MCP SERVER | NO CODE | CLOUD HOSTED

Data Sorting & Filtering Engine MCP for AI Agents

Structure, sort, and deduplicate massive JSON arrays reliably for data analysis.

This Data Sorting & Filtering Engine lets your AI client reliably process massive JSON datasets that overwhelm standard LLMs. It handles array sorting and deduplication using native JavaScript performance, ensuring data integrity even with thousands of records. Stop losing context or misordering large lists; get deterministic results every time.

A+ Quality Score 100/100

data-processing | array-manipulation | json-sorting | data-deduplication | performance-optimization | data-integrity

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/data-sorting-filtering-engine

SCAN TO VIEW

The connectivity layer between AI and the world's software.

Vinkius sits between AI and every application. All communication passes through Vinkius Cloud via the Model Context Protocol (MCP) — with governance, observability, and security at every layer.

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the connectivity layer where AI connects to the software your business already runs. We handle the hosting, the security, the credentials, the uptime — you get agents that actually do things.

We operate the world's largest managed MCP catalog. Major SaaS platforms, CRMs, databases, and cloud providers — running, monitored, production-ready. This MCP server is hosted and maintained by the Vinkius Cloud for AI Agents.

> *The agent doesn't manage credentials, doesn't manage uptime, doesn't manage security. Vinkius does.*
>
> — Architecture principle

Four Pillars of the Vinkius Runtime

01 — Security by design

Credentials stay encrypted at rest via AES-256. The AI agent never touches raw keys — they're injected into a sandboxed V8 isolate at runtime. Actions are logged, and connections have an emergency kill switch.

02 — Built on MCP Fusion

This MCP server was built with **MCP Fusion**, the open-source framework (Apache 2.0) that powers the entire Vinkius catalog. Schema-as-firewall strips undeclared fields, compiled PII redaction runs at zero overhead, and cryptographic lockfiles produce git-diffable audit trails.

03 — Deterministic observability

Eight immutable metrics per endpoint: request volume, p95 latency, error rate, active connections, cost attribution. A live payload feed logs every tool call with mutation detection.

04 — Autonomous operations

Servers are deployed, monitored, and patched autonomously. New capabilities and security patches ship weekly. Zero-downtime deployments ensure continuous availability across all managed MCP servers.

AES-256	Ed25519	24h TTL	V8 Isolate
Encryption at rest	PKI vault signatures	Ephemeral session keys	Sandboxed execution

One Token. Instant Access.

Every MCP server on Vinkius is accessed through a **Connection Token**. Tokens are generated in the cloud dashboard and produce a unique MCP endpoint URL. Paste this URL into any MCP-compatible client — no SDK required.

A single token can serve **multiple AI clients simultaneously**, or you can issue separate tokens per client for granular access control. Each token tracks its own request count, last activity timestamp, and can be individually enabled or revoked.

MCP ENDPOINT

```
https://edge.vinkius.com/{token}/mcp
```

Claude · Cursor · VS Code · Windsurf · Grok · Gemini

Security Is the Architecture

Security in Vinkius is not a feature — it's the foundation of the runtime. The gateway enforces multiple independent protection layers between AI agents and third-party APIs.

01 — Ed25519 PKI Vault

Every workspace has an Ed25519 Master Key. Session keys are generated ephemerally (24h TTL) and signed by the Master Key. Credentials never leave the vault boundary.

02 — V8 Isolate Sandboxing

Tool code runs inside isolated-vm V8 isolates with 64 MB memory caps and per-request timeouts. No filesystem access, no network access except through the SSRF-guarded fetch bridge.

03 — SSRF Guard

All outbound HTTP requests are DNS-resolved and validated before execution. Private IP ranges (10.x, 172.16-31.x, 192.168.x, AWS metadata 169.254.x) are blocked at the network layer.

04 — DLP & PII Redaction

A ResponseGuard pipeline intercepts every tool response. Configurable redaction patterns strip sensitive fields (emails, SSNs, card numbers) before data reaches the AI agent.

05 — Cryptographic Audit Trail

Every request is signed into a SHA-256 hash chain with Ed25519 signatures. Events form a tamper-proof, SIEM-exportable forensic record.

06 — Honeytoken Trap System

Phantom credentials are injected into isolated environments. If a honeytoken is used outside Vinkius infrastructure, the server is quarantined instantly.

Emergency Kill Switch

EU AI Act Art. 14(1) Compliant

The kill switch is an **emergency halt** mechanism — not a simple toggle. When triggered, it executes three actions atomically:

01 — Server deactivated

The MCP server is immediately taken offline across the entire cluster.

02 — All tokens revoked

Every connection token is invalidated. Total lockout — reconnection blocked until new tokens are issued.

03 — WebSocket connections killed

Active connections terminated via Redis pubsub broadcast. Propagates to every runtime node in the cluster.

Full Visibility. Zero Guesswork.

The Vinkius cloud dashboard includes a full MCP Governance suite — real-time analytics and security controls for production AI operations.

Control Plane

KPI dashboard with request volume, latency, success rate, token consumption, and AI-generated operational briefings.

Agent Activity

Which AI clients are connecting, how often, and what they're doing — real-time session tracking.

FinOps

Cost tracking per tool, payload compression savings, budget optimization signals, and consumption trends.

Tool Health

Slowest and most error-prone tools, with actionable root-cause insights and performance baselines.

Firewall & DLP

PII redaction activity, sensitive data protection counters, and security event timeline.

Incident Log

Error trends, failure rates, status-code breakdowns, and forensic audit trail access.

> Get started at **cloud.vinkius.com** — connect your AI agent in under 60 seconds.

Data Sorting & Filtering Engine MCP

2 tools available
Cloud-hosted on Vinkius

When you're dealing with complex data—say, an array containing over 500 user profiles or product records—standard LLMs hit a wall. They lose elements, hallucinate missing fields, and can’t reliably maintain order across huge datasets. This MCP bypasses those limitations by using native JavaScript Array operations for perfect results.

It guarantees flawless sorting, whether you need alphabetical, numerical, or length-based ordering. Plus, it cleans up your data by identifying and grouping exact duplicates based on a key you provide. When you connect this to Vinkius, your AI client can run these powerful data cleanup routines directly against structured JSON, giving you deterministic results without relying on the model's limited memory. You just pass the list, specify what needs fixing, and get back a perfect, clean dataset.

Core Capabilities

01 — Deterministic Array Sorting

Sort massive JSON arrays reliably by any specified key (alphabetical or numerical) in ascending or descending order.

02 — Structured Duplicate Removal

Remove exact duplicate records from a large array, grouping them deterministically based on a specific identifier key.

03 — Bulk Data Filtering and Cleanup

Process raw JSON data to eliminate inconsistencies and structure the output for immediate use in downstream analysis.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/data-sorting-filtering-engine** — connect your AI agent in three steps.

01 You feed this MCP a large, unsorted, or duplicate-filled JSON array—the dataset you need cleaned up.

02 Your AI client determines whether the data needs sorting (by key and direction) or filtering (for deduplication).

03 The engine runs the task using highly optimized native JavaScript functions and returns a guaranteed clean, perfectly structured JSON output.

The bottom line is that your agent gets back a mathematically perfect version of your data set, regardless of how large it was to start with.

Built For

Data Analysts and Backend Engineers who regularly process raw API dumps or database exports. If you've ever had an AI agent fail on a dataset over 100 records, this MCP is for you.

Data Analyst

Using the engine to clean massive CSV imports or API dumps before running statistical models. They need guaranteed sorted output to ensure accurate trend analysis.

Backend Engineer

Implementing data validation and cleanup pipelines, ensuring that incoming JSON payloads are consistently structured and free of duplicates in production code.

Data Scientist

Preparing training datasets by reliably sorting features or removing redundant records to improve model accuracy without manual pre-processing steps.

What Changes When You Connect

01 Guaranteed Data Integrity: You won't lose elements or hallucinate missing fields when processing thousands of records. The engine maintains the full dataset structure.

02 Perfect Sorting: Use `sort_array` to guarantee flawless sorting by any key—whether it's a date, price, or name—in exact order (A-Z or highest-lowest).

03 Efficient Deduplication: Quickly run `remove_duplicates` on large lists, using a specific field as the grouping key so you can eliminate redundant entries reliably.

04 Bypasses LLM Limits: This MCP uses native JavaScript performance. Your agent doesn't rely on the model's context window to handle data larger than what it can remember.

05 Structured Output: The output is always a clean, structured JSON object ready for immediate use in your next step or script.

Real-World Applications

Cleaning up API Dump Data

A data analyst receives a 2,000-record JSON dump from an external service. They ask their agent to use the engine to sort all records by 'transactionDate' and remove duplicates based on 'receiptId'. The MCP returns a perfectly clean, chronological list ready for reporting.

Preparing Product Catalog Data

A backend engineer needs to validate incoming product data. They use the engine to deduplicate a batch of 500 products by 'SKU' and then sort them by 'price' descending before saving them to the database.

Analyzing User Activity Logs

A data scientist has massive user activity logs. They use the MCP to remove duplicate log entries (based on a combination of user ID and timestamp) and then sort the remaining list by 'activityType' for pattern recognition.

Validating Database Exports

A team member downloads multiple database exports. They feed them into the engine to ensure all arrays are sorted by a primary key, verifying consistency across different sources before merging data streams.

Patterns to Avoid

Relying on LLMs for large sorting

✗ AVOID

Prompting an agent: 'Sort this 800-item list of users by last name.' The resulting JSON is incomplete, missing elements, or misordered.

✓ INSTEAD

Instead, use the `sort_array` tool. Pass the full array and specify the key and direction. The engine handles the sorting deterministically, guaranteeing every element stays put.

Manual deduplication via text prompts

✗ AVOID

Asking an agent: 'Remove duplicate records from this list.' The agent might remove duplicates but fail to group them correctly or lose context on which key defines a true duplicate.

✓ INSTEAD

Use the `remove_duplicates` tool. You must provide the specific grouping key (e.g., 'email' or 'productID'). This forces deterministic cleanup based on your exact criteria.

Processing unsorted, mixed data

✗ AVOID

Feeding a list of product records where some prices are floats and others are strings into the agent for sorting. The sort fails or treats numbers as text.

✓ INSTEAD

Use `sort_array` while explicitly telling it to treat the key as numeric (or string, if appropriate). This ensures mathematical precision when sorting by values like 'price'.

The Right Fit

You should use this MCP whenever your data processing task involves arrays of items over a few dozen. Use it if you need guaranteed deterministic results—meaning the output is predictable and correct, regardless of how large or messy the input is. Don't use it if you only have two or three records to process; simple prompts will work fine there. Also, don't use it if your goal is data transformation (e.g., summarizing text); this MCP strictly handles structural manipulation. If you are trying to group related items but not remove duplicates, consider a different tool for categorization instead of relying on `remove_duplicates`.

Using Data Sorting & Filtering Engine with JSON Array Cleanup

Right now, dealing with raw data dumps is manual hell. You're downloading massive API responses into a spreadsheet or pasting them into an AI agent, only to find out the dataset is unsorted, has redundant records, and worse—the LLM you used messes up the structure because it can't hold all those details in its memory.

With this MCP, you pass the messy JSON array straight through. The engine automatically handles the cleanup: running `remove_duplicates` to eliminate noise or using `sort_array` to put everything in perfect order. You get back a clean, predictable list that's ready for modeling.

Data Sorting & Filtering Engine and Array Manipulation Precision

The manual steps of validating data consistency—checking if all records are sorted by the primary key or manually hunting down duplicate entries across different tabs—are time sinks. You spend more time cleaning data than analyzing it.

This MCP eliminates those checks entirely. It gives you deterministic control over your array structure, making data preparation instantaneous and mathematically reliable.

Data Sorting & Filtering Engine: 2 Tools for JSON Array Operations

Use these tools to reliably sort, filter, and remove duplicate records from large JSON arrays with deterministic accuracy.

#	TOOL	DESCRIPTION
01	`remove_duplicates`	Pass an array and a grouping key, and the engine returns a map containing only unique entries from that list.
02	`sort_array`	Sorts any JSON array deterministically by specifying a key and whether the order should be ascending or descending.

See It in Action

Real prompts you can use once this MCP is connected to your AI agent through Vinkius Cloud.

U Sort this JSON list of 100 users by 'email' ascending so I can check for duplicates.

✅ **Sorted & Cleaned:**

The array is now perfectly sorted. We found and removed 7 duplicate records based on the unique identifier.

Example Output Snippet:

```json
[
{"id": 102, "email": "alice@corp.com", "name": "Alice"},
{"id": 234, "email": "bob@corp.com", "name": "Bob"},
{"id": 500, "email": "chris@corp.com", "name": "Chris"}
]
```

U I have a product list of 1,200 items. Can you sort them by 'price' descending and remove any duplicates based on the 'productID'?

✅ **Sort Applied & Duplicates Removed:**

Successfully processed 1,200 records. The array is now perfectly ordered by price.

- **Records Processed:** 1,200
- **Duplicates Found and Removed:** 45
- **Final Count:** 1,155 items

The top item (highest price) is at index 0.

U Take this historical event array. I need it sorted by year descending, but first, make sure we only keep the unique events based on 'eventName'.

✅ **Operation Complete:**

The dataset has been filtered and perfectly reordered.

Filtering Action: *Removed 12 duplicate event entries.
* Key used: eventName*

• **Sorting Action:** Sorted the remaining events by year (descending).

Example Output Snippet:

```json
[
{"year": 2050, "event": "Mars Colony Founding"},
{"year": 1985, "event": "Global Internet Launch"},
{"year": 1888, "event": "Industrial Revolution Begins"}
]
```

Frequently Asked Questions

01 **Why do I need the Data Sorting & Filtering Engine MCP for AI Agents instead of just asking Claude to sort my data?**

This MCP uses native JavaScript, which is far more reliable than an LLM's internal logic. If your array has hundreds of items, general AI agents often lose context or fail to maintain order. This engine guarantees perfect sorting and integrity every time.

02 **Can this Data Sorting & Filtering Engine MCP handle JSON data that is extremely large?**

Yes. It was built specifically for datasets too big for standard LLM context windows. You can reliably process thousands of records without worrying about the model forgetting elements or losing track.

03 **How do I use the Data Sorting & Filtering Engine MCP if my list has duplicates?**

You simply point it to your array and tell it which field defines a duplicate (like 'email' or 'productID'). The engine uses that grouping key to remove all redundant entries deterministically.

04 **What kind of data can the Data Sorting & Filtering Engine MCP sort? Is it limited?**

It handles standard JSON arrays. You can sort by names (alphabetical), dates, or numbers. It uses native JavaScript logic, so the sorting is always precise and predictable.

05 **Is this Data Sorting & Filtering Engine MCP better than using Python code for data cleanup?**

It's a high-level abstraction of those best practices. You get powerful, deterministic array manipulation without having to write the underlying JavaScript or Python logic yourself.

Go Live in 60 Seconds

Get your connection token from **cloud.vinkius.com**, then paste the endpoint URL into any MCP-compatible client.

YOUR MCP ENDPOINT

```
https://edge.vinkius.com/[TOKEN]/mcp
```

CLIENT	WHERE TO CONFIGURE
Claude AI	Profile → Customize → Connectors → "+" → Add custom connector → Paste endpoint
Cursor	Settings → Features → MCP Servers → "+ Add New MCP Server" → Type: SSE → Paste endpoint
VS Code	Ctrl/Cmd+Shift+P → "MCP: Add Server" → add `"data-sorting-filtering-engine": { "url": "…" }`
Windsurf	MCP Settings → `mcp_settings.json` → Add endpoint URL
ChatGPT	Settings → Tools & plugins → Add MCP server → Paste endpoint
Gemini	Extensions → Add MCP Server → Paste endpoint URL

ASK AN AI ABOUT THIS

Let your preferred AI explain this MCP server

- Ask ChatGPT
- Ask Claude
- Ask Perplexity
- Ask Gemini
- Ask Grok

READY TO CONNECT

Data Sorting & Filtering Engine is live on Vinkius Cloud.

Get your connection token, paste it into your AI agent, and start building. No SDK. No deployment. Just results.

Start at cloud.vinkius.com →

vinkius.com · support@vinkius.com

INDEPENDENT PLATFORM DISCLAIMER

Vinkius is an independent platform and is not affiliated with, endorsed by, sponsored by, verified by, or otherwise authorized by Data Sorting & Filtering Engine. All third-party trademarks, logos, and brand names are the property of their respective owners. Their use in this document is strictly for informational purposes to identify service compatibility and interoperability.

DOCUMENT INFORMATION

Generated	June 2026
MCP Server	Data Sorting & Filtering Engine MCP
Server ID	`019e3885-8bbe-730d-8f39-837552243978`
Platform	Vinkius Cloud for AI Agents
Endpoint	`https://edge.vinkius.com/{token}/mcp`

LICENSE & USAGE

This document is generated automatically by the Vinkius PDF Engine. Content reflects the MCP server configuration at the time of generation and may change as updates are deployed. For the most current information, visit vinkius.com/mcp/data-sorting-filtering-engine.