Fundamentals

Agent Runtime

OpenClaw runs a single embedded agent runtime derived from **pi-mono**.

Workspace (required)

OpenClaw uses a single agent workspace directory (`agents.defaults.workspace`) as the agent's **only** working directory (`cwd`) for tools and context.

Recommended: use `openclaw setup` to create `~/.openclaw/openclaw.json` if missing and initialize the workspace files.

Full workspace layout + backup guide: **Agent workspace**

If `agents.defaults.sandbox` is enabled, non-main sessions can override this with per-session workspaces under `agents.defaults.sandbox.workspaceRoot` (see **Gateway configuration**).

Bootstrap files (injected)

Inside `agents.defaults.workspace` , OpenClaw expects these user-editable files:

- `AGENTS.md` — operating instructions + "memory"
- `SOUL.md` — persona, boundaries, tone

- `TOOLS.md` — user-maintained tool notes (e.g. `imsg`, `sag`, conventions)
- `BOOTSTRAP.md` — one-time first-run ritual (deleted after completion)
- `IDENTITY.md` — agent name/vibe/emoji
- `USER.md` — user profile + preferred address

On the first turn of a new session, OpenClaw injects the contents of these files directly into the agent context.

Blank files are skipped. Large files are trimmed and truncated with a marker so prompts stay lean (read the file for full content).

If a file is missing, OpenClaw injects a single “missing file” marker line (and `openclaw setup` will create a safe default template).

`BOOTSTRAP.md` is only created for a **brand new workspace** (no other bootstrap files present). If you delete it after completing the ritual, it should not be recreated on later restarts.

To disable bootstrap file creation entirely (for pre-seeded workspaces), set:

```
{ agent: { skipBootstrap: true } }
```

Built-in tools

Core tools (read/exec/edit/write and related system tools) are always available, subject to tool policy. `apply_patch` is optional and gated by `tools.exec.applyPatch`. `TOOLS.md` does **not** control which tools exist; it’s guidance for how *you* want them used.

Skills

OpenClaw loads skills from three locations (workspace wins on name conflict):

- Bundled (shipped with the install)
- Managed/local: `~/.openclaw/skills`
- Workspace: `<workspace>/skills`

Skills can be gated by config/env (see `skills` in **Gateway configuration**).

pi-mono integration

OpenClaw reuses pieces of the pi-mono codebase (models/tools), but **session management, discovery, and tool wiring are OpenClaw-owned**.

- No pi-coding agent runtime.
- No `~/.pi/agent` or `<workspace>/.pi` settings are consulted.

Sessions

Session transcripts are stored as JSONL at:

```
~/.openclaw/agents/<agentId>/sessions/<SessionId>.jsonl
```

The session ID is stable and chosen by OpenClaw. Legacy Pi/Tau session folders are **not** read.

Steering while streaming

When queue mode is `steer`, inbound messages are injected into the current run. The queue is checked **after each tool call**; if a queued message is present, remaining tool calls from the current assistant message are skipped (error tool results with “Skipped due to queued

user message."), then the queued user message is injected before the next assistant response.

When queue mode is `followup` or `collect`, inbound messages are held until the current turn ends, then a new agent turn starts with the queued payloads. See **Queue** for mode + debounce/cap behavior.

Block streaming sends completed assistant blocks as soon as they finish; it is **off by default** (`agents.defaults.blockStreamingDefault: "off"`). Tune the boundary via `agents.defaults.blockStreamingBreak` (`text_end` vs `message_end`; defaults to text_end). Control soft block chunking with `agents.defaults.blockStreamingChunk` (defaults to 800–1200 chars; prefers paragraph breaks, then newlines; sentences last). Coalesce streamed chunks with `agents.defaults.blockStreamingCoalesce` to reduce single-line spam (idle-based merging before send). Non-Telegram channels require explicit `*.blockStreaming: true` to enable block replies. Verbose tool summaries are emitted at tool start (no debounce); Control UI streams tool output via agent events when available. More details: **Streaming + chunking**.

Model refs

Model refs in config (for example `agents.defaults.model` and `agents.defaults.models`) are parsed by splitting on the **first** `/`.

- Use `provider/model` when configuring models.
- If the model ID itself contains `/` (OpenRouter-style), include the provider prefix (example: `openrouter/moonshotai/kimi-k2`).
- If you omit the provider, OpenClaw treats the input as an alias or a model for the **default provider** (only works when there is no `/` in the model ID).

Configuration (minimal)

At minimum, set:

- `agents.defaults.workspace`
- `channels.whatsapp.allowFrom` (strongly recommended)

*Next: **Group Chats***

‹ Gateway Architecture

Agent Loop ›