MCP SERVER | NO CODE | CLOUD HOSTED

Descope MCP for AI Agents

Manage and test complex user sign-up and OAuth flows

Descope MCP lets your AI agent manage complex user authentication flows directly from natural conversation. Test sign-ups, logins, and session management using OTPs (SMS, Email, Voice), Magic Links, OAuth providers like Google, or traditional passwords. It's built for developers who need to verify auth logic without leaving their terminal.

A+ Quality Score 100/100

authentication | otp | oauth | magic-link | user-management

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/descope-auth-platform

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.

Descope (Auth Platform) MCP

33 tools available
Cloud-hosted on Vinkius

Testing a new login flow used to mean spinning up test accounts in a dashboard, clicking through multiple redirect URLs, and manually checking logs. Now, you just talk to your AI agent. This MCP connects Descope's full suite of authentication tools directly into your workflow. Your agent handles the complexity: initiating an SMS code, waiting for confirmation, then using that success token to verify the session. It covers everything from simple password signups (`auth_password_signup`) to advanced OAuth exchanges (`auth_oauth_exchange`). If you're building any application with user accounts, this makes testing those security boundaries trivial. When your team needs a comprehensive catalog of ways to test auth logic, Vinkius brings all these capabilities together in one place.

Core Capabilities

01 — Test One-Time Password (OTP) Flows

Initiate and verify user accounts using email, SMS, or voice codes for sign-up and sign-in.

02 — Manage Secure Link Authentication

Send out secure Magic Links or Enchanted Links to onboard users remotely, then poll until the session is complete.

03 — Run OAuth Provider Workflows

Start third-party logins (like Google) and exchange authorization codes for active user sessions.

04 — Perform Basic User Credentials Signups

Handle standard sign-up or password resets using traditional username/password methods.

05 — Manage Platform Users and Roles

Create, read, update, and delete platform users, roles, permissions, and tenants for internal system testing.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/descope-auth-platform** — connect your AI agent in three steps.

01 First, subscribe to this MCP on Vinkius and provide your Descope Project ID and any necessary management keys.

02 Next, you prompt your AI client with a natural language request describing the user action—for example, 'Sign up a new test user using Google OAuth.'

03 Your agent executes the required tool sequence (like `auth_oauth_authorize`) and reports the result back to you: success confirmation, required inputs, or specific error codes.

The bottom line is that your AI agent acts as a fully functional QA engineer for your authentication stack, completing multi-step user journeys in real time.

Built For

This MCP is essential for developers and quality assurance teams who struggle with manually testing complex, multi-step sign-in flows. If your application relies on any form of user identity—whether it's passwords, Google logins, or SMS codes—you need this.

QA Engineer

Automating the verification of edge cases: what happens if an OTP expires? How does a Magic Link fail to verify? Running these scenarios repeatedly without writing boilerplate code.

Backend Developer

Testing new auth endpoints or providers (like adding support for GitHub OAuth) by simply instructing the agent to run the flow and analyzing the session token output.

Product Manager

Inspecting how different identity methods are configured in your project before handing off flows to engineering. You can verify if your system supports all planned sign-in routes.

What Changes When You Connect

01 Test the full range of authentication methods, from traditional passwords to advanced OAuth redirects. You can initiate a flow using `auth_oauth_authorize` and immediately validate it with your agent.

02 Streamline QA testing by automating multi-step verification. Instead of simulating clicks, your agent executes tool calls like `auth_magiclink_signup_email`, handles the code, and confirms the session status via `auth_enchantedlink_poll`.

03 Handle user identity management in bulk. You can use tools like `mgmt_create_user` or `mgmt_list_tenants` to provision and de-provision test accounts quickly before running a full auth cycle.

04 Cover all OTP scenarios: The agent handles both the initial signup (`auth_otp_signup_sms`) and subsequent verification steps (`auth_otp_verify_email`), giving you comprehensive coverage.

05 Manage system permissions alongside user flow testing. Tools like `mgmt_create_role` let you ensure that a newly authenticated user has the correct access levels for their role.

Real-World Applications

Verifying Google OAuth Integration

A developer needs to confirm that connecting Google works correctly. They ask their agent, 'Run the Google OAuth flow.' The agent triggers `auth_oauth_authorize` and then uses `auth_oauth_exchange` to prove a valid session was created.

Testing New OTP Methods

QA needs to validate voice authentication. They instruct the agent to 'Test new Voice OTP sign-up.' The MCP initiates the process (`auth_otp_signup_voice`) and then verifies a sample code using `auth_otp_verify_voice`.

Onboarding a Test User Via Magic Link

A PM needs to simulate remote sign-up. They prompt the agent to 'Sign up user X with a Magic Link.' The MCP executes `auth_magiclink_signup_email` and then confirms the account is active using `auth_enchantedlink_poll`.

Debugging Role Permissions

A developer changes a system role. They first run `mgmt_create_role` to set up the new role, create a test user with that role (`mgmt_create_user`), and then use a simple login flow to verify access.

Patterns to Avoid

Testing flows manually in logs

✗ AVOID

Spending hours copying success/failure codes from multiple browser tabs or console outputs after each sign-in attempt.

✓ INSTEAD

Use the MCP to automate these checks. Start with `auth_password_signup` and then immediately use a subsequent tool call like `mgmt_get_user` to verify the credentials were saved correctly.

Ignoring account lifecycle steps

✗ AVOID

Only testing sign-up, but forgetting to test what happens when an admin needs to delete or update that user's details.

✓ INSTEAD

Always pair your auth tests with management tools. After using `auth_otp_signup_email`, follow up by calling `mgmt_update_user` to ensure the record can be modified.

Mixing credential types

✗ AVOID

Writing a single test script that tries to use both an OAuth code and a password login simultaneously, leading to confusing failure modes.

✓ INSTEAD

Isolate your tests. Run the entire Google flow using `auth_oauth_authorize` in one session, then start a clean, separate session for testing passwords with `auth_password_signin`.

The Right Fit

Use this MCP if you need to validate any user identity pathway—OAuth, SMS OTP, Magic Links, or passwords. It's your go-to tool when the success of your application depends on a complex, multi-step authentication journey. Don't use it if you only need simple data retrieval; for that, basic CRUD tools might suffice. If your problem is solely managing user database records without worrying about login

logic, stick to basic management APIs. However, because you need to confirm the *active session* status after a sign-in (which requires `auth_oauth_exchange` or `auth_otp_verify_email`), this MCP provides the necessary depth and flow control.

Descope (Auth Platform) MCP for AI Agents: Mastering User Authentication Workflows

Building an identity system means manually testing dozens of edge cases: what if a user signs up with Google, but then their access token expires? What happens when they try to reset a password using a different method than expected? Developers spend hours clicking through test accounts and copy-pasting logs just to prove the basic sign-in loop works.

With this MCP, your agent handles it all. You prompt for 'Verify the account lifecycle.' The system runs `auth_password_signup`, simulates the user logging in with a password (`auth_password_signin`), and then confirms session integrity using management tools like `mgmt_get_user`. You get instant, actionable confirmation that every path works.

Descope (Auth Platform) MCP for AI Agents: Managing User Credentials in Development

The biggest manual pain point is the credential sprawl. Every time you change a permission or add a new feature, you have to manually create and test specific accounts with different roles—a process that's slow and prone to human error.

This MCP lets your agent manage all user identities programmatically. You can use `mgmt_create_role` to define 'Premium User' access and then instantly provision dozens of test users using `mgmt_create_user`, allowing you to validate permissions across the entire platform in minutes.

Descope (Auth Platform): 25 Auth & User Management Tools

Use these tools to programmatically manage users, roles, permissions, and execute every type of sign-up or sign-in flow your application requires.

#	TOOL	DESCRIPTION
01	`mgmt_create_access_key`	Creates a new, unique access key for machine-to-machine operations.
02	`mgmt_create_permission`	Defines and creates a specific permission within the platform's role structure.
03	`mgmt_create_role`	Builds a new user role, grouping necessary permissions together.
04	`mgmt_create_tenant`	Initializes and sets up an entirely new isolated tenant environment for testing.
05	`mgmt_create_user`	Programmatically creates a new user account within the system.
06	`mgmt_delete_user`	Removes an existing user account from your environment.
07	`auth_enchantedlink_poll`	Checks if a session created by an Enchanted Link has been completed successfully.
08	`auth_enchantedlink_signup`	Signs up a user using the secure, one-time use Enchanted Link method.
09	`auth_enchantedlink_verify`	Validates an existing Enchanted Link token to confirm a session's legitimacy.
10	`auth_exchange_access_key`	Trades an access key for a temporary, active JSON Web Token (JWT) session.
11	`auth_get_keys`	Retrieves the public keys necessary to validate session tokens across your system.
12	`mgmt_get_user`	Loads and retrieves a user's details using their unique login ID.
13	`mgmt_list_tenants`	Lists all the tenant environments currently set up in your project.
14	`auth_magiclink_signup_email`	Signs a user into or signs them up using a unique Magic Link sent to their email.

#	TOOL	DESCRIPTION
15	auth_magiclink_verify	Validates the token received from a Magic Link, confirming successful access.
16	auth_oauth_authorize	Starts the entire OAuth flow process with an external provider like Google.
17	auth_oauth_exchange	Swaps a code obtained from an OAuth redirect for a permanent session JWT.
18	auth_otp_signin_email	Signs in a user who provides an email-based One-Time Password (OTP).
19	auth_otp_signup_email	Registers a new user account using a dedicated Email OTP code.
20	auth_otp_signup_sms	Creates a brand-new user account using a One-Time Password sent via SMS text message.
21	auth_otp_signup_voice	Registers a new user by verifying identity through a voice-based OTP system.
22	auth_otp_verify_email	Validates an existing Email OTP code to authenticate the current user session.
23	auth_otp_verify_sms	Confirms a user's identity by verifying an SMS-delivered One-Time Password.
24	auth_otp_verify_voice	Verifies the authenticity of a Voice OTP code submitted during a session.
25	auth_password_reset	Initiates and completes the process for resetting a user's forgotten password.
26	auth_password_signin	Authenticates a user who signs in using their traditional username and password combination.
27	auth_password_signup	Registers a new account by having the user set up a unique, strong password.
28	mgmt_search_users	Search for users
29	mgmt_update_user	Update an existing user
30	auth_webauthn_signin_finish	Finish WebAuthn sign in
31	auth_webauthn_signin_start	Start WebAuthn sign in
32	auth_webauthn_signup_finish	Finish WebAuthn sign up
33	auth_webauthn_signup_start	Start WebAuthn sign up

See It in Action

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

U I need to simulate a full user onboarding process. Start with an SMS sign-up and then use that account to list their current details.

User Onboarding Simulation:

1. **Signup initiated.** A unique OTP has been sent to the provided phone number.
2. **Verification pending.** Please provide the 6-digit code when prompted.
3. **Success!** User `test_user@example.com` is now active.

System Output: The user profile was successfully created with a default 'Basic' role and an active session JWT has been generated. You can now query their details using the system management tools.

U Start a login flow for my client, John Smith, using his Google account credentials.

OAuth Flow Initiated (Google):

Please complete the authentication at this URL. Once redirected back, I will exchange the code.

System Output: Authentication successful. The user session JWT has been generated and is valid for 2 hours. We confirmed access rights via `auth_oauth_exchange`.

U I need to test what happens if a user tries to reset their password but only remembers the old username.

Password Reset Procedure:

The system has initiated the password reset flow for 'john.smith@test.com'. A temporary verification link was sent via email.

Please click the link and set your new password. I've verified that this action successfully triggers a session renewal.

Frequently Asked Questions

01 **How do I test the entire user sign-up and login cycle with Descope MCP?**

You can initiate a full flow using your agent. For example, ask it to run an OAuth process; the system handles the redirects, code exchange, and session creation so you get confirmation that every step worked.

02 **Can Descope MCP handle testing multiple account types (e.g., Google vs SMS) in one project?**

Yes. Because it exposes tools for all methods—from `auth_otp_signup_sms` to `auth_oauth_authorize`—you can switch between different authentication mechanisms easily within your agent commands.

03 **Does Descope MCP help me manage user roles and permissions?**

It does. Beyond just logging in, you can use management tools to create new system roles (`mgmt_create_role`) and assign them to test users before running a login check.

04 **What if my app uses custom OAuth providers not listed?**

While the MCP handles major providers, you must use it for flow testing. For highly unique flows, you'll need to combine multiple tools like `auth_oauth_authorize` and `auth_oauth_exchange` to simulate the required steps.

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 `"descope-auth-platform": { "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

Descope (Auth Platform) 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 Descope (Auth Platform). 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	Descope (Auth Platform) MCP
Server ID	`019e3889-45de-73b1-8f1f-cc619031829e`
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/descope-auth-platform.

© 2026 Vinkius · vinkius.com · support@vinkius.com