



---

≡ Configuration and operations > **Background Exec and Process Tool**

---

## Configuration and operations

# Background Exec and Process Tool

OpenClaw runs shell commands through the `exec` tool and keeps long-running tasks in memory. The `process` tool manages those background sessions.

## exec tool

Key parameters:

`command` (required)

`yieldMs` (default 10000): auto-background after this delay

`background` (bool): background immediately

`timeout` (seconds, default 1800): kill the process after this timeout

`elevated` (bool): run on host if elevated mode is enabled/allowed

Need a real TTY? Set `pty: true` .

`workdir` , `env`

Behavior:

Foreground runs return output directly.

When backgrounded (explicit or timeout), the tool returns `status:`  
"running" + `sessionId` and a short tail.

Output is kept in memory until the session is polled or cleared.



If the `process tool` is disallowed, `exec` runs synchronously and ignores `yieldMs / background` .

&gt;

## Child process bridging

When spawning long-running child processes outside the `exec/process` tools (for example, CLI respawns or gateway helpers), attach the `child-process bridge` helper so termination signals are forwarded and listeners are detached on `exit/error`. This avoids orphaned processes on `systemd` and keeps shutdown behavior consistent across platforms.

Environment overrides:

`PI_BASH_YIELD_MS` : default yield (ms)

`PI_BASH_MAX_OUTPUT_CHARS` : in-memory output cap (chars)

`OPENCLAW_BASH_PENDING_MAX_OUTPUT_CHARS` : pending stdout/stderr cap per stream (chars)

`PI_BASH_JOB_TTL_MS` : TTL for finished sessions (ms, bounded to 1m-3h)

Config (preferred):

`tools.exec.backgroundMs` (default 10000)

`tools.exec.timeoutSec` (default 1800)

`tools.exec.cleanupMs` (default 1800000)

`tools.exec.notifyOnExit` (default true): enqueue a system event + request heartbeat when a backgrounded `exec` exits.

`tools.exec.notifyOnExitEmptySuccess` (default false): when true, also enqueue completion events for successful backgrounded runs that produced no output.

## process tool

Actions:



```
list : running + finished sessions
poll : drain new output for a session (also reports exit status)
log  : read the aggregated, output (supports offset + limit)
write : send stdin ( data , optional eof )
kill  : terminate a background session
clear : remove a finished session from memory
remove : kill if running, otherwise clear if finished
```

## Notes:

Only backgrounded sessions are listed/persisted in memory.

Sessions are lost on process restart (no disk persistence).

Session logs are only saved to chat history if you run `process poll/log` and the tool result is recorded.

`process` is scoped per agent; it only sees sessions started by that agent.

`process list` includes a derived `name` (command verb + target) for quick scans.

`process log` uses line-based `offset / limit` .


When both `offset` and `limit` are omitted, it returns the last 200 lines and includes a paging hint.

When `offset` is provided and `limit` is omitted, it returns from `offset` to the end (not capped to 200).

## Examples

Run a long task and poll later:

```
{ "tool": "exec", "command": "sleep 5 && echo done", "yieldMs": 1000 }
```

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

Start immediately in background:

`{ "tool": "exec", "command": "npm run build", "background": true }`

Send stdin:

`{ "tool": "process", "action": "write", "sessionId": "<id>", "data": ' ' }`

< Gateway Lock

Multiple Gateways >

Powered by [mintlify](#)