Protocols and APIs

OpenAI Chat Completions

OpenClaw's Gateway can serve a small OpenAI-compatible Chat Completions endpoint.

This endpoint is **disabled by default**. Enable it in config first.

```
POST /v1/chat/completions
```

Same port as the Gateway (WS + HTTP multiplex): `http://<gateway-host>:<port>/v1/chat/completions`

Under the hood, requests are executed as a normal Gateway agent run (same codepath as `openclaw agent`), so routing/permissions/config match your Gateway.

Authentication

Uses the Gateway auth configuration. Send a bearer token:

```
Authorization: Bearer <token>
```

Notes:

- When `gateway.auth.mode="token"`, use `gateway.auth.token` (or `OPENCLAW_GATEWAY_TOKEN`).
- When `gateway.auth.mode="password"`, use `gateway.auth.password` (or `OPENCLAW_GATEWAY_PASSWORD`).

If `gateway.auth.rateLimit` is configured and too many auth failures occur, the endpoint returns `429` with `Retry-After`.

Choosing an agent

No custom headers required: encode the agent id in the OpenAI `model` field:

- `model: "openclaw:<agentId>"` (example: `"openclaw:main"`, `"openclaw:beta"`)
- `model: "agent:<agentId>"` (alias)

Or target a specific OpenClaw agent by header:

- `x-openclaw-agent-id: <agentId>` (default: `main`)

Advanced:

- `x-openclaw-session-key: <sessionKey>` to fully control session routing.

Enabling the endpoint

Set `gateway.http.endpoints.chatCompletions.enabled` to `true`:

```
{
  gateway: {
    http: {
      endpoints: {
        chatCompletions: { enabled: true },
      },
    },
  },
}
```

Disabling the endpoint

Set `gateway.http.endpoints.chatCompletions.enabled` to `false`:

```
{
  gateway: {
    http: {
      endpoints: {
        chatCompletions: { enabled: false },
      },
    },
  },
}
```

Session behavior

By default the endpoint is **stateless per request** (a new session key is generated each call).

If the request includes an OpenAI `user` string, the Gateway derives a stable session key from it, so repeated calls can share an agent session.

Streaming (SSE)

Set `stream: true` to receive Server-Sent Events (SSE):

- `Content-Type: text/event-stream`
- Each event line is `data: <json>`
- Stream ends with `data: [DONE]`

Examples

Non-streaming:

```
curl -sS http://127.0.0.1:18789/v1/chat/completions \
  -H 'Authorization: Bearer YOUR_TOKEN' \
  -H 'Content-Type: application/json' \
  -H 'x-openclaw-agent-id: main' \
  -d '{
    "model": "openclaw",
    "messages": [{"role":"user","content":"hi"}]
  }'
```

Streaming:

```
curl -N http://127.0.0.1:18789/v1/chat/completions \
  -H 'Authorization: Bearer YOUR_TOKEN' \
  -H 'Content-Type: application/json' \
  -H 'x-openclaw-agent-id: main' \
  -d '{
    "model": "openclaw",
    "stream": true,
    "messages": [{"role":"user","content":"hi"}]
  }'
```

‹ Bridge Protocol Tools Invoke API ›