info@jana.earth

Jana MCP Server — User Guide

Version: 1.0.0
Last updated: 2026-04-03

The **Jana MCP Server** exposes the Jana environmental data platform through the Model Context Protocol (MCP). Compatible assistants (Cursor, Claude Code, ChatGPT, and other MCP clients) can **list** and **invoke** tools that query air quality, emissions, analytics, exports, GLEIF entities, Global Carbon Project (GCP) data, NOAA storm events, and platform metadata.

This guide explains **how the server behaves**, **how authentication works**, and **what each of the 35 tools does**, including common parameters. For **installing and configuring** the server in a specific client, use the setup guides linked below.

Related documentation

- REST API reference: jana_openapi.yaml and customer_data_dictionary.md (field-level detail for many underlying endpoints).

Table of contents

What you need

- A **Jana account** with API access (your organization provisions credentials).
- An **MCP-capable client** configured to connect to a running Jana MCP server instance (URL and transport as provided by your administrator — typically HTTP with SSE).
- Optional: a **pre-issued JWT** (`JANA_TOKEN`) for headless or CI use, if your deployment supports it (see Authentication).

The server implementation lives in the **`jana-mcp-server`** codebase; it proxies requests to the **Jana backend API** (`JANA_BACKEND_URL`).

Client setup (install & configuration)

Client	Guide
Cursor	cursor-setup-guide.md
Claude Code	claude-code-setup-guide.md
ChatGPT	chatgpt_setup_guide.md

How it works

1. The client connects to the MCP server and calls **`list_tools`** to discover available tools (names, descriptions, and JSON Schemas for arguments).
2. The client calls **`call_tool`** with a tool name and a JSON object of arguments.
3. The server validates arguments where applicable, calls the Jana backend, and returns **text** content (typically JSON serialized as a string) describing the result or an error.

Architecture (conceptual)

```
MCP client (Cursor / Claude / ChatGPT / …)
        → Jana MCP Server (FastAPI, MCP over HTTP/SSE)
        → Jana backend API (Django / DRF)
```

Authentication

Most data tools require an authenticated session to the Jana backend.

Device code flow (interactive)

When a tool runs and the user is not yet authorized, the server may return a structured **`AUTH_PENDING`** payload (instead of failing silently). The assistant should show you:

- **`verification_uri`** — open in a browser
- **`user_code`** — enter at that URL
- **`seconds_remaining`** — time left to complete the step

After you approve access, retry the tool call.

Pre-configured JWT (automation)

Deployments can set **`JANA_TOKEN`** (a valid JWT) so the server skips interactive device authorization. This is intended for **trusted automation** (for example CI), not for sharing broadly.

OAuth 2.1 (hosted MCP)

Hosted configurations may enable **OAuth 2.1** for MCP clients (for example Claude Code). Your administrator configures **`mcp_issuer_url`**, **`mcp_auth_backend_url`**, and related settings on the server.

Common query patterns

Many environmental data tools share the same **location** and **time** concepts.

Location (choose at least one where required)

Parameter	Meaning
`location_bbox`	`[min_lon, min_lat, max_lon, max_lat]`
`location_point` + **`radius_km`**	Center `[longitude, latitude]` and radius in km (0.1–500). **Both** point and radius are required when using a radius search.
`country_codes`	Array of **ISO-3166-1 alpha-3** codes (for example `USA`, `GBR`, `CHN`).

Tools that **require** a location filter include: `get_air_quality`, `get_emissions`, `get_unified_data`, `get_aggregations`, `get_quality`, `get_correlations`, `get_analytics`, and `request_export`.

Exception: `find_nearby` always requires **`location_point`** and **`radius_km`**.

Time

- **`date_from`**, **`date_to`** — ISO 8601 **dates** (for example `2024-01-01`).

Data sources

Where a tool accepts **`sources`**, values are typically a subset of: **`openaq`**, **`climatetrace`**, **`edgar`**.

Limits

- Many list-style tools accept **`limit`** (often 1–1000, with a default around 100).

GLEIF / LEI

- **`lei`** — 20-character Legal Entity Identifier (for example `HWUPKR0MPOU8FGXBT394`).
- Country filters in GLEIF search often use **ISO-2** (for example `US`, `DE`).

Tool reference (35 tools)

Tools are grouped by function. **Required** JSON arguments are noted; all other properties are optional unless your backend enforces additional rules.

Core environmental data

Tool	Summary
`get_air_quality`	OpenAQ measurements (PM2.5, PM10, O3, NO2, SO2, CO) with location and time. **Requires** at least one of: `location_bbox`, (`location_point` +

Tool	Summary
	`radius_km`), or `country_codes`. Optional: `parameters` (enum `pm25`, `pm10`, `o3`, `no2`, `so2`, `co`), `date_from`, `date_to`, `limit`.
`get_emissions`	GHG emissions from Climate TRACE and EDGAR (facility/national context, sector and gas fields per API). **Requires** the same location pattern as air quality.
`get_unified_data`	Single request across OpenAQ, Climate TRACE, and EDGAR: integrated environmental rows with optional **`temporal_resolution`** (`raw`, `hourly`, `daily`, `monthly`), **`quality_min`** (0–1), **`format`** (`json`, `geojson`, `csv`), **`sources`**, **`parameters`**, dates, location filters, **`limit`**. **Requires** a location filter.
`get_aggregations`	Pre-computed hourly/daily/monthly rollups (faster than raw aggregation). **Requires** a location filter. Notable arguments: **`temporal_resolution`** (`hourly`, `daily`, `monthly`), **`sources`**, **`parameters`**, dates, **`limit`**.
`find_nearby`	Stations and emission-related facilities near a point with distances. **Required:** `location_point`, `radius_km`. Optional: `sources`, `limit`.

Analytics and quality

Tool	Summary
`get_trends`	Backend-computed time-series trends. **Required:** `source` (`openaq` | `climatetrace` | `edgar`), `parameter`. Optional: `date_from`, `date_to`, **`temporal_resolution`** (`daily`, `weekly`, `monthly`).
`get_correlations`	Correlations between parameters / space / time / sectors. **Requires** a location filter. Key arguments: **`correlation_type`** (`spatial`, `temporal`, `parameter`, `sector`), **`method`** (`pearson`, `spearman`), **`lag_hours`** (0–720), `sources`, `parameters`, dates.
`get_analytics`	Anomaly detection, pattern recognition, statistical summaries. **Requires** a location filter. **`analysis_type:`** `anomaly_detection`, `pattern_recognition`, `statistical_summary`.
`get_quality`	Data-quality scores, completeness, flags, and recommendations. **Requires** a location filter.

Async export workflow

Tool	Summary
`request_export`	Starts a background export (CSV or JSON). **Requires** a location filter. Arguments mirror query filters: `sources`, `country_codes`, `parameters`, bbox/point/radius, dates, **`export_format`** (`csv` | `json`), **`email_on_completion`**. Returns a **`job_id`**.
`check_export_status`	**Required:** `job_id`. Returns status (`queued`, `running`, `completed`, `failed`), progress, and download URL when ready.

Tool	Summary
`download_export`	**Required:** `job_id`. Returns a **time-limited signed URL** (on the order of one hour) for a **completed** job.

Metadata and discovery

Tool	Summary
`get_parameters`	Parameters available per source (names, units, descriptions). Optional filter: `sources`.
`get_sectors`	Emission sectors available for Climate TRACE / EDGAR for given countries. **Required:** `country_codes`. Optional: `sources` (`climatetrace`, `edgar`), `date_from`, `date_to`, `limit`.
`get_definitions`	Overview of definition categories exposed by the API.
`get_unit_definitions`	Units, descriptions, types, conversion factors.
`get_source_metadata`	Provenance for OpenAQ, Climate TRACE, EDGAR (coverage, refresh, attribution).

System and platform statistics

Tool	Summary
`get_data_summary`	High-level coverage, counts by source, geography, freshness. No arguments.
`get_system_health`	Availability and connectivity signals. No arguments.
`get_statistics`	Platform-wide table list with counts and freshness; links to per-table stats. No arguments.
`get_table_statistics`	**Required:** `table_name` (for example `openaq_measurements`, `climatetrace_emissions`).
`get_openaq_stats`	OpenAQ-specific metrics (locations, measurements, parameters). No arguments.
`get_climatetrace_stats`	Climate TRACE–specific metrics (assets, emissions, sectors). No arguments.

Global Carbon Project (GCP)

Tool	Summary
`get_gcp_national_emissions`	National CO2 (territorial/consumption, per-capita). Filters: `country_code` (ISO-3), `year`, `budget_version`.
`get_gcp_emissions_by_fuel`	Global fossil CO2 by fuel (coal, oil, gas, cement, flaring, …). Filters: `year`, `budget_version`.
`get_gcp_carbon_budget`	Global carbon budget components by year. Filters: `year`, `budget_version`.

Tool	Summary
`get_gcp_methane_budget`	Global methane budget components by year. Filters: `year`, `budget_version`.

NOAA Storm Events (United States)

Tool	Summary
`get_noaa_storm_events`	Severe weather records (type, state, county, time, damage, injuries/fatalities, coordinates). Optional filters: **`event_type`**, **`state`** (uppercase full name, e.g. `TEXAS`), **`year`** (2015–2024), **`month`**, **`bbox`**, **`lat`** + **`lon`** + **`radius_km`**, **`limit`**.

GLEIF and company–asset linkage

Tool	Summary
`search_gleif_entities`	Search/list legal entities: `search`, `entity_status`, `registration_status`, `entity_category`, `legal_address_country`, `headquarters_country`, `jurisdiction`, `ordering`, `limit`, `offset`.
`get_gleif_entity`	**Required:** `lei`. Full entity record.
`get_gleif_entity_parents`	**Required:** `lei`. Direct and ultimate parents.
`get_gleif_entity_children`	**Required:** `lei`. Direct subsidiaries.
`get_gleif_relationships`	Relationship graph filters: `start_lei`, `end_lei`, `relationship_type`, `relationship_status`, `ordering`, `limit`, `offset`.
`get_gleif_exceptions`	Reporting exceptions: optional `lei`, `exception_category`, `exception_reason`, `limit`, `offset`.
`get_company_asset_matches`	Links GLEIF LEIs to Climate TRACE assets (confidence, methods). Filters: `legal_entity_lei`, `company_id`, `matching_method`, `relationship_type`, `verified`, `search`, `ordering`, `limit`, `offset`.

Responses and errors

- Successful tool calls return **text** content; in practice the payload is **JSON** (pretty-printed or compact depending on server settings).
- **`AUTH_PENDING`** — see Authentication; complete browser authorization and retry.
- **`VALIDATION_ERROR`** — missing required arguments or invalid bbox/dates/coordinates.
- **`API_ERROR`** / **`NETWORK_ERROR`** — backend or connectivity failure.
- **`UNKNOWN_TOOL`** — client requested a tool name not registered on the server.

For HTTP-level API behavior (status codes, pagination, and response envelopes), align with the **OpenAPI** document and data dictionary cited above.

---

*This guide is derived from the tool definitions and dispatch table in the `jana-mcp-server` repository (`src/jana_mcp/tools/`, `src/jana_mcp/server.py`). Tool count: **35**.*
