
- Important: sandboxing is **off by default**. If sandboxing is off, `host=sandbox` runs directly on the gateway host (no container) and **does not require approvals**. To require approvals, run with `host=gateway` and configure exec approvals (or enable sandboxing).

## Config

- `tools.exec.notifyOnExit` (default: true): when true, backgrounded exec sessions enqueue a system event and request a heartbeat on exit.
- `tools.exec.approvalRunningNoticeMs` (default: 10000): emit a single “running” notice when an approval-gated exec runs longer than this (0 disables).
- `tools.exec.host` (default: `sandbox`)
- `tools.exec.security` (default: `deny` for sandbox, `allowlist` for gateway + node when unset)

- `tools.exec.ask` (default: `on-miss`)
- `tools.exec.node` (default: unset)
- `tools.exec.pathPrepend`: list of directories to prepend to `PATH` for exec runs (gateway + sandbox only).
- `tools.exec.safeBins`: stdin-only safe binaries that can run without explicit allowlist entries. For behavior details, see **Safe bins**.

Example:

```
{
  tools: {
    exec: {
      pathPrepend: ["~/bin", "/opt/oss/bin"],
    },
  },
}
```

## PATH handling

- `host=gateway`: merges your login-shell `PATH` into the exec environment. `env.PATH` overrides are rejected for host execution. The daemon itself still runs with a minimal `PATH`:
  - macOS: `/opt/homebrew/bin`, `/usr/local/bin`, `/usr/bin`, `/bin`
  - Linux: `/usr/local/bin`, `/usr/bin`, `/bin`
- `host=sandbox`: runs `sh -lc` (login shell) inside the container, so `/etc/profile` may reset `PATH`. OpenClaw prepends `env.PATH` after profile sourcing via an internal env var (no shell interpolation); `tools.exec.pathPrepend` applies here too.
- `host=node`: only non-blocked env overrides you pass are sent to the node. `env.PATH` overrides are rejected for host execution and ignored by node hosts. If you need additional PATH entries on a node, configure the node host service environment (systemd/launchd) or install tools in standard locations.

Per-agent node binding (use the agent list index in config):

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

Control UI: the Nodes tab includes a small “Exec node binding” panel for the same settings.

## Session overrides ( `/exec` )

Use `/exec` to set **per-session** defaults for `host` , `security` , `ask` , and `node` . Send `/exec` with no arguments to show the current values.

Example:

```
/exec host=gateway security=allowlist ask=on-miss node=mac-1
```

## Authorization model

`/exec` is only honored for **authorized senders** (channel allowlists/pairing plus `commands.useAccessGroups` ). It updates **session state only** and does not write config. To hard-disable exec, deny it via tool policy ( `tools.deny: ["exec"]` or per-agent). Host approvals still apply unless you explicitly set `security=full` and `ask=off` .

## Exec approvals (companion app / node host)

Sandboxed agents can require per-request approval before `exec` runs on the gateway or node host. See for the policy, allowlist, and UI flow.

When approvals are required, the exec tool returns immediately with `status: "approval-pending"` and an approval id. Once approved (or denied /

timed out), the Gateway emits system events ( `Exec finished` / `Exec denied` ). If the command is still running after `tools.exec.approvalRunningNoticeMs` , a single `Exec running` notice is emitted.

## Allowlist + safe bins

Allowlist enforcement matches **resolved binary paths only** (no basename matches). When `security=allowlist` , shell commands are auto-allowed only if every pipeline segment is allowlisted or a safe bin. Chaining ( `;` , `&&` , `||` ) and redirections are rejected in allowlist mode unless every top-level segment satisfies the allowlist (including safe bins). Redirections remain unsupported.

## Examples

Foreground:

```
{ "tool": "exec", "command": "ls -la" }
```

Background + poll:

```
{"tool":"exec","command":"npm run build","yieldMs":1000}
{"tool":"process","action":"poll","sessionId":"<id>"}
```

Send keys (tmux-style):

```
{"tool":"process","action":"send-keys","sessionId":"<id>","keys":["Ent
{"tool":"process","action":"send-keys","sessionId":"<id>","keys":["C-c"]}
{"tool":"process","action":"send-keys","sessionId":"<id>","keys":["Up","Up","Ente
```

Submit (send CR only):

```
{ "tool": "process", "action": "submit", "sessionId": "<id>" }
```

Paste (bracketed by default):

```
{ "tool": "process", "action": "paste", "sessionId": "<id>", "text": '
```

## apply_patch (experimental)

`apply_patch` is a subtool of `exec` for structured multi-file edits. Enable it explicitly:

```
{
  tools: {
    exec: {
      applyPatch: { enabled: true, workspaceOnly: true, allowModels: ["gpt-5.2"] 
    },
  },
}
```

Notes:

- Only available for OpenAI/OpenAI Codex models.
- Tool policy still applies; `allow: ["exec"]` implicitly allows `apply_patch` .
- Config lives under `tools.exec.applyPatch` .
- `tools.exec.applyPatch.workspaceOnly` defaults to `true` (workspace-contained). Set it to `false` only if you intentionally want `apply_patch` to write/delete outside the workspace directory.

‹ LLM Task Web Tools ›