Sessions and memory

Session Management

OpenClaw treats **one direct-chat session per agent** as primary. Direct chats collapse to `agent:<agentId>:<mainKey>` (default `main`), while group/channel chats get their own keys. `session.mainKey` is honored.

Use `session.dmScope` to control how **direct messages** are grouped:

- `main` (default): all DMs share the main session for continuity.
- `per-peer` : isolate by sender id across channels.
- `per-channel-peer` : isolate by channel + sender (recommended for multi-user inboxes).
- `per-account-channel-peer` : isolate by account + channel + sender (recommended for multi-account inboxes). Use `session.identityLinks` to map provider-prefixed peer ids to a canonical identity so the same person shares a DM session across channels when using `per-peer` , `per-channel-peer` , or `per-account-channel-peer` .

Secure DM mode (recommended for multi-user setups)

> **Security Warning:** If your agent can receive DMs from **multiple people**, you should strongly consider enabling secure DM mode. Without it, all users share the same conversation context, which can leak private information between users.

Example of the problem with default settings:

- Alice (`<SENDER_A>`) messages your agent about a private topic (for example, a medical appointment)
- Bob (`<SENDER_B>`) messages your agent asking “What were we talking about?”
- Because both DMs share the same session, the model may answer Bob using Alice’s prior context.

The fix: Set `dmScope` to isolate sessions per user:

```
// ~/.openclaw/openclaw.json
{
  session: {
    // Secure DM mode: isolate DM context per channel + sender.
    dmScope: "per-channel-peer",
  },
}
```

When to enable this:

- You have pairing approvals for more than one sender
- You use a DM allowlist with multiple entries
- You set `dmPolicy: "open"`
- Multiple phone numbers or accounts can message your agent

Notes:

- Default is `dmScope: "main"` for continuity (all DMs share the main session). This is fine for single-user setups.
- For multi-account inboxes on the same channel, prefer `per-account-channel-peer` .
- If the same person contacts you on multiple channels, use `session.identityLinks` to collapse their DM sessions into one canonical identity.

You can verify your DM settings with `openclaw security audit` (see security).

Gateway is the source of truth

All session state is **owned by the gateway** (the “master” OpenClaw). UI clients (macOS app, WebChat, etc.) must query the gateway for session lists and token counts instead of reading local files.

- In **remote mode**, the session store you care about lives on the remote gateway host, not your Mac.
- Token counts shown in UIs come from the gateway’s store fields (`inputTokens` , `outputTokens` , `totalTokens` , `contextTokens`). Clients do not parse JSONL transcripts to “fix up” totals.

Where state lives

- On the **gateway host**:
  - Store file: `~/.openclaw/agents/<agentId>/sessions/sessions.json` (per agent).
- Transcripts: `~/.openclaw/agents/<agentId>/sessions/<SessionId>.jsonl` (Telegram topic sessions use `.../<SessionId>-topic-<threadId>.jsonl`).
- The store is a map `sessionKey -> { sessionId, updatedAt, ... }` . Deleting entries is safe; they are recreated on demand.
- Group entries may include `displayName` , `channel` , `subject` , `room` , and `space` to label sessions in UIs.
- Session entries include `origin` metadata (label + routing hints) so UIs can explain where a session came from.
- OpenClaw does **not** read legacy Pi/Tau session folders.

Session pruning

OpenClaw trims **old tool results** from the in-memory context right before LLM calls by default. This does **not** rewrite JSONL history. See **/concepts/session-pruning**.

Pre-compaction memory flush

When a session nears auto-compaction, OpenClaw can run a **silent memory flush** turn that reminds the model to write durable notes to disk. This only runs when the workspace is writable. See **Memory** and **Compaction**.

Mapping transports → session keys

- Direct chats follow `session.dmScope` (default `main`).
  - `main` : `agent:<agentId>:<mainKey>` (continuity across devices/channels).
    - Multiple phone numbers and channels can map to the same agent main key; they act as transports into one conversation.
  - `per-peer` : `agent:<agentId>:dm:<peerId>` .
  - `per-channel-peer` : `agent:<agentId>:<channel>:dm:<peerId>` .
  - `per-account-channel-peer` : `agent:<agentId>:<channel>:<accountId>:dm:<peerId>` (accountId defaults to `default`).
  - If `session.identityLinks` matches a provider-prefixed peer id (for example `telegram:123`), the canonical key replaces `<peerId>` so the same person shares a session across channels.
- Group chats isolate state: `agent:<agentId>:<channel>:group:<id>` (rooms/channels use `agent:<agentId>:<channel>:channel:<id>`).
  - Telegram forum topics append `:topic:<threadId>` to the group id for isolation.
  - Legacy `group:<id>` keys are still recognized for migration.

Inbound contexts may still use `group:<id>` ; the channel is inferred from `Provider` and normalized to the canonical `agent:<agentId>:<channel>:group:<id>` form.

Other sources:

- Cron jobs: `cron:<job.id>`
- Webhooks: `hook:<uuid>` (unless explicitly set by the hook)
- Node runs: `node-<nodeId>`

Lifecycle

Reset policy: sessions are reused until they expire, and expiry is evaluated on the next inbound message.

Daily reset: defaults to **4:00 AM local time on the gateway host**. A session is stale once its last update is earlier than the most recent daily reset time.

Idle reset (optional): `idleMinutes` adds a sliding idle window. When both daily and idle resets are configured, **whichever expires first** forces a new session.

Legacy idle-only: if you set `session.idleMinutes` without any `session.reset` / `resetByType` config, OpenClaw stays in idle-only mode for backward compatibility.

Per-type overrides (optional): `resetByType` lets you override the policy for `direct` , `group` , and `thread` sessions (thread = Slack/Discord threads, Telegram topics, Matrix threads when provided by the connector).

Per-channel overrides (optional): `resetByChannel` overrides the reset policy for a channel (applies to all session types for that channel and takes precedence over `reset` / `resetByType`).

Reset triggers: exact `/new` or `/reset` (plus any extras in `resetTriggers`) start a fresh session id and pass the remainder of the message through. `/new <model>` accepts a model alias, `provider/model`, or provider name (fuzzy match) to set the new session model. If `/new` or `/reset` is sent alone, OpenClaw runs a short “hello” greeting turn to confirm the reset.

Manual reset: delete specific keys from the store or remove the JSONL transcript; the next message recreates them.

Isolated cron jobs always mint a fresh `sessionId` per run (no idle reuse).

Send policy (optional)

Block delivery for specific session types without listing individual ids.

```
{
  session: {
    sendPolicy: {
      rules: [
        { action: "deny", match: { channel: "discord", chatType: "group" } },
        { action: "deny", match: { keyPrefix: "cron:" } },
        // Match the raw session key (including the `agent:<id>:` prefix).
        { action: "deny", match: { rawKeyPrefix: "agent:main:discord:" } },
      ],
      default: "allow",
    },
  },
}
```

Runtime override (owner only):

- `/send on` → allow for this session
- `/send off` → deny for this session
- `/send inherit` → clear override and use config rules Send these as standalone messages so they register.

Configuration (optional rename example)

```
// ~/.openclaw/openclaw.json
{
  session: {
    scope: "per-sender", // keep group keys separate
    dmScope: "main", // DM continuity (set per-channel-peer/per-account-channel-pe
    identityLinks: {
      alice: ["telegram:123456789", "discord:987654321012345678"],
    },
    reset: {
      // Defaults: mode=daily, atHour=4 (gateway host local time).
      // If you also set idleMinutes, whichever expires first wins.
      mode: "daily",
      atHour: 4,
      idleMinutes: 120,
    },
    resetByType: {
      thread: { mode: "daily", atHour: 4 },
      direct: { mode: "idle", idleMinutes: 240 },
      group: { mode: "idle", idleMinutes: 120 },
    },
    resetByChannel: {
      discord: { mode: "idle", idleMinutes: 10080 },
    },
    resetTriggers: ["/new", "/reset"],
    store: "~/.openclaw/agents/{agentId}/sessions/sessions.json",
    mainKey: "main",
  },
}
```

Inspecting

- `openclaw status` — shows store path and recent sessions.
- `openclaw sessions --json` — dumps every entry (filter with `--active <minutes>`).

`openclaw gateway call sessions.list --params '{}'` — fetch sessions from the running gateway (use `--url` / `--token` for remote gateway access).

Send `/status` as a standalone message in chat to see whether the agent is reachable, how much of the session context is used, current thinking/verbose toggles, and when your WhatsApp web creds were last refreshed (helps spot relink needs).

Send `/context list` or `/context detail` to see what's in the system prompt and injected workspace files (and the biggest context contributors).

Send `/stop` as a standalone message to abort the current run, clear queued followups for that session, and stop any sub-agent runs spawned from it (the reply includes the stopped count).

Send `/compact` (optional instructions) as a standalone message to summarize older context and free up window space. See `/concepts/compaction`.

JSONL transcripts can be opened directly to review full turns.

Tips

Keep the primary key dedicated to 1:1 traffic; let groups keep their own keys.

When automating cleanup, delete individual keys instead of the whole store to preserve context elsewhere.

Session origin metadata

Each session entry records where it came from (best-effort) in `origin`:

- `label` : human label (resolved from conversation label + group subject/channel)
- `provider` : normalized channel id (including extensions)
- `from` / `to` : raw routing ids from the inbound envelope

- `accountId` : provider account id (when multi-account)
- `threadId` : thread/topic id when the channel supports it The origin fields are populated for direct messages, channels, and groups. If a connector only updates delivery routing (for example, to keep a DM main session fresh), it should still provide inbound context so the session keeps its explainer metadata. Extensions can do this by sending `ConversationLabel` , `GroupSubject` , `GroupChannel` , `GroupSpace` , and `SenderName` in the inbound context and calling `recordSessionMetaFromInbound` (or passing the same context to `updateLastRoute`).

‹ Bootstrapping Sessions ›