Protocols and APIs

CLI Backends

OpenClaw can run **local AI CLIs** as a **text-only fallback** when API providers are down, rate-limited, or temporarily misbehaving. This is intentionally conservative:

- **Tools are disabled** (no tool calls).
- **Text in → text out** (reliable).
- **Sessions are supported** (so follow-up turns stay coherent).
- **Images can be passed through** if the CLI accepts image paths.

This is designed as a **safety net** rather than a primary path. Use it when you want “always works” text responses without relying on external APIs.

Beginner-friendly quick start

You can use Claude Code CLI **without any config** (OpenClaw ships a built-in default):

```
openclaw agent --message "hi" --model claude-cli/opus-4.6
```

Codex CLI also works out of the box:

```
openclaw agent --message "hi" --model codex-cli/gpt-5.3-codex
```

If your gateway runs under launchd/systemd and PATH is minimal, add just the command path:

```
{
  agents: {
    defaults: {
      cliBackends: {
        "claude-cli": {
          command: "/opt/homebrew/bin/claude",
        },
      },
    },
  },
}
```

That’s it. No keys, no extra auth config needed beyond the CLI itself.

Using it as a fallback

Add a CLI backend to your fallback list so it only runs when primary models fail:

```
{
  agents: {
    defaults: {
      model: {
        primary: "anthropic/claude-opus-4-6",
        fallbacks: ["claude-cli/opus-4.6", "claude-cli/opus-4.5"],
      },
      models: {
        "anthropic/claude-opus-4-6": { alias: "Opus" },
        "claude-cli/opus-4.6": {},
        "claude-cli/opus-4.5": {},
      },
    },
  },
}
```

Notes:

- If you use `agents.defaults.models` (allowlist), you must include `claude-cli/...`.
- If the primary provider fails (auth, rate limits, timeouts), OpenClaw will try the CLI backend next.

Configuration overview

All CLI backends live under:

```
agents.defaults.cliBackends
```

Each entry is keyed by a **provider id** (e.g. `claude-cli`, `my-cli`). The provider id becomes the left side of your model ref:

```
<provider>/<model>
```

Example configuration

```
{
  agents: {
    defaults: {
      cliBackends: {
        "claude-cli": {
          command: "/opt/homebrew/bin/claude",
        },
        "my-cli": {
          command: "my-cli",
          args: ["--json"],
          output: "json",
          input: "arg",
          modelArg: "--model",
          modelAliases: {
            "claude-opus-4-6": "opus",
            "claude-opus-4-5": "opus",
            "claude-sonnet-4-5": "sonnet",
          },
          sessionArg: "--session",
          sessionMode: "existing",
          sessionIdFields: ["session_id", "conversation_id"],
          systemPromptArg: "--system",
          systemPromptWhen: "first",
          imageArg: "--image",
          imageMode: "repeat",
          serialize: true,
        },
      },
    },
  },
}
```

How it works

1. **Selects a backend** based on the provider prefix (`claude-cli/...`).

2. **Builds a system prompt** using the same OpenClaw prompt + workspace context.
3. **Executes the CLI** with a session id (if supported) so history stays consistent.
4. **Parses output** (JSON or plain text) and returns the final text.
5. **Persists session ids** per backend, so follow-ups reuse the same CLI session.

Sessions

If the CLI supports sessions, set `sessionArg` (e.g. `--session-id`) or `sessionArgs` (placeholder `{sessionId}`) when the ID needs to be inserted into multiple flags.

If the CLI uses a **resume subcommand** with different flags, set `resumeArgs` (replaces `args` when resuming) and optionally `resumeOutput` (for non-JSON resumes).

`sessionMode`:

- `always`: always send a session id (new UUID if none stored).
- `existing`: only send a session id if one was stored before.
- `none`: never send a session id.

Images (pass-through)

If your CLI accepts image paths, set `imageArg`:

```
imageArg: "--image",
imageMode: "repeat"
```

OpenClaw will write base64 images to temp files. If `imageArg` is set, those paths are passed as CLI args. If `imageArg` is missing, OpenClaw appends the file paths to the prompt (path injection), which is enough

for CLIs that auto- load local files from plain paths (Claude Code CLI behavior).

Inputs / outputs

`output: "json"` (default) tries to parse JSON and extract text + session id.

`output: "jsonl"` parses JSONL streams (Codex CLI `--json`) and extracts the last agent message plus `thread_id` when present.

`output: "text"` treats stdout as the final response.

Input modes:

`input: "arg"` (default) passes the prompt as the last CLI arg.

`input: "stdin"` sends the prompt via stdin.

If the prompt is very long and `maxPromptArgChars` is set, stdin is used.

Defaults (built-in)

OpenClaw ships a default for `claude-cli`:

```
command: "claude"
args: ["-p", "--output-format", "json", "--dangerously-skip-permissions"]
resumeArgs: ["-p", "--output-format", "json", "--dangerously-skip-permissions", "--resume", "{sessionId}"]
modelArg: "--model"
systemPromptArg: "--append-system-prompt"
sessionArg: "--session-id"
systemPromptWhen: "first"
sessionMode: "always"
```

OpenClaw also ships a default for `codex-cli`:

```
command: "codex"
args: ["exec","--json","--color","never","--sandbox","read-only","--skip-git-repo-check"]
resumeArgs: ["exec","resume","{sessionId}","--color","never","--sandbox","read-only","--skip-git-repo-check"]
output: "jsonl"
resumeOutput: "text"
modelArg: "--model"
imageArg: "--image"
sessionMode: "existing"
```

Override only if needed (common: absolute `command` path).

Limitations

No OpenClaw tools (the CLI backend never receives tool calls). Some CLIs may still run their own agent tooling.

No streaming (CLI output is collected then returned).

Structured outputs depend on the CLI's JSON format.

Codex CLI sessions resume via text output (no JSONL), which is less structured than the initial `--json` run. OpenClaw sessions still work normally.

Troubleshooting

CLI not found: set `command` to a full path.

Wrong model name: use `modelAliases` to map `provider/model` → CLI model.

No session continuity: ensure `sessionArg` is set and `sessionMode` is not `none` (Codex CLI currently cannot resume with JSON output).

Images ignored: set `imageArg` (and verify CLI supports file paths).

‹ Tools Invoke API

Local Models ›