Media and devices

Audio and Voice Notes

What works

Media understanding (audio): If audio understanding is enabled (or auto-detected), OpenClaw:

1. Locates the first audio attachment (local path or URL) and downloads it if needed.
2. Enforces `maxBytes` before sending to each model entry.
3. Runs the first eligible model entry in order (provider or CLI).
4. If it fails or skips (size/timeout), it tries the next entry.
5. On success, it replaces `Body` with an `[Audio]` block and sets `{{Transcript}}` .

Command parsing: When transcription succeeds, `CommandBody` / `RawBody` are set to the transcript so slash commands still work.

Verbose logging: In `--verbose` , we log when transcription runs and when it replaces the body.

Auto-detection (default)

If you **don't configure models** and `tools.media.audio.enabled` is **not** set to `false` , OpenClaw auto-detects in this order and stops at the first working option:

1. **Local CLIs** (if installed)

- `sherpa-onnx-offline` (requires `SHERPA_ONNX_MODEL_DIR` with encoder/decoder/joiner/tokens)
- `whisper-cli` (from `whisper-cpp` ; uses `WHISPER_CPP_MODEL` or the bundled tiny model)
- `whisper` (Python CLI; downloads models automatically)

2. **Gemini CLI** (`gemini`) using `read_many_files`
3. **Provider keys** (OpenAI → Groq → Deepgram → Google)

To disable auto-detection, set `tools.media.audio.enabled: false` . To customize, set `tools.media.audio.models` . Note: Binary detection is best-effort across macOS/Linux/Windows; ensure the CLI is on `PATH` (we expand `~`), or set an explicit CLI model with a full command path.

Config examples

Provider + CLI fallback (OpenAI + Whisper CLI)

```
{
  tools: {
    media: {
      audio: {
        enabled: true,
        maxBytes: 20971520,
        models: [
          { provider: "openai", model: "gpt-4o-mini-transcribe" },
          {
            type: "cli",
            command: "whisper",
            args: ["--model", "base", "{{MediaPath}}"],
            timeoutSeconds: 45,
          },
        ],
      },
    },
  },
}
```

Provider-only with scope gating

```
{
  tools: {
    media: {
      audio: {
        enabled: true,
        scope: {
          default: "allow",
          rules: [{ action: "deny", match: { chatType: "group" } }],
        },
        models: [{ provider: "openai", model: "gpt-4o-mini-transcribe" }],
      },
    },
  },
}
```

Provider-only (Deepgram)

```
{
  tools: {
    media: {
      audio: {
        enabled: true,
        models: [{ provider: "deepgram", model: "nova-3" }],
      },
    },
  },
}
```

Notes & limits

- Provider auth follows the standard model auth order (auth profiles, env vars, `models.providers.*.apiKey`).
- Deepgram picks up `DEEPGRAM_API_KEY` when `provider: "deepgram"` is used.
- Deepgram setup details: .
- Audio providers can override `baseUrl` , `headers` , and `providerOptions` via `tools.media.audio` .
- Default size cap is 20MB (`tools.media.audio.maxBytes`). Oversize audio is skipped for that model and the next entry is tried.
- Default `maxChars` for audio is **unset** (full transcript). Set `tools.media.audio.maxChars` or per-entry `maxChars` to trim output.
- OpenAI auto default is `gpt-4o-mini-transcribe` ; set `model: "gpt-4o-transcribe"` for higher accuracy.
- Use `tools.media.audio.attachments` to process multiple voice notes (`mode: "all"` + `maxAttachments`).
- Transcript is available to templates as `{{Transcript}}` .
- CLI stdout is capped (5MB); keep CLI output concise.

Mention Detection in Groups

When `requireMention: true` is set for a group chat, OpenClaw now transcribes audio **before** checking for mentions. This allows voice notes to be processed even when they contain mentions.

How it works:

1. If a voice message has no text body and the group requires mentions, OpenClaw performs a “preflight” transcription.
2. The transcript is checked for mention patterns (e.g., `@BotName` , emoji triggers).
3. If a mention is found, the message proceeds through the full reply pipeline.
4. The transcript is used for mention detection so voice notes can pass the mention gate.

Fallback behavior:

- If transcription fails during preflight (timeout, API error, etc.), the message is processed based on text-only mention detection.
- This ensures that mixed messages (text + audio) are never incorrectly dropped.

Example: A user sends a voice note saying “Hey @Claude, what’s the weather?” in a Telegram group with `requireMention: true` . The voice note is transcribed, the mention is detected, and the agent replies.

Gotchas

- Scope rules use first-match wins. `chatType` is normalized to `direct` , `group` , or `room` .
- Ensure your CLI exits 0 and prints plain text; JSON needs to be massaged via `jq -r .text` .

Keep timeouts reasonable (`timeoutSeconds` , default 60s) to avoid blocking the reply queue.

Preflight transcription only processes the **first** audio attachment for mention detection. Additional audio is processed during the main media understanding phase.

‹ Image and Media Support Camera Capture ›