Sessions and memory

Compaction

Every model has a **context window** (max tokens it can see). Long-running chats accumulate messages and tool results; once the window is tight, OpenClaw **compacts** older history to stay within limits.

What compaction is

Compaction **summarizes older conversation** into a compact summary entry and keeps recent messages intact. The summary is stored in the session history, so future requests use:

- The compaction summary
- Recent messages after the compaction point

Compaction **persists** in the session's JSONL history.

Configuration

Use the `agents.defaults.compaction` setting in your `openclaw.json` to configure compaction behavior (mode, target tokens, etc.).

Auto-compaction (default on)

When a session nears or exceeds the model's context window, OpenClaw triggers auto-compaction and may retry the original request using the compacted context.

You'll see:

- 🧹 **`Auto-compaction complete`** in verbose mode
- **`/status`** showing 🧹 **`Compactions: <count>`**

Before compaction, OpenClaw can run a **silent memory flush** turn to store durable notes to disk. See **Memory** for details and config.

Manual compaction

Use **`/compact`** (optionally with instructions) to force a compaction pass:

```
/compact Focus on decisions and open questions
```

Context window source

Context window is model-specific. OpenClaw uses the model definition from the configured provider catalog to determine limits.

Compaction vs pruning

- **Compaction**: summarises and **persists** in JSONL.
- **Session pruning**: trims old **tool results** only, **in-memory**, per request.

See for pruning details.

Tips

- Use **`/compact`** when sessions feel stale or context is bloated.

Large tool outputs are already truncated; pruning can further reduce tool-result buildup.

If you need a fresh slate, `/new` or `/reset` starts a new session id.

‹ Memory Multi-Agent Routing ›