eCFR-MCP — Model Context Protocol Server for U.S. Federal Regulations

Overview

eCFR-MCP is a Model Context Protocol (MCP) server that allows AI systems (such as Claude Desktop, Cursor, and other MCP-compatible tools) to query the official **Electronic Code of Federal Regulations (eCFR) API** to:

- Search federal regulations
- Retrieve XML snapshots of CFR titles/parts/sections
- List CFR titles and agencies
- Provide explainable, citation-based responses

Instead of relying on hallucinated legal answers, an LLM can use this MCP server to **ground its reasoning in authoritative regulatory text**.

This is especially valuable for:

- Healthcare, pharmacy, life sciences, medical compliance
- Legal research / regulatory analysis
- Government technology & policy engineering
- Enterprise AI governance workflows

> **Note:** This project does *not* interpret law — it provides machine-accessible access to the CFR text. Always consult qualified professionals for legal or regulatory decisions.

Why This Matters

Modern LLMs are powerful but **unreliable without external knowledge grounding**. Regulations are particularly sensitive — incorrect answers can result in:

- compliance violations
- billing or licensing risk
- legal exposure
- operational failures

This project addresses that risk by:

- giving AI tools **structured access** to real regulatory data
- enabling **transparent citations** and **verifiable answers**
- supporting research workflows that require **traceability**

It transforms LLMs from *guessing about regulations* into *querying official data sources.*

Provided MCP Tools

Once installed, the MCP server exposes these tools:

Tool	Description
`ecfr_search_results`	Full-text regulation search (JSON results)
`ecfr_search_with_date_range`	Search with optional title + date window filters (start/end)
`ecfr_search_summary`	Aggregated search summaries / counts
`ecfr_list_titles`	Lists all CFR titles and metadata
`ecfr_list_agencies`	Lists agencies and CFR authority assignments
`ecfr_get_title_xml`	Retrieves full XML snapshot of a CFR title (with optional part/section hints)
`ecfr_get_part_xml`	Retrieves XML for a specific part to avoid large title payloads
`ecfr_get_title_structure`	Returns title hierarchy (JSON by default, XML optional)
`ecfr_get_title_versions`	Lists issue dates/versions for a title (filters for issue_date, part, section)
`ecfr_get_title_ancestry`	Retrieves the full title hierarchy for a given date
`ecfr_get_section_content`	Retrieves section content by structure index/section (XML fallback for 404s)
`ecfr_compare_title_dates`	Compares structures between two dates and reports added/removed/modified sections
`ecfr_get_recent_changes`	Convenience wrapper to compare over a trailing window (e.g., last 180 days)

These tools support workflows where the LLM can:

1. Search the regulation corpus
2. Identify candidate sections
3. Retrieve authoritative text
4. Interpret and explain it
5. Provide citations

Installation & Setup

Prerequisites

You'll need the following before installing the MCP server:

- Node.js **18+** (LTS recommended)
- Git (optional but recommended for cloning)

Below are platform-specific setup steps.

Install prerequisites on Windows

1. **Install Node.js**

   - Download from https://nodejs.org (choose *LTS*)
   - Run installer → keep default options → ensure **"Add to PATH"** remains checked

   Verify installation:

   ```
   node --version
   npm --version
   ```

2. **Install Git (optional but recommended)**

   Download from https://git-scm.com/download/win

   After install, verify:

   ```
   git --version
   ```

   If `node` is not found, restart the terminal or sign out/in so PATH updates.

Install prerequisites on macOS

1. **Install Homebrew (recommended)**

   ```
   /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/inst
   ```

2. **Install Node.js and Git**

   ```
   brew install node git
   ```

   Verify:

   ```
   node --version
   npm --version
   git --version
   ```

   macOS System node (from Xcode tools) may be outdated — prefer Homebrew.

Install prerequisites on Linux   Ubuntu / Debian:

```
sudo apt update
sudo apt install -y nodejs npm git
```

(If `node --version` shows < 18, install Node LTS from NodeSource: https://github.com/nodesource/distributions)

Fedora / RHEL:

```
sudo dnf install -y nodejs npm git
```

Verify:

```
node --version
npm --version
git --version
```

Once prerequisites are installed, continue with the platform setup below.

Do I need to run the server manually?

In most MCP-aware apps (like **Claude Desktop** or some editor integrations), you **do not need to run** `node build/index.js` yourself once everything is built.

- The client (e.g., Claude Desktop) will automatically start the MCP server using the command you configure in its settings.
- Manually running `node build/index.js` is mainly useful for **sanity checks** or **debugging**.

So in typical usage:

1. Install prerequisites (Node, Git)
2. Clone this repo
3. Run `npm install` and `npm run build`
4. Configure your MCP client (Claude / Cursor / etc.) to use `node /path/to/build/index.js`

After that, let the client manage the MCP process.

Windows Installation

```
git clone https://github.com/sidneyanderson/ecfr-mcp
cd ecfr-mcp
npm install
npm run build
```

Optional sanity test (manual run):

```
node build/index.js
```

> You don't need to run this for normal Claude/Cursor use — it's just to confirm the server starts without errors.

---

macOS Installation

```
git clone https://github.com/sidneyanderson/ecfr-mcp
cd ecfr-mcp
npm install
npm run build
chmod +x build/index.js
```

(Optional direct execution)

```
./build/index.js
```

Linux Installation

```
git clone https://github.com/sidneyanderson/ecfr-mcp
cd ecfr-mcp
npm install
npm run build
chmod +x build/index.js
```

Optional sanity test (manual run):

```
node build/index.js
```

> You don't need to run this for normal Claude/Cursor use — it's just
> to confirm the server starts without errors.

Using with Claude Desktop (Recommended)

Open:

> Settings → Developer → **Edit Config**

Claude expects **valid JSON** — that means:

- All paths must be quoted strings
- On **Windows**, backslashes must be **escaped** in JSON (\\ → becomes \)
- Or simply use **forward slashes**, which Windows also accepts (e.g., `C:/ecfr-mcp/build/index.js`)

Windows example (Node on PATH)

If your project is at `C:\ecfr-mcp` and `node` is on your PATH:

```
{
  "mcpServers": {
    "ecfr-mcp": {
      "command": "node",
```

```json
      "args": [
        "C:\\ecfr-mcp\\build\\index.js"
      ]
    }
  }
}
```

Inside JSON, each \\ becomes \ in the real Windows path.

Windows example (simpler — forward-slash path)

```json
{
  "mcpServers": {
    "ecfr-mcp": {
      "command": "node",
      "args": [
        "C:/ecfr-mcp/build/index.js"
      ]
    }
  }
}
```

macOS / Linux example

Use forward slashes — no escaping required:

```json
{
  "mcpServers": {
    "ecfr-mcp": {
      "command": "node",
      "args": [
        "/Users/yourname/ecfr-mcp/build/index.js"
      ]
    }
  }
}
```

Linux variant:

```json
{
  "mcpServers": {
    "ecfr-mcp": {
      "command": "node",
      "args": [
        "/home/yourname/ecfr-mcp/build/index.js"
      ]
    }
```

```
    }
}
```

After editing, **restart Claude Desktop** so it reloads the MCP configuration. Make sure the Claude system tray instance is also closed. If you are unsure, reboot.

Test Prompt

> Using the eCFR tools, fetch and explain 21 CFR 1306.04 with citations.

You should see tool execution in the side panel (calls to `ecfr_get_title_xml`, `ecfr_search_results`, etc.).

Using with Cursor / VS Code MCP clients

Add to your MCP configuration:

```
"ecfr-mcp": {
  "command": "node",
  "args": [
    "/path/to/ecfr-mcp/build/index.js"
  ]
}
```

Restart the client.

CLI Testing (No UI)

List tools:

```
npx -y @modelcontextprotocol/cli call node ./build/index.js tools
```

Call a tool manually:

```
npx -y @modelcontextprotocol/cli call node ./build/index.js call ecfr_list_titles
```

Helper Scripts

Windows startup script

Create `run-ecfr.ps1`:

```
cd C:/ecfr-mcp
node build/index.js
```

macOS / Linux startup script

Create `run-ecfr.sh`:

```bash
#!/bin/bash
cd ~/ecfr-mcp
node build/index.js
```

Then:

```bash
chmod +x run-ecfr.sh
./run-ecfr.sh
```

Troubleshooting

MCP tools don't appear in Claude

- Restart Claude Desktop
- Verify the path in config
- Check Developer → **Open Logs Folder**
- Ensure `npm run build` was executed

Node quits immediately

Run manually to see the error:

```
node build/index.js
```

Copy the stack trace into a GitHub issue.

Project Structure

```
ecfr-mcp/
   src/            TypeScript source
   build/          Compiled MCP server
   scripts/        Optional helper utilities
   package.json
   tsconfig.json
   README.md
```

Support & Contributions

Issues and PRs are welcome!

- File bugs, questions, or feature requests here:
  https://github.com/sidneyanderson/ecfr-mcp/issues

Disclaimer

This project provides **technical access** to regulatory data.
It does **not** constitute legal guidance or regulatory interpretation.

Always consult regulatory professionals for formal decisions.

License & Usage Restrictions

This project is proprietary and is licensed under a restricted-use license.

- Redistribution, modification, or public hosting is not permitted

- No AI training, dataset creation, model tuning, or ingestion

- No commercial or production use without a paid license

- No benchmarking, reverse-engineering, or competitive analysis

- No sharing outside your organization
- Internal use only, by the original authorized recipient
- Attribution to "Software created and owned by Sidney Anderson" must remain

Violations automatically terminate the right to use the software.

Full terms are available in the LICENSE file.