MCP SERVER | NO CODE | CLOUD HOSTED

Design Token Transformer MCP for AI Agents

Automating Multi-Platform Styling and Color System Generation

Design Token Transformer converts and generates consistent design tokens for every platform you use. It handles the complex process of moving values between syntaxes like JSON, CSS variables, SCSS, and Tailwind, resolving aliases along the way. You can also generate full color scales or create specific dark mode themes instantly.

B Quality Score 85/100

design-tokens | css | tailwind | scss | android | ios

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JULY 2026
vinkius.com/mcp/design-token-transformer

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.

Design Token Transformer MCP

0 tools available
Cloud-hosted on Vinkius

Working across multiple platforms means juggling dozens of styling systems. This MCP solves that headache by letting your AI client manage design tokens everywhere—from Figma definitions to live CSS variables. It takes your core design values and converts them, resolving any internal aliases so you don't have conflicting styles. Need a dark mode theme? It generates the right colors using OKLCH lightness for perfect visual consistency. Plus, if you just give it three anchor colors, it spits out a full 50-950 color weight scale you can drop straight into your system. Vinkius hosts this MCP so you connect once through any compatible client and gain instant access to reliable token conversion every time.

Core Capabilities

01 — Convert tokens between syntaxes

You pass in design definitions in one format (like JSON) and get the exact corresponding code structure in another, such as CSS or Tailwind.

02 — Generate dark mode themes

The MCP creates a full set of color tokens optimized for dark mode by adjusting values using OKLCH lightness standards.

03 — Build comprehensive color scales

It generates an entire 10-step, weighted color scale (50 through 950) based only on three initial anchor colors you provide.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/design-token-transformer** — connect your AI agent in three steps.

01 Start by providing the source design tokens and specifying the target output format. For example, inputting a set of JSON definitions and requesting CSS.

02 The MCP processes your request, resolving all internal aliases to ensure every token points to its absolute primitive value before conversion begins.

03 Your AI client receives clean, structured code—be it SCSS, Tailwind classes, or standard CSS variables—ready for immediate implementation in your project.

The bottom line is you define the tokens once and get usable, multi-format code across all necessary platforms.

Built For

This MCP is essential for senior frontend developers, design system engineers, and UI/UX product managers. If your team spends more time translating color values between JSON files, CSS variables, and Sass, this tool saves you hours of tedious, error-prone work.

Design System Engineer

They use the MCP to maintain token consistency across multiple frameworks, ensuring that a primary color defined in Figma translates correctly into both Tailwind classes and native CSS variables.

Frontend Developer

When implementing a new feature or migrating an old component, they rely on this MCP to quickly generate full color scales or dark mode tokens without manually calculating every shade and variable.

Product Designer

They use it to validate token structures and test how three core brand colors will behave when expanded into a complete, usable 50-950 scale for component states.

What Changes When You Connect

01 Stop manually converting tokens. Use `transform_tokens` to instantly get your design definitions in CSS, SCSS, or Tailwind syntax.

02 Maintain perfect brand consistency using `generate_color_scale`. You only need three anchor colors to produce a full 50-950 palette.

03 Implement true dark mode without guesswork. `generate_dark_mode_tokens` handles the complex OKLCH lightness adjustments for you.

04 Reduce build friction by resolving aliases automatically. The MCP ensures your tokens are always pointing to their simplest, most reliable primitive value.

05 Save time debugging mismatched styles. This MCP treats design token conversion as a single action, minimizing manual cross-platform checks.

Real-World Applications

Migrating an old component library

The developer needs to move a legacy component that used hardcoded colors into a modern token system. They use the MCP to take the existing JSON color definitions and generate clean, structured CSS variables for immediate adoption.

Launching dark mode support

The product manager asks their agent to prepare the site for a dark theme. The agent calls `generate_dark_mode_tokens` using the main brand palette, delivering all necessary token adjustments optimized for visual comfort.

Establishing a new color system

The design team has three core shades (primary, secondary, accent). They ask their agent to run `generate_color_scale`, receiving the full 50-950 spectrum needed for every component state across the whole application.

Cross-platform token syncing

A designer updates a primary color in Figma. They ask their agent to use `transform_tokens` to update both the SCSS variables and the Tailwind utility classes simultaneously, ensuring zero discrepancies across web and mobile.

Patterns to Avoid

Manual token translation

✗ AVOID

A developer manually copies a color value from a JSON file into an SCSS variable, then has to re-check if that same value works correctly in the Tailwind config. This process is slow and prone to human error.

✓ INSTEAD

Instead, use `transform_tokens` within your agent. You feed the source tokens once, and the MCP delivers perfectly formatted, multi-syntax code for both SCSS and Tailwind automatically.

Ignoring alias resolution

✗ AVOID

The team defines a token 'color-text' that references another internal alias 'color-gray-700'. If the reference is outdated or incomplete, the compiled CSS will break, leaving parts of the UI unstyled.

✓ INSTEAD

Always let the MCP run the transformation. The underlying process resolves all nested aliases to their primitive values first, guaranteeing clean and predictable output.

Building dark mode manually

✗ AVOID

Attempting to calculate complementary colors for a dark theme by hand is difficult because simple inversions don't account for perceived lightness (OKLCH). The result often looks flat or too bright.

✓ INSTEAD

Use `generate_dark_mode_tokens`. It applies specialized color adjustments based on advanced lightness metrics, delivering professional-grade, visually consistent dark mode tokens.

The Right Fit

Use this MCP if your primary bottleneck is the *translation* or *expansion* of design definitions. Specifically, if you need to move a core set of values from one styling language (JSON) into several others (CSS, Tailwind, SCSS), or if you are building out a full color system for the first time. It’s critical when consistency across platforms is non-negotiable.

Don't use this MCP if your problem is purely conceptual—for example, if you just need to *decide* on a primary color before writing any code. For that, you might need a specialized brainstorming tool. Also, don't try to use it for complex logic (e.g., 'if the user is premium, make it red'). This MCP handles static token conversion and generation only; it doesn’t run conditional UI logic.

Design Token Transformer: Solving Multi-Format Styling Issues

Today, when you update a core brand color, the process is manual and tedious. You define it in JSON for your design system documentation; then a developer has to copy that hexadecimal code into the CSS variables file, remembering to adjust it if Tailwind requires a different format. This constant copy-pasting and cross-referencing across multiple files guarantees discrepancies.

With this MCP, you simply tell your agent the source tokens and the target formats (CSS, SCSS, Tailwind). The system handles the entire conversion process in one shot, delivering clean, usable code snippets for all platforms. You get instant token synchronization.

Design Token Transformer: Generating Full Color Palettes

Before this MCP, creating a full color scale was guesswork. If you picked three colors to represent your brand's depth, you had no standardized way

Now, give the tool three anchor colors. It outputs a complete, mathematically consistent color scale

to derive the necessary 50-950 shades for subtle backgrounds or hover states without spending hours in an online palette generator.

that spans the entire range needed by your design system. The whole palette is ready when you are.

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 convert these primary brand tokens from JSON into CSS variables and Tailwind classes.

Here are your converted tokens:

CSS Variables:

```css
:root {
--color-primary: #007bff;
}
```

`Tailwind Classes:`

- bg-primary : bg-[#007bff]
- text-primary : text-[#007bff]`

Remember to update your config file for full integration.

U Generate a dark mode theme based on this light mode JSON: {"background": "#ffffff", "surface": "#f8f9fa"}.

Dark Mode Tokens Generated:

Based on your input, here are the suggested adjustments for low-light environments:

- `--color-background` : `#121212` (Deep background)
- `--color-surface` : `#1e1e1e` (Component surface)

These values maintain high contrast and readability.

U Give me a full color scale using #f8f9fa, #6c757d, and #212529 as anchors.

Generated 50-950 Scale:

The following palette provides full weight coverage for your brand:

Weight	Hex Code
50	`#f8f9fa`
700	`#495057`
950	`#1a1d20`

This range covers everything from lightest backgrounds to deepest text.

Frequently Asked Questions

01 **How does the Design Token Transformer MCP help with CSS variables?**

It automatically takes your defined tokens and outputs them in standard CSS variable format. This saves you from manually writing out `:root { --token: value; }` for every single color or spacing unit.

02 **Can the Design Token Transformer MCP handle dark mode changes?**

Yes, it's built specifically for that. You give it your light mode tokens, and it calculates the necessary contrast adjustments using advanced lightness metrics to make sure the dark theme looks right everywhere.

03 **What if I only have a few colors? Can I build a full palette?**

Absolutely. You don't need dozens of hex codes. Just provide three key anchor shades, and the MCP will generate a complete 50-950 scale, giving you every shade needed for component states.

04 **Is the Design Token Transformer MCP better than copying tokens manually?**

Yes. Manual copying fails when your token structure changes or if you have aliases. This MCP resolves those internal references automatically, guaranteeing that what you paste into Tailwind is identical to what's in your CSS variables.

05 **Does the Design Token Transformer MCP support multiple formats?**

It supports several major web standards. You can convert tokens between JSON, CSS, SCSS, and utility frameworks like Tailwind, making it a central hub for token management.

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 `"design-token-transformer": { "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

Design Token Transformer 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 Design Token Transformer. 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	July 2026
MCP Server	Design Token Transformer MCP
Server ID	`019f21a7-c042-72d8-98aa-8d4ebf17cc4f`
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/design-token-transformer.