Messaging platforms

Signal

Status: external CLI integration. Gateway talks to `signal-cli` over HTTP JSON-RPC + SSE.

Prerequisites

OpenClaw installed on your server (Linux flow below tested on Ubuntu 24).

`signal-cli` available on the host where the gateway runs.

A phone number that can receive one verification SMS (for SMS registration path).

Browser access for Signal captcha (`signalcaptchas.org`) during registration.

Quick setup (beginner)

1. Use a **separate Signal number** for the bot (recommended).
2. Install `signal-cli` (Java required if you use the JVM build).
3. Choose one setup path:

   Path A (QR link): `signal-cli link -n "OpenClaw"` and scan with Signal.

   Path B (SMS register): register a dedicated number with captcha + SMS verification.

4. Configure OpenClaw and restart the gateway.
5. Send a first DM and approve pairing (`openclaw pairing approve signal <CODE>`).

Minimal config:

```
{
  channels: {
    signal: {
      enabled: true,
      account: "+15551234567",
      cliPath: "signal-cli",
      dmPolicy: "pairing",
      allowFrom: ["+15557654321"],
    },
  },
}
```

Field reference:

Field	Description
`account`	Bot phone number in E.164 format (`+15551234567`)
`cliPath`	Path to `signal-cli` (`signal-cli` if on `PATH`)
`dmPolicy`	DM access policy (`pairing` recommended)
`allowFrom`	Phone numbers or `uuid:<id>` values allowed to DM

What it is

Signal channel via `signal-cli` (not embedded libsignal).

Deterministic routing: replies always go back to Signal.

DMs share the agent's main session; groups are isolated (`agent:<agentId>:signal:group:<groupId>`).

Config writes

By default, Signal is allowed to write config updates triggered by `/config set|unset` (requires `commands.config: true`).

Disable with:

```
{
  channels: { signal: { configWrites: false } },
}
```

The number model (important)

The gateway connects to a **Signal device** (the `signal-cli` account).

If you run the bot on **your personal Signal account**, it will ignore your own messages (loop protection).

For “I text the bot and it replies,” use a **separate bot number**.

Setup path A: link existing Signal account (QR)

1. Install `signal-cli` (JVM or native build).
2. Link a bot account:

   `signal-cli link -n "OpenClaw"` then scan the QR in Signal.
3. Configure Signal and start the gateway.

Example:

```
{
  channels: {
    signal: {
      enabled: true,
      account: "+15551234567",
      cliPath: "signal-cli",
      dmPolicy: "pairing",
      allowFrom: ["+15557654321"],
    },
  },
}
```

Multi-account support: use `channels.signal.accounts` with per-account config and optional `name`. See for the shared pattern.

Setup path B: register dedicated bot number (SMS, Linux)

Use this when you want a dedicated bot number instead of linking an existing Signal app account.

1. Get a number that can receive SMS (or voice verification for landlines).

   Use a dedicated bot number to avoid account/session conflicts.

2. Install `signal-cli` on the gateway host:

```
VERSION=$(curl -Ls -o /dev/null -w %{url_effective} https://github.com
curl -L -O "https://github.com/AsamK/signal-cli/releases/download/v${VERSION}/sig
sudo tar xf "signal-cli-${VERSION}-Linux-native.tar.gz" -C /opt
sudo ln -sf /opt/signal-cli /usr/local/bin/
signal-cli --version
```

If you use the JVM build (`signal-cli-${VERSION}.tar.gz`), install JRE 25+ first. Keep `signal-cli` updated; upstream notes that old releases can break as Signal server APIs change.

3. Register and verify the number:

```
signal-cli -a +<BOT_PHONE_NUMBER> register
```

If captcha is required:

1. Open `https://signalcaptchas.org/registration/generate.html` .
2. Complete captcha, copy the `signalcaptcha://...` link target from "Open Signal".
3. Run from the same external IP as the browser session when possible.
4. Run registration again immediately (captcha tokens expire quickly):

```
signal-cli -a +<BOT_PHONE_NUMBER> register --captcha '<SIGNALCAPTCHA_L
signal-cli -a +<BOT_PHONE_NUMBER> verify <VERIFICATION_CODE>
```

4. Configure OpenClaw, restart gateway, verify channel:

```
# If you run the gateway as a user systemd service:
systemctl --user restart openclaw-gateway

# Then verify:
openclaw doctor
openclaw channels status --probe
```

5. Pair your DM sender:

   Send any message to the bot number.

Approve code on the server: `openclaw pairing approve signal <PAIRING_CODE>`.

Save the bot number as a contact on your phone to avoid “Unknown contact”.

Important: registering a phone number account with `signal-cli` can de-authenticate the main Signal app session for that number. Prefer a dedicated bot number, or use QR link mode if you need to keep your existing phone app setup.

Upstream references:

- `signal-cli` README: `https://github.com/AsamK/signal-cli`
- Captcha flow: `https://github.com/AsamK/signal-cli/wiki/Registration-with-captcha`
- Linking flow: `https://github.com/AsamK/signal-cli/wiki/Linking-other-devices-(Provisioning)`

External daemon mode (httpUrl)

If you want to manage `signal-cli` yourself (slow JVM cold starts, container init, or shared CPUs), run the daemon separately and point OpenClaw at it:

```
{
  channels: {
    signal: {
      httpUrl: "http://127.0.0.1:8080",
      autoStart: false,
    },
  },
}
```

This skips auto-spawn and the startup wait inside OpenClaw. For slow starts when auto-spawning, set `channels.signal.startupTimeoutMs`.

Access control (DMs + groups)

DMs:

- Default: `channels.signal.dmPolicy = "pairing"`.
- Unknown senders receive a pairing code; messages are ignored until approved (codes expire after 1 hour).
- Approve via:

```
openclaw pairing list signal
openclaw pairing approve signal <CODE>
```

- Pairing is the default token exchange for Signal DMs. Details: Pairing
- UUID-only senders (from `sourceUuid`) are stored as `uuid:<id>` in `channels.signal.allowFrom`.

Groups:

- `channels.signal.groupPolicy = open | allowlist | disabled`.
- `channels.signal.groupAllowFrom` controls who can trigger in groups when `allowlist` is set.

How it works (behavior)

- `signal-cli` runs as a daemon; the gateway reads events via SSE.
- Inbound messages are normalized into the shared channel envelope.
- Replies always route back to the same number or group.

Media + limits

Outbound text is chunked to `channels.signal.textChunkLimit` (default 4000).

Optional newline chunking: set `channels.signal.chunkMode="newline"` to split on blank lines (paragraph boundaries) before length chunking.

Attachments supported (base64 fetched from `signal-cli`).

Default media cap: `channels.signal.mediaMaxMb` (default 8).

Use `channels.signal.ignoreAttachments` to skip downloading media.

Group history context uses `channels.signal.historyLimit` (or `channels.signal.accounts.*.historyLimit`), falling back to `messages.groupChat.historyLimit`. Set `0` to disable (default 50).

Typing + read receipts

Typing indicators: OpenClaw sends typing signals via `signal-cli sendTyping` and refreshes them while a reply is running.

Read receipts: when `channels.signal.sendReadReceipts` is true, OpenClaw forwards read receipts for allowed DMs.

Signal-cli does not expose read receipts for groups.

Reactions (message tool)

Use `message action=react` with `channel=signal`.

Targets: sender E.164 or UUID (use `uuid:<id>` from pairing output; bare UUID works too).

`messageId` is the Signal timestamp for the message you're reacting to.

Group reactions require `targetAuthor` or `targetAuthorUuid`.

Examples:

```
message action=react channel=signal target=uuid:123e4567-e89b-12d3-a45
message action=react channel=signal target=+15551234567 messageId=1737630212345 em
message action=react channel=signal target=signal:group:<groupId> targetAuthor=uui
                        >
```

Config:

- `channels.signal.actions.reactions` : enable/disable reaction actions (default true).
- `channels.signal.reactionLevel` : `off | ack | minimal | extensive` .
  - `off` / `ack` disables agent reactions (message tool `react` will error).
  - `minimal` / `extensive` enables agent reactions and sets the guidance level.

Per-account overrides: `channels.signal.accounts.<id>.actions.reactions` , `channels.signal.accounts.<id>.reactionLevel` .

Delivery targets (CLI/cron)

DMs: `signal:+15551234567` (or plain E.164).

UUID DMs: `uuid:<id>` (or bare UUID).

Groups: `signal:group:<groupId>` .

Usernames: `username:<name>` (if supported by your Signal account).

Troubleshooting

Run this ladder first:

```
openclaw status
openclaw gateway status
openclaw logs --follow
openclaw doctor
openclaw channels status --probe
```

Then confirm DM pairing state if needed:

```
openclaw pairing list signal
```

Common failures:

- Daemon reachable but no replies: verify account/daemon settings (`httpUrl`, `account`) and receive mode.
- DMs ignored: sender is pending pairing approval.
- Group messages ignored: group sender/mention gating blocks delivery.
- Config validation errors after edits: run `openclaw doctor --fix`.
- Signal missing from diagnostics: confirm `channels.signal.enabled: true`.

Extra checks:

```
openclaw pairing list signal
pgrep -af signal-cli
grep -i "signal" "/tmp/openclaw/openclaw-$(date +%Y-%m-%d).log" | tail -20
```

For triage flow: .

Security notes

`signal-cli` stores account keys locally (typically `~/.local/share/signal-cli/data/`).

Back up Signal account state before server migration or rebuild.

Keep `channels.signal.dmPolicy: "pairing"` unless you explicitly want broader DM access.

SMS verification is only needed for registration or recovery flows, but losing control of the number/account can complicate re-registration.

Configuration reference (Signal)

Full configuration: **Configuration**

Provider options:

- `channels.signal.enabled`: enable/disable channel startup.
- `channels.signal.account`: E.164 for the bot account.
- `channels.signal.cliPath`: path to `signal-cli`.
- `channels.signal.httpUrl`: full daemon URL (overrides host/port).
- `channels.signal.httpHost`, `channels.signal.httpPort`: daemon bind (default 127.0.0.1:8080).
- `channels.signal.autoStart`: auto-spawn daemon (default true if `httpUrl` unset).
- `channels.signal.startupTimeoutMs`: startup wait timeout in ms (cap 120000).
- `channels.signal.receiveMode`: `on-start | manual`.
- `channels.signal.ignoreAttachments`: skip attachment downloads.
- `channels.signal.ignoreStories`: ignore stories from the daemon.
- `channels.signal.sendReadReceipts`: forward read receipts.
- `channels.signal.dmPolicy`: `pairing | allowlist | open | disabled` (default: pairing).

`channels.signal.allowFrom` : DM allowlist (E.164 or `uuid:<id>`). `open` requires `"*"` . Signal has no usernames; use phone/UUID ids.

`channels.signal.groupPolicy` : `open | allowlist | disabled` (default: allowlist).

`channels.signal.groupAllowFrom` : group sender allowlist.

`channels.signal.historyLimit` : max group messages to include as context (0 disables).

`channels.signal.dmHistoryLimit` : DM history limit in user turns. Per-user overrides: `channels.signal.dms["<phone_or_uuid>"].historyLimit` .

`channels.signal.textChunkLimit` : outbound chunk size (chars).

`channels.signal.chunkMode` : `length` (default) or `newline` to split on blank lines (paragraph boundaries) before length chunking.

`channels.signal.mediaMaxMb` : inbound/outbound media cap (MB).

Related global options:

`agents.list[].groupChat.mentionPatterns` (Signal does not support native mentions).

`messages.groupChat.mentionPatterns` (global fallback).

`messages.responsePrefix` .

‹ Mattermost iMessage ›