Built-in tools

Web Tools

OpenClaw ships two lightweight web tools:

- `web_search` — Search the web via Brave Search API (default) or Perplexity Sonar (direct or via OpenRouter).
- `web_fetch` — HTTP fetch + readable extraction (HTML → markdown/text).

These are **not** browser automation. For JS-heavy sites or logins, use the **Browser tool**.

How it works

- `web_search` calls your configured provider and returns results.
  - **Brave** (default): returns structured results (title, URL, snippet).
  - **Perplexity**: returns AI-synthesized answers with citations from real-time web search.

  Results are cached by query for 15 minutes (configurable).
- `web_fetch` does a plain HTTP GET and extracts readable content (HTML → markdown/text). It does **not** execute JavaScript.
- `web_fetch` is enabled by default (unless explicitly disabled).

Choosing a search provider

Provider	Pros	Cons	API Key
Brave (default)	Fast, structured results, free tier	Traditional search results	`BRAVE_API_KEY`
Perplexity	AI-synthesized answers, citations, real-time	Requires Perplexity or OpenRouter access	`OPENROUTER_API_KEY` or `PERPLEXITY_API_KEY`

See **Brave Search setup** and **Perplexity Sonar** for provider-specific details.

Set the provider in config:

```
{
  tools: {
    web: {
      search: {
        provider: "brave", // or "perplexity"
      },
    },
  },
}
```

Example: switch to Perplexity Sonar (direct API):

```
{
  tools: {
    web: {
      search: {
        provider: "perplexity",
        perplexity: {
          apiKey: "pplx-...",
          baseUrl: "https://api.perplexity.ai",
          model: "perplexity/sonar-pro",
        },
      },
    },
  },
}
```

Getting a Brave API key

1. Create a Brave Search API account at
2. In the dashboard, choose the **Data for Search** plan (not “Data for AI”) and generate an API key.
3. Run `openclaw configure --section web` to store the key in config (recommended), or set `BRAVE_API_KEY` in your environment.

Brave provides a free tier plus paid plans; check the Brave API portal for the current limits and pricing.

Where to set the key (recommended)

Recommended: run `openclaw configure --section web` . It stores the key in `~/.openclaw/openclaw.json` under `tools.web.search.apiKey` .

Environment alternative: set `BRAVE_API_KEY` in the Gateway process environment. For a gateway install, put it in `~/.openclaw/.env` (or your service environment). See .

Using Perplexity (direct or via OpenRouter)

Perplexity Sonar models have built-in web search capabilities and return AI-synthesized answers with citations. You can use them via OpenRouter (no credit card required - supports crypto/prepaid).

Getting an OpenRouter API key

1. Create an account at https://openrouter.ai/
2. Add credits (supports crypto, prepaid, or credit card)
3. Generate an API key in your account settings

Setting up Perplexity search

```
{
  tools: {
    web: {
      search: {
        enabled: true,
        provider: "perplexity",
        perplexity: {
          // API key (optional if OPENROUTER_API_KEY or PERPLEXITY_API_KEY is set)
          apiKey: "sk-or-v1-...",
          // Base URL (key-aware default if omitted)
          baseUrl: "https://openrouter.ai/api/v1",
          // Model (defaults to perplexity/sonar-pro)
          model: "perplexity/sonar-pro",
        },
      },
    },
  },
}
```

Environment alternative: set `OPENROUTER_API_KEY` or `PERPLEXITY_API_KEY` in the Gateway environment. For a gateway install, put it in

`~/.openclaw/.env` .

If no base URL is set, OpenClaw chooses a default based on the API key source: ›

- `PERPLEXITY_API_KEY` or `pplx-...` → `https://api.perplexity.ai`
- `OPENROUTER_API_KEY` or `sk-or-...` → `https://openrouter.ai/api/v1`
- Unknown key formats → OpenRouter (safe fallback)

Available Perplexity models

Model	Description	Best for
`perplexity/sonar`	Fast Q&A with web search	Quick lookups
`perplexity/sonar-pro` (default)	Multi-step reasoning with web search	Complex questions
`perplexity/sonar-reasoning-pro`	Chain-of-thought analysis	Deep research

web_search

Search the web using your configured provider.

Requirements

- `tools.web.search.enabled` must not be `false` (default: enabled)
- API key for your chosen provider:
  - **Brave**: `BRAVE_API_KEY` or `tools.web.search.apiKey`
  - **Perplexity**: `OPENROUTER_API_KEY` , `PERPLEXITY_API_KEY` , or `tools.web.search.perplexity.apiKey`

Config

```
{
  tools: {
    web: {
      search: {
        enabled: true,
        apiKey: "BRAVE_API_KEY_HERE", // optional if BRAVE_API_KEY is set
        maxResults: 5,
        timeoutSeconds: 30,
        cacheTtlMinutes: 15,
      },
    },
  },
}
```

Tool parameters

`query` (required)

`count` (1–10; default from config)

`country` (optional): 2-letter country code for region-specific results (e.g., “DE”, “US”, “ALL”). If omitted, Brave chooses its default region.

`search_lang` (optional): ISO language code for search results (e.g., “de”, “en”, “fr”)

`ui_lang` (optional): ISO language code for UI elements

`freshness` (optional): filter by discovery time

- Brave: `pd`, `pw`, `pm`, `py`, or `YYYY-MM-DDtoYYYY-MM-DD`
- Perplexity: `pd`, `pw`, `pm`, `py`

Examples:

```
// German-specific search
await web_search({
  query: "TV online schauen",
  count: 10,
  country: "DE",
  search_lang: "de",
});

// French search with French UI
await web_search({
  query: "actualités",
  country: "FR",
  search_lang: "fr",
  ui_lang: "fr",
});

// Recent results (past week)
await web_search({
  query: "TMBG interview",
  freshness: "pw",
});
```

web_fetch

Fetch a URL and extract readable content.

web_fetch requirements

`tools.web.fetch.enabled` must not be `false` (default: enabled)

Optional Firecrawl fallback: set `tools.web.fetch.firecrawl.apiKey` or `FIRECRAWL_API_KEY`.

web_fetch config

```
{
  tools: {
    web: {
      fetch: {
        enabled: true,
        maxChars: 50000,
        maxCharsCap: 50000,
        maxResponseBytes: 2000000,
        timeoutSeconds: 30,
        cacheTtlMinutes: 15,
        maxRedirects: 3,
        userAgent: "Mozilla/5.0 (Macintosh; Intel Mac OS X 14_7_2) AppleWebKit/53
        readability: true,
        firecrawl: {
          enabled: true,
          apiKey: "FIRECRAWL_API_KEY_HERE", // optional if FIRECRAWL_API_KEY is s
          baseUrl: "https://api.firecrawl.dev",
          onlyMainContent: true,
          maxAgeMs: 86400000, // ms (1 day)
          timeoutSeconds: 60,
        },
      },
    },
  },
}
```

web_fetch tool parameters

- `url` (required, http/https only)
- `extractMode` (`markdown` | `text`)
- `maxChars` (truncate long pages)

Notes:

- `web_fetch` uses Readability (main-content extraction) first, then Firecrawl (if configured). If both fail, the tool returns an error.

Firecrawl requests use bot-circumvention mode and cache results by default.

`web_fetch` sends a Chrome-like User-Agent and `Accept-Language` by default; override `userAgent` if needed.

`web_fetch` blocks private/internal hostnames and re-checks redirects (limit with `maxRedirects`).

`maxChars` is clamped to `tools.web.fetch.maxCharsCap`.

`web_fetch` caps the downloaded response body size to `tools.web.fetch.maxResponseBytes` before parsing; oversized responses are truncated and include a warning.

`web_fetch` is best-effort extraction; some sites will need the browser tool.

See **Firecrawl** for key setup and service details.

Responses are cached (default 15 minutes) to reduce repeated fetches.

If you use tool profiles/allowlists, add `web_search` / `web_fetch` or `group:web`.

If the Brave key is missing, `web_search` returns a short setup hint with a docs link.

‹ Exec Tool apply_patch Tool ›