MCP SERVER | NO CODE | CLOUD HOSTED

Parseur MCP

Turn Messy PDFs and Emails Into Actionable JSON Data

Parseur automates document processing and data extraction for your AI agents. It connects directly to complex pipelines, allowing you to upload PDFs, emails, or bulk documents and extract structured fields—like invoice numbers, total amounts, dates, and line items—into usable JSON format. You define the rules using templates, and our OCR engine handles the rest, turning unstructured paper into actionable data points for your workflow.

A+ Quality Score 100/100

ocr | data-parsing | email-automation | pdf-processing | template-extraction | structured-data

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/parseur

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.

Parseur MCP

10 tools available
Cloud-hosted on Vinkius

When you need to read things that aren't in neat tables, this MCP is what you use. Forget manually opening every PDF or email attachment just to pull out an invoice number. This connector lets your agent process entire document streams automatically. It handles the messy stuff—whether it’s a scanned receipt with skewed text or a multi-page email thread. You set up specific mailboxes and templates, telling the system exactly what fields you need (e.g., 'invoice total' or 'date'). Then, when documents arrive, your agent pushes them through the pipeline for parsing. The result is clean JSON data that your next step can use immediately. If you’re managing document logic across multiple AI clients, Vinkius makes connecting this entire process reliable and straightforward.

Core Capabilities

01 — Create Document Pipelines

You set up dedicated parsing mailboxes for specific document types like invoices or emails.

02 — Define Extraction Logic

You create templates that map fields and define the precise rules needed to pull structured data from documents.

03 — Submit Documents for Parsing

Your agent uploads document URLs or raw payloads into a configured mailbox queue.

04 — Retrieve Structured Data

You pull the fully extracted JSON data from documents once they have finished processing.

05 — Check Document Status

You list all processed or failed documents to track a batch job’s progress and status.

06 — Force Pipeline Retries

If an extraction fails due to a minor error, you can instantly push the document back into the pipeline for reprocessing.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/parseur** — connect your AI agent in three steps.

01 First, list all available parsing pipelines using `list_mailboxes` or create new ones with `create_mailbox` to define what type of documents you process.

02 Next, use `create_template` to build the extraction rules and tell the system exactly which fields (e.g., total amount) you expect to find in those documents.

03 Finally, run your workflow by uploading a document URL using `upload_document`; after processing, retrieve the clean data structure with `get_document_data`.

The bottom line is that this MCP takes unstructured files and converts them into predictable, structured JSON objects for any application or agent to use.

Built For

Anyone dealing with high volumes of varied physical documents—invoices, contracts, receipts—will need this. If your job involves reading data from PDFs that aren't in neat tables, stop using spreadsheets and start connecting this MCP.

Accounts Payable Specialist

They use the tool to automate processing new vendor invoices, ensuring every required field like PO number and net total is captured instantly without manual data entry.

Financial Analyst

They rely on it to batch process large sets of scanned receipts or financial statements, extracting key metrics into a usable JSON format for monthly reporting.

Integration Developer

They use the tool to simulate document loads and test how data flows from an external source into their internal record-keeping systems via webhook logic.

What Changes When You Connect

01 Stop manually pulling data. By using `upload_document`, you route entire batches of documents into the parsing engine, getting clean, structured output instantly.

02 Handle different document types without changing logic. You can define multiple pipelines—one for invoices, one for receipts, etc.—using separate mailboxes and templates.

03 Don't get stuck on failed documents. If a parse fails due to an error, just call `retry_document` to re-run the pipeline against that specific document ID.

04 Get exactly what you need. Use `get_document_data` to retrieve only the structured fields (like total amount and date) without getting bogged down in raw metadata.

05 Understand your setup before sending files. Check the mailbox configuration with `get_mailbox` to verify that the correct parsing rules are active for a given document type.

Real-World Applications

Processing End-of-Month Vendor Invoices

The AP manager needs to process 50 invoices from different vendors. Instead of manually entering the invoice number and total into a ledger, they use `list_mailboxes` to identify the 'Vendor Invoice' pipeline and then run their agent to execute `upload_document` for all 50 files, retrieving structured data via `get_document_data`.

Cleaning up Failed Scans

A batch of scanned receipts failed parsing due to a bad template rule. Instead of manually fixing the documents, an agent uses `list_documents` to identify the failed IDs and then calls `retry_document`, forcing the system to re-run the OCR against the fixed template.

Building a Multi-Source Data Stream

A developer needs to ingest both PDF contracts and email attachments. They use `list_mailboxes` to confirm two separate pipelines exist, then route documents using `upload_document` into the correct stream for parsing.

Debugging Data Flow

An integration needs to verify if a document is ready for processing. It first calls `get_mailbox` to check the configuration details before attempting any file uploads, ensuring data integrity across systems.

Patterns to Avoid

Assuming all documents are the same format

✗ AVOID

Trying to extract a 'Date' field from both a handwritten note and a formal PDF using one single template fails because the system doesn't know which parsing engine to use.

✓ INSTEAD

You must define separate, specific pipelines. First, `create_mailbox` for 'Handwritten Notes', then another with `create_mailbox` for 'Formal PDFs'. Use different templates and rules for each type.

Only retrieving document metadata

✗ AVOID

Using only the tool to get basic details is useless because you still have to copy-paste the actual invoice amount from a separate view.

✓ INSTEAD

After confirming the status with `get_document_details`, always follow up by calling `get_document_data` to pull the structured, usable JSON fields directly.

Manually checking every document for errors

✗ AVOID

A job fails on 10 documents; the user has to open each one and manually determine if a retry is needed.

✓ INSTEAD

List all failed jobs using `list_documents`. Then, use `retry_document` across the batch of IDs that need fixing. This automates the error remediation.

The Right Fit

Use this MCP when your input data source is unstructured (scanned PDFs, emails, varied forms) or semi-structured but messy. If you can't reliably point to a field using simple XPaths or if the format changes often, you need Parseur. Don't use it if you are dealing with clean databases where every field already has a consistent schema and is stored in a single source of truth (in that case, standard database connectors work fine). Also, don't use this just to read

text; use `get_document_data` because it guarantees the data comes back as structured JSON fields. If you only need basic file management, simply listing documents with `list_documents` is enough.

The constant copy-paste tax on your team's time

Think about the end of the month: opening dozens of PDFs and emails. You open one, find the invoice number in a corner, copy it into your spreadsheet. Open the next three, repeating that process—finding the total amount, pasting it, verifying the date. Your workflow becomes a cycle of clicking 'View Attachment,' manually reading, selecting text, copying, and pasting. This is tedious, error-prone work.

With this MCP, you just point your agent at the folder full of documents. The system handles opening, reading, parsing, and extracting those specific fields into clean data packets. You don't copy anything; you receive a ready-to-use JSON object containing everything you needed.

Extract structured data with Parseur

Manual steps like verifying the document status, defining which fields are critical for extraction, and confirming that all necessary templates are active disappear. You manage these processes using tools like `get_mailbox` and `list_templates`, letting your agent handle the complexity.

The difference is control. Instead of hoping a spreadsheet formula catches everything, you define the rules explicitly. Your data moves from being 'a PDF' to being 'Invoice Number: A-201, Total Amount: 1400.99'—and that structure never changes.

Parseur: 10 Document Parsing Tools

These tools let your agent manage entire document pipelines—from creating new parsing rules to retrieving the final structured JSON output.

#	TOOL	DESCRIPTION
01	`create_mailbox`	Sets up a new dedicated parsing pipeline, specifying if the mailbox handles PDFs, emails, or attachments.
02	`create_template`	Defines the extraction rules and field mappings needed for the system to pull structured data from incoming documents.
03	`get_document_data`	Retrieves the complete, parsed JSON dictionary of extracted fields from a document that has been successfully processed.
04	`get_document_details`	Fetches only the metadata about a single parsed document, such as its ID and status, without the actual extracted data.
05	`get_mailbox`	Provides detailed configuration information for a specific parsing mailbox to confirm its setup parameters.
06	`list_documents`	Lists all documents within a mailbox, showing their ID, current status (processed/failed), and date details.
07	`list_mailboxes`	Retrieves a list of every existing parsing pipeline configured for the account, along with their unique IDs.
08	`list_templates`	Shows all defined extraction templates associated with a mailbox, detailing the rules used to pull data.
09	`retry_document`	Forces a failed or errored document back into the parsing queue so it can be matched against the latest template rules.
10	`upload_document`	Sends a document URL to a specified mailbox, immediately entering the file into the processing queue for OCR logic.

See It in Action

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

U Check my Parseur mailboxes to find the specific bounding IDs.

Extracted physical boundaries querying mailboxes. I found 2 limits explicitly registered: 1. 'Invoices Mailbox' (ID: xyz12), expecting PDF accounting payloads. 2. 'Web contact forms' (ID: wtf9). Should I list templates associated cleanly?

U Get the data schema parsed tightly inside document doc_987.

Running payload bounds against doc_987. Status: parsed perfectly. Extracted JSON dictionary properties explicitly reveal: Invoice_Number: 'A-201', Total_Amount: 1400.99, Date: '2026-04-10'.

U Upload this snippet of parsed text directly into Mailbox xyz12 for OCR processing.

Push executed successfully running `upload_document`. Triggered physical limits. The document natively entered 'Invoices Mailbox' parsing queue. Processing document UUID: doc_112. Status: pending. I'll dynamically pull it later if told.

Frequently Asked Questions

01 **How do I get started with Parseur and structured data?**

You start by calling `list_mailboxes` to see what pipelines are available or creating a new one using `create_mailbox`. Then, you define the rules for that pipeline using `create_template`.

02 **Does Parseur handle scanned documents?**

Yes. The MCP uses powerful OCR logic to read text from images and scans. You just need to upload the document via `upload_document`, and the engine handles the rest of the parsing process.

03 **What is the difference between get_document_data and list_documents?**

Use `list_documents` when you only want a summary table showing which files exist and their status. Use `get_document_data` when you need the actual, fully parsed structured data from one specific file ID.

04 **Can I fix documents that failed parsing using Parseur?**

Absolutely. If a document fails validation, use `list_documents` to get the IDs of the failures, and then call `retry_document` to force a fresh parse run.

05 **What is an 'extraction template' in Parseur?**

An extraction template defines the rules—the field names, locations, and regex patterns—that tell the system exactly what data points (like tax ID or date) to pull from a messy document.

06 **How do I test my parsing setup before uploading files?**

Before running `upload_document`, it's smart to first use `get_mailbox` and `list_templates`. This lets you review the current configuration, ensuring your rules are set up correctly.

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 `"parseur": { "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

Parseur 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 Parseur. 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	Parseur MCP
Server ID	`019d75ef-8869-708e-8e07-012b5684d5fd`
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/parseur.