Messages and delivery

Messages

This page ties together how OpenClaw handles inbound messages, sessions, queueing, streaming, and reasoning visibility.

Message flow (high level)

```
Inbound message
  -> routing/bindings -> session key
  -> queue (if a run is active)
  -> agent run (streaming + tools)
  -> outbound replies (channel limits + chunking)
```

Key knobs live in configuration:

- `messages.*` for prefixes, queueing, and group behavior.
- `agents.defaults.*` for block streaming and chunking defaults.
- Channel overrides (`channels.whatsapp.*` , `channels.telegram.*` , etc.) for caps and streaming toggles.

See for full schema.

Inbound dedupe

Channels can redeliver the same message after reconnects. OpenClaw keeps a short-lived cache keyed by

channel/account/peer/session/message id so duplicate deliveries do not trigger another agent run.

Inbound debouncing

Rapid consecutive messages from the **same sender** can be batched into a single agent turn via `messages.inbound`. Debouncing is scoped per channel + conversation and uses the most recent message for reply threading/IDs.

Config (global default + per-channel overrides):

```
{
  messages: {
    inbound: {
      debounceMs: 2000,
      byChannel: {
        whatsapp: 5000,
        slack: 1500,
        discord: 1500,
      },
    },
  },
}
```

Notes:

- Debounce applies to **text-only** messages; media/attachments flush immediately.
- Control commands bypass debouncing so they remain standalone.

Sessions and devices

Sessions are owned by the gateway, not by clients.

Direct chats collapse into the agent main session key.

Groups/channels get their own session keys.

The session store and transcripts live on the gateway host.

Multiple devices/channels can map to the same session, but history is not fully synced back to every client. Recommendation: use one primary device for long conversations to avoid divergent context. The Control UI and TUI always show the gateway-backed session transcript, so they are the source of truth.

Details: Session management.

Inbound bodies and history context

OpenClaw separates the **prompt body** from the **command body**:

`Body` : prompt text sent to the agent. This may include channel envelopes and optional history wrappers.

`CommandBody` : raw user text for directive/command parsing.

`RawBody` : legacy alias for `CommandBody` (kept for compatibility).

When a channel supplies history, it uses a shared wrapper:

```
[Chat messages since your last reply - for context]
[Current message - respond to this]
```

For **non-direct chats** (groups/channels/rooms), the **current message body** is prefixed with the sender label (same style used for history entries). This keeps real-time and queued/history messages consistent in the agent prompt.

History buffers are **pending-only**: they include group messages that did *not* trigger a run (for example, mention-gated messages) and **exclude** messages already in the session transcript.

Directive stripping only applies to the **current message** section so history remains intact. Channels that wrap history should set `CommandBody` (or `RawBody`) to the original message text and keep `Body` as the combined prompt. History buffers are configurable via `messages.groupChat.historyLimit` (global default) and per-channel overrides like `channels.slack.historyLimit` or `channels.telegram.accounts.<id>.historyLimit` (set `0` to disable).

Queueing and followups

If a run is already active, inbound messages can be queued, steered into the current run, or collected for a followup turn.

- Configure via `messages.queue` (and `messages.queue.byChannel`).
- Modes: `interrupt`, `steer`, `followup`, `collect`, plus backlog variants.

Details: **Queueing**.

Streaming, chunking, and batching

Block streaming sends partial replies as the model produces text blocks. Chunking respects channel text limits and avoids splitting fenced code.

Key settings:

- `agents.defaults.blockStreamingDefault` (`on|off`, default off)
- `agents.defaults.blockStreamingBreak` (`text_end|message_end`)
- `agents.defaults.blockStreamingChunk` (`minChars|maxChars|breakPreference`)
- `agents.defaults.blockStreamingCoalesce` (idle-based batching)
- `agents.defaults.humanDelay` (human-like pause between block replies)

Channel overrides: `*.blockStreaming` and `*.blockStreamingCoalesce` (non-Telegram channels require explicit `*.blockStreaming: true`)

Details: **Streaming + chunking**.

Reasoning visibility and tokens

OpenClaw can expose or hide model reasoning:

- `/reasoning on|off|stream` controls visibility.
- Reasoning content still counts toward token usage when produced by the model.
- Telegram supports reasoning stream into the draft bubble.

Details: **Thinking + reasoning directives** and **Token use**.

Prefixes, threading, and replies

Outbound message formatting is centralized in `messages`:

- `messages.responsePrefix`, `channels.<channel>.responsePrefix`, and `channels.<channel>.accounts.<id>.responsePrefix` (outbound prefix cascade), plus `channels.whatsapp.messagePrefix` (WhatsApp inbound prefix)
- Reply threading via `replyToMode` and per-channel defaults

Details: **Configuration** and channel docs.

‹ Presence

Streaming and Chunking ›

Powered by mintlify