Media and devices

Image and Media Support

The WhatsApp channel runs via **Baileys Web**. This document captures the current media handling rules for send, gateway, and agent replies.

Goals

- Send media with optional captions via `openclaw message send --media`.
- Allow auto-replies from the web inbox to include media alongside text.
- Keep per-type limits sane and predictable.

CLI Surface

```
openclaw message send --media <path-or-url> [--message <caption>]
```

- `--media` optional; caption can be empty for media-only sends.
- `--dry-run` prints the resolved payload; `--json` emits `{ channel, to, messageId, mediaUrl, caption }`.

WhatsApp Web channel behavior

- Input: local file path **or** HTTP(S) URL.
- Flow: load into a Buffer, detect media kind, and build the correct payload:

- **Images:** resize & recompress to JPEG (max side 2048px) targeting `agents.defaults.mediaMaxMb` (default 5 MB), capped at 6 MB.
- **Audio/Voice/Video:** pass-through up to 16 MB; audio is sent as a voice note (`ptt: true`).
- **Documents:** anything else, up to 100 MB, with filename preserved when available.

WhatsApp GIF-style playback: send an MP4 with `gifPlayback: true` (CLI: `--gif-playback`) so mobile clients loop inline.

MIME detection prefers magic bytes, then headers, then file extension.

Caption comes from `--message` or `reply.text`; empty caption is allowed.

Logging: non-verbose shows ↩️ / ✅; verbose includes size and source path/URL.

Auto-Reply Pipeline

`getReplyFromConfig` returns `{ text?, mediaUrl?, mediaUrls? }`.

When media is present, the web sender resolves local paths or URLs using the same pipeline as `openclaw message send`.

Multiple media entries are sent sequentially if provided.

Inbound Media to Commands (Pi)

When inbound web messages include media, OpenClaw downloads to a temp file and exposes templating variables:

`{{MediaUrl}}` pseudo-URL for the inbound media.

`{{MediaPath}}` local temp path written before running the command.

When a per-session Docker sandbox is enabled, inbound media is copied into the sandbox workspace and `MediaPath` / `MediaUrl` are rewritten to a relative path like `media/inbound/<filename>` .

Media understanding (if configured via `tools.media.*` or shared `tools.media.models`) runs before templating and can insert `[Image]` , `[Audio]` , and `[Video]` blocks into `Body` .

- Audio sets `{{Transcript}}` and uses the transcript for command parsing so slash commands still work.
- Video and image descriptions preserve any caption text for command parsing.

By default only the first matching image/audio/video attachment is processed; set `tools.media.<cap>.attachments` to process multiple attachments.

Limits & Errors

Outbound send caps (WhatsApp web send)

- Images: ~6 MB cap after recompression.
- Audio/voice/video: 16 MB cap; documents: 100 MB cap.
- Oversize or unreadable media → clear error in logs and the reply is skipped.

Media understanding caps (transcription/description)

- Image default: 10 MB (`tools.media.image.maxBytes`).
- Audio default: 20 MB (`tools.media.audio.maxBytes`).
- Video default: 50 MB (`tools.media.video.maxBytes`).
- Oversize media skips understanding, but replies still go through with the original body.

Notes for Tests

- Cover send + reply flows for image/audio/document cases.
- Validate recompression for images (size bound) and voice-note flag for audio.
- Ensure multi-media replies fan out as sequential sends.

‹ Node Troubleshooting Audio and Voice Notes ›