MCP SERVER | NO CODE | CLOUD HOSTED

Arbiscan (Arbitrum Explorer) MCP for AI Agents

Analyze Arbitrum L2 Smart Contracts and Track Token Movements

Arbiscan lets you query the Arbitrum blockchain in real time directly from your chat or IDE. You can check ETH balances for multiple wallets, trace every type of token transfer (NFTs, ERC20), and pull verified smart contract source code. It gives developers and analysts instant access to deep L2 network data without leaving their workflow.

A+ Quality Score 98.33/100

arbitrum | block-explorer | smart-contracts | ethereum-l2 | web3

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/arbiscan-arbitrum-explorer

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.

Arbiscan (Arbitrum Explorer) MCP

16 tools available
Cloud-hosted on Vinkius

Arbiscan connects your AI agent directly to the Arbitrum Layer 2 network. This means you can query complex blockchain data—like transaction history, token movements, or even raw contract source code—right inside your chat interface or development environment. Instead of opening a browser and clicking through multiple tabs just to check balances, you simply ask your agent. You get immediate answers on whether an address holds ETH, how many times a specific NFT moved, or what functions a smart contract actually contains. This kind of deep access is crucial for anyone working with decentralized applications (dApps). When you use this MCP via Vinkius, all that raw blockchain data flows into your agent's context, letting it perform complex analysis using natural language prompts. It drastically cuts down the manual labor of monitoring multiple addresses or tracking historical token transfers.

Core Capabilities

01 — Check Balances for Multiple Addresses

Get the current Ether balance across a list of one or more blockchain addresses.

02 — Track Full Transaction History

Retrieve complete transaction records, including internal calls, standard transfers, and NFT activity.

03 — Analyze Contract Code Structure

Fetch the contract's Application Binary Interface (ABI) or its verified source code to understand how it operates.

04 — Monitor Network Metrics

Get current data points like the real-time ETH price on Arbitrum, total supply, and block rewards.

05 — List Specific Token Movements

Filter transaction logs to show only ERC20 token transfers or specific NFT (ERC721) movements for an address.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/arbiscan-arbitrum-explorer** — connect your AI agent in three steps.

01 Subscribe to this MCP and provide your Arbiscan API key.

02 Reference the blockchain data you need in a natural language prompt to your AI agent.

03 Your agent calls the appropriate tool, executes the query against Arbitrum, and returns the structured data directly into the conversation.

The bottom line is that your AI client performs all the complex API calls behind the scenes, so you just get clean, actionable insights about the blockchain.

Built For

Anyone who spends time looking at decentralized finance or smart contracts needs this. It's for developers needing to verify code and analysts needing to track complex asset movements that standard explorers can’t easily handle.

Web3 Developer

Debugging a dApp means checking if the contract logic holds up. You use this MCP to fetch ABIs and source code while writing local functions, making sure your smart contracts interact correctly.

Crypto Analyst

Tracking 'whale' movements or monitoring token flows across multiple wallets becomes simple. You ask the agent for all ERC20 transfers on a target address and get a clean list instantly.

Security Researcher

When auditing transactions, you need to see more than just the final result. Use this MCP to pull internal transaction calls and event logs to spot suspicious or unexpected activity.

What Changes When You Connect

01 Audit multiple wallets at once. Instead of checking balances one by one, use `get_balance_multi` to check ETH holdings across an entire list of addresses in a single prompt.

02 Deep transaction visibility. You can track the full lifecycle of assets using tools like `get_token_nft_tx` and `get_token_tx`, going beyond simple balance checks.

03 Developer-grade insights. Pulling contract source code with `get_source_code` or checking the ABI via `get_abi` lets your agent read exactly how a dApp functions, which is vital for building.

Real-World Applications

Investigating an Unknown Token Transfer

A user suspects funds were moved incorrectly. They prompt the agent to use `get_tx_list` and then filter results using `get_logs`. The agent identifies a specific internal call that reveals the token type, allowing the analyst to report accurately.

Verifying Smart Contract Integrity

A developer is integrating with a new dApp. Instead of manually comparing documentation, they ask their agent to `verify_source_code` and pull the ABI using `get_abi`. The AI confirms the contract structure before any code is written.

Tracking Complex NFT Ownership

A collector wants to know every time a specific piece of digital art moved. They prompt for ERC721 transfers, and the agent uses `get_token_nft_tx` to provide a clean, chronological list of ownership changes.

Assessing Network Health

A financial analyst wants to know if ETH prices are fluctuating relative to total supply. They query both the current market price using `get_eth_price` and the total circulating supply using `get_eth_supply` in one go.

Patterns to Avoid

Treating all transaction types equally

✗ AVOID

Just asking 'Show me transactions for this address' gives a massive, unfilterable list of data that mixes basic transfers with complex internal calls and NFT activity. It's impossible to find the needle.

✓ INSTEAD

You need specific tools. If you want token history, use `get_token_tx` or `get_token_nft_tx`. For deep debugging, pull both general transactions using `get_tx_list` AND internal calls using `get_tx_list_internal` to get the whole picture.

Ignoring contract structure details

✗ AVOID

Assuming a token is standard when in reality it uses complex, custom logic. The AI can't read the source code and will only report balances, missing critical functional details.

✓ INSTEAD

Always run `get_source_code` or `get_abi` first. This lets your agent understand *how* the contract works before interpreting any transaction data.

Focusing only on the last balance

✗ AVOID

Just asking for the current ETH balance (`get_balance`) doesn't tell you if that balance was acquired through a suspicious internal transfer or a routine sale.

✓ INSTEAD

To understand where the funds came from, pull transaction details. Use `get_tx_list` and check the corresponding event logs via `get_logs` to trace the origin of the funds.

The Right Fit

Use this MCP when your task requires deep, technical visibility into the Arbitrum L2 state. Specifically, if you need to track multiple addresses' balances (`get_balance_multi`), analyze the underlying code structure (`get_abi` or `get_source_code`), or distinguish between different types of token movements (ERC20 vs. NFT), this is your tool. Don't use it if you just need a general market overview —use dedicated price APIs for that. Also, don't rely on it to execute transactions; it only reads data. If you are trying to initiate a transfer, you need a separate sending service.

Arbiscan (Arbitrum Explorer) MCP: Analyzing Smart Contract Logic

Manually checking smart contract behavior is tedious. You're stuck opening the block explorer, finding the contract address, and clicking through tabs for source code, ABI details, and transaction logs. It’s a multi-step process involving copy/pasting hashes just to confirm basic functions.

With Arbiscan, you simply ask your agent, 'What does this contract do?' The MCP pulls the verified source code using `get_source_code` , retrieves the ABI with `get_abi` , and structures that data for you. You get instant confirmation of functionality without touching a browser tab.

Arbiscan (Arbitrum Explorer) MCP: Tracking Complex Token Movements

Tracking asset flows across different standards is a nightmare. You have to manually check if the transfer was an ERC-721 NFT event or a standard ERC-20 token movement, and then cross-reference internal calls for any associated fees.

This MCP solves that by letting your agent intelligently query both `get_token_nft_tx` and `get_token_tx`. You get one unified report showing the full history of every asset class linked to an address. The complexity is handled automatically.

Arbiscan (Arbitrum Explorer): 16 Tools for Blockchain Data Analysis

Use these tools to retrieve everything from a single wallet's ETH balance to the full, verified source code of complex smart contracts on Arbitrum.

#	TOOL	DESCRIPTION
01	`get_abi`	Get Contract ABI for Verified Source Codes
02	`get_balance_multi`	Get Ether Balance for Multiple Addresses
03	`get_balance`	Get Ether Balance for a Single Address
04	`get_block_countdown`	Get Estimated Block Countdown Time by BlockNo
05	`get_block_reward`	Get Block Rewards by BlockNo
06	`get_eth_price`	Get Ether Last Price
07	`get_eth_supply`	Get Total Supply of Ether on Arbitrum
08	`get_logs`	Get Event Logs
09	`get_source_code`	Get Contract Source Code for Verified Source Codes
10	`get_status`	Check Contract Execution Status
11	`get_token_nft_tx`	Get List of ERC721 Token Transfer Events By Address
12	`get_token_tx`	Get List of ERC20 Token Transfer Events By Address
13	`get_tx_list_internal`	Get List of Internal Transactions By Address
14	`get_tx_list`	Get List of Normal Transactions By Address
15	`get_tx_receipt_status`	Check Transaction Receipt Status
16	`verify_source_code`	Verify Source Code

See It in Action

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

U Show me all the token transfers for this wallet last week, broken down by NFT and standard ERC20 tokens.

✅ **Token Movement Summary (Past 7 Days)**

- **Total ETH Balance:** 4,512.89 ARB
- **NFT Activity (ERC-721):** 3 transfers recorded.
* `[Date]` received: CoolCat #1001 (From 0xabc...)
* `[Date]` sent: Bored Ape #55 (To 0xxyz...)
- **Standard Tokens (ERC-20):** 7 transfers recorded.
* Most recent: +5,000 USDC from Uniswap LP
* Oldest: -12.5 ETH to payment address

U What is the current Ether balance for these three addresses and what's the total supply?

📊 **Arbitrum Network Snapshot**

ADDRESS	CURRENT BALANCE (ETH)
0x123...	78.12
0x456...	1,900.55
0x789...	0.00

Network Metrics:

- Total ETH Supply: 120 Million ARB
- Current ETH Price: $1,245.67

U I need to check the source code for this contract and see if it has any internal calls.

🔎 **Contract Analysis Results**

Source Code Status: Verified Source Code found.
- The contract uses 4 primary functions, including `transfer` and `mintNew`.
- **ABI Snippet:** Shows the function signature: `bytes32 transfer(address to, uint256 amount)`

Internal Calls Detected: Yes. The transaction history reveals an internal call to a fee collection contract at `0xfee...`, suggesting mandatory gas payments are processed off-chain.

Frequently Asked Questions

01 **How does Arbiscan (Arbitrum Explorer) help me check if my smart contracts are secure?**

You can audit contract logic by using the MCP to fetch the ABI and verified source code. This lets your agent read exactly what functions exist and how they are written, helping you spot potential vulnerabilities before deployment.

02 **Can Arbiscan (Arbitrum Explorer) track all kinds of crypto asset movements?**

Yes, it tracks everything. It can list standard ERC-20 token transfers, unique NFT (ERC721) movements, and even internal calls that show where funds went after a main transaction completed.

03 **What if I need to check balances for dozens of wallets? Is Arbiscan (Arbitrum Explorer) good for that?**

Absolutely. Instead of checking each wallet individually, you provide a list of addresses in one prompt. The MCP uses the `get_balance_multi` tool to give you all the current ETH balances simultaneously.

04 **Does Arbiscan (Arbitrum Explorer) just show me transaction history or can it analyze it?**

It does both. It retrieves the raw list of transactions using `get_tx_list`, and then your agent analyzes that data—for example, by pulling event logs (`get_logs`) to tell you exactly what happened during the transfer.

05 **Is Arbiscan (Arbitrum Explorer) only for developers?**

No. Crypto analysts use it constantly. You can track 'whale' movements, monitor token flows between addresses, and get a clear picture of market activity without needing to write any code.

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 `"arbiscan-arbitrum-explorer": { "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

Arbiscan (Arbitrum Explorer) 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 Arbiscan (Arbitrum Explorer). 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	Arbiscan (Arbitrum Explorer) MCP
Server ID	`019e3867-0796-72a8-b485-7d71463e42c7`
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/arbiscan-arbitrum-explorer.