Protocols and APIs

Local Models

Local is doable, but OpenClaw expects large context + strong defenses against prompt injection. Small cards truncate context and leak safety. Aim high: **≥2 maxed-out Mac Studios or equivalent GPU rig (~$30k+)**. A single **24 GB** GPU works only for lighter prompts with higher latency. Use the **largest / full-size model variant you can run**; aggressively quantized or "small" checkpoints raise prompt-injection risk (see **Security**).

Recommended: LM Studio + MiniMax M2.1 (Responses API, full-size)

Best current local stack. Load MiniMax M2.1 in LM Studio, enable the local server (default `http://127.0.0.1:1234`), and use Responses API to keep reasoning separate from final text.

```
{
  agents: {
    defaults: {
      model: { primary: "lmstudio/minimax-m2.1-gs32" },
      models: {
        "anthropic/claude-opus-4-6": { alias: "Opus" },
        "lmstudio/minimax-m2.1-gs32": { alias: "Minimax" },
      },
    },
  },
  models: {
    mode: "merge",
    providers: {
      lmstudio: {
        baseUrl: "http://127.0.0.1:1234/v1",
        apiKey: "lmstudio",
        api: "openai-responses",
        models: [
          {
            id: "minimax-m2.1-gs32",
            name: "MiniMax M2.1 GS32",
            reasoning: false,
            input: ["text"],
            cost: { input: 0, output: 0, cacheRead: 0, cacheWrite: 0 },
            contextWindow: 196608,
            maxTokens: 8192,
          },
        ],
      },
    },
  },
}
```

Setup checklist

1. Install LM Studio:

   In LM Studio, download the **largest MiniMax M2.1 build available** (avoid “small”/heavily quantized variants), start the server,

confirm `http://127.0.0.1:1234/v1/models` lists it.

Keep the model loaded; cold-load adds startup latency.

Adjust `contextWindow` / `maxTokens` if your LM Studio build differs.

For WhatsApp, stick to Responses API so only final text is sent.

Keep hosted models configured even when running local; use `models.mode: "merge"` so fallbacks stay available.

Hybrid config: hosted primary, local fallback

```
{
  agents: {
    defaults: {
      model: {
        primary: "anthropic/claude-sonnet-4-5",
        fallbacks: ["lmstudio/minimax-m2.1-gs32", "anthropic/claude-opus-4-6"],
      },
      models: {
        "anthropic/claude-sonnet-4-5": { alias: "Sonnet" },
        "lmstudio/minimax-m2.1-gs32": { alias: "MiniMax Local" },
        "anthropic/claude-opus-4-6": { alias: "Opus" },
      },
    },
  },
  models: {
    mode: "merge",
    providers: {
      lmstudio: {
        baseUrl: "http://127.0.0.1:1234/v1",
        apiKey: "lmstudio",
        api: "openai-responses",
        models: [
          {
            id: "minimax-m2.1-gs32",
            name: "MiniMax M2.1 GS32",
            reasoning: false,
            input: ["text"],
            cost: { input: 0, output: 0, cacheRead: 0, cacheWrite: 0 },
            contextWindow: 196608,
            maxTokens: 8192,
          },
        ],
      },
    },
  },
}
```

Local-first with hosted safety net

Swap the primary and fallback order; keep the same providers block and `models.mode: "merge"` so you can fall back to Sonnet or Opus when the local box is down.

Regional hosting / data routing

Hosted MiniMax/Kimi/GLM variants also exist on OpenRouter with region-pinned endpoints (e.g., US-hosted). Pick the regional variant there to keep traffic in your chosen jurisdiction while still using `models.mode: "merge"` for Anthropic/OpenAI fallbacks.

Local-only remains the strongest privacy path; hosted regional routing is the middle ground when you need provider features but want control over data flow.

Other OpenAI-compatible local proxies

vLLM, LiteLLM, OAI-proxy, or custom gateways work if they expose an OpenAI-style `/v1` endpoint. Replace the provider block above with your endpoint and model ID:

```
{
  models: {
    mode: "merge",
    providers: {
      local: {
        baseUrl: "http://127.0.0.1:8000/v1",
        apiKey: "sk-local",
        api: "openai-responses",
        models: [
          {
            id: "my-local-model",
            name: "Local Model",
            reasoning: false,
            input: ["text"],
            cost: { input: 0, output: 0, cacheRead: 0, cacheWrite: 0 },
            contextWindow: 120000,
            maxTokens: 8192,
          },
        ],
      },
    },
  },
}
```

Keep `models.mode: "merge"` so hosted models stay available as fallbacks.

Troubleshooting

Gateway can reach the proxy? `curl http://127.0.0.1:1234/v1/models` .

LM Studio model unloaded? Reload; cold start is a common “hanging” cause.

Context errors? Lower `contextWindow` or raise your server limit.

Safety: local models skip provider-side filters; keep agents narrow and compaction on to limit prompt injection blast radius.

‹ CLI Backends Network model ›