

KD Gated Proxy — API Key Provider & Role-Based SDK Config Variation

Issue 3 · Suggestion on the gated proxy of the API-key provider Bug finder & solution author: ctaxnagomi (Applied Agentic AI Engineer · Olympic Dancer 1963) Environment: Desktop · Windows 11 · Dual-boot with WSL2

0. Provenance & Disclaimer

Sources read for this issue:

- [RFI/KD Agentic LiteLLM DeepSWE Whitepaper Proposal ctaxnagomi.pdf](#) (v1.0, 16 Sep 2026)
- [RFI/KD Agentic Execution Stack Whitepaper COMPACT.pdf](#)
- [RFI/kd-ade-kit-architecture.pdf](#) (KD-ADE Kit architecture reference, 23 Sep 2026)
- [RFI/CLAUDE SDK CONFIG \(AGENTIC\).pdf](#) (Anthropic SDK config sample)
- [RFI/KRACKEDDEVS_LITELLM_PROXY_CONFIG.md](#) (+ [.docx](#))
- [RFI/SDK_CONFIG_curl_python_node.md](#)
- [RFI/SKILLS.md](#)
- [RFI/litellm_universal_agentic_sample.ts](#) , [RFI/litellm_universal_sdk_sample.ts](#)
- [RFI/build_docx.py](#) , [RFI/universal_gated-proxy_tools.txt](#)
- [RFI/bugreportkd.zip](#) → [BUGS_REPORT.pdf](#) (8 bugs), [BUGS REPORT 160926_COMPACT.pdf](#) , [kd-labs_permission_request_letter.pdf](#)
- [RFI/uggestionuniversalgatedproxysettingstoolcallingtok.zip](#)

Two images ([azure.litellm.PNG](#) , [IMG_0696.PNG](#)) were present in the RFI folder but could **not** be read by the drafting agent (no image input). The Azure/LiteLLM provider-config reference and the second screenshot are therefore cited by filename only and should be attached to the final packet.

This is a **proxy-admin suggestion**, not a per-app request and not a bug report. It is unofficial and provided as-is; production authority stays with the LiteLLM/KD proxy owner.

1. The Core Comment — Why the Current Pipeline Is Incomplete

The key-usage portal documents a single minimal recipe: *"this is an OpenAI-compatible endpoint; change two settings — base URL and API key — and anything that talks to OpenAI works."* That is correct but **insufficient**. It states the minimum for a plain chat call and stays **unclear on the use case of variation of separate configs**.

In daily practice the SDK config is **not one config**. It must vary along four independent axes:

Axis	Variation	Why it matters
Role / privilege	User · Admin (gated proxy) · Owner (rented upstream streamline)	Access, allowed params and allowed endpoints differ per role
Upstream provider SDK	OpenAI-compatible · Anthropic (Claude) · DeepSeek · Azure	Request/response shape and tool schema differ per provider
Capability	Agentic enable · agentic tools · <code>tool_calls</code> · <code>specific tool_calls</code>	A chat-only config silently disables the agentic surface
Output budget	min/max output per model class	Token efficiency, especially image-gen, video-gen, high-reasoning

An `import` of the OpenAI SDK is really a function of **access and privilege** — whether the caller is a User, an admin of the gated proxy, or the owner of the rented stream of a given AI model-service provider. The portal does not model that. **Currently it is limiting the capabilities** the same key could expose.

2. The Role / Privilege Model (missing today)

The gateway should expose **profiles**, not one global default:

Profile	Who	Allowed endpoints	Allowed params	Tool scope
User (member)	Any gated-key holder	chat, images	<code>tool_choice:auto</code> , temp, per-model <code>max_tokens</code>	reads (web_fetch) + write_file
Admin (gated proxy)	Proxy operator	+ <code>/v1/models</code> , video, guardrail config	<code>drop_params</code> , <code>force_tool_use</code> , per-model caps, routing	full catalogue + tool permissions
Owner (rented streamline)	Provider-stream owner	upstream direct (D. reference)	all provider-native params, <code>provider_specific_fields</code> passthrough	MCP/service routing, audit

Without this split, a user gets admin-level silence about tools, and an owner gets user-level truncation. The capability ceiling is set by the **lowest** common config — which is the current symptom.

3. Per-Provider SDK Config Variation

Same gateway, four request shapes. The portal shows one; the config should ship all.

Provider family	SDK / base	Agentic shape	Key preservation needs
OpenAI-compatible (LiteLLM default)	<code>openai</code> SDK, <code>baseURL</code> → proxy	<code>tools[]</code> , <code>tool_choice</code> , <code>max_tokens</code> , structured content parts	<code>message.tool_calls</code> , <code>content</code> arrays, full error body
DeepSeek (V4 Flash)	OpenAI-compatible via proxy	tools + <code>tool_choice:auto</code>	<code>provider_specific_fields.reasoning_content</code>
Anthropic (Claude)	<code>@anthropic-ai/sdk</code> , <code>messages.create</code>	<code>tools[]</code> , <code>tool_use</code> blocks, <code>tool_result</code> blocks, <code>code_execution_*</code>	<code>stop_reason:"tool_use"</code> , image blocks (<code>source.base64</code>)
Azure (LiteLLM-proxied)	OpenAI-compatible + Azure upstream	same as OpenAI-compat	Azure deployment-name mapping (see <code>azure.litellm.PNG</code>)

Reference already in the packet: `CLAUDE SDK CONFIG (AGENTIC).pdf` shows the Anthropic-native variant — `client.messages.create({ model, max_tokens, tools, messages })` , a tool-result loop on `stop_reason ≡ "tool_use"` , and a **browser-safety note** ("Never ship a real API key to the browser. Point baseURL at your own backend proxy"). That is the correct pattern and should be a first-class provider profile, not a side note.

4. Agentic Enablement — `tool_calls` and *specific* `tool_calls`

The portal documents the chat surface and leaves tool semantics to defaults. The config must explicitly allow:

- **Agentic enable** — the loop is client-side; the proxy relays `tool_calls` unmodified. Evidence: live test returned HTTP 200, `finish_reason: "tool_calls"` , correct `tool_calls[].function` payload, with `tools` + `tool_choice:auto` forwarded intact.
- **Agentic tools** — schema-first definitions (`write_file` , `web_fetch` , `apify_browse` , `apify_search` , `neo_open` , `neo_snapshot` , `neo_act` , `preview_capture`).
- `tool_calls` — preserved before any response transformer runs.
- **Specific `tool_calls`** — per-persona control: `auto` (build), a named function when the turn demands one, and — critically — **never** `force_tool_use: "none"` , which silently disables the whole surface.
- **Parallel tool calls** — allowed (the ADE build persona allows ~8 files per cycle).

Currently the surface exists but isn't advertised or guaranteed by profile, so clients guess and regress to plain chat.

5. Min/Max Output Tuning — Token Efficiency

This is the "high reasoning / image / video" part the portal omits.

Model-class budgets

Class	Min sensible	Max to allow	Note
Chat / ask	8192	8192	plan/ask personas
Code build (multi-file)	16384	16384+	raising 8192 → 16384 removes the truncation crash
High-reasoning models	per provider	provider max	reasoning tokens must be preserved, not stripped
Image (<code>gpt-image-2</code>)	—	quality-gated	iterate low , render final high once
Video (<code>/v1/videos</code>)	—	resolution/duration-gated	async; keep prompts short

Rules

- **Per-model, not flat.** A single flat `max_tokens` starves builds and wastes budget on short asks.
- **Truncation is the real limiter.** A completed-then-cut `write_file` argument JSON surfaces as *"Unterminated string in JSON"*. Fix at both layers: client safe-parse (recover longest valid prefix) **and** proxy per-model `max_tokens ≥ 16384` .
- **Image/video cost is driven by quality/resolution/duration, not prompt length** — so gate by those, iterate cheap, render final once.
- **Reasoning budget is a config field**, not a by-product: preserve `reasoning_content` / provider-specific thinking fields or the model's chain-of-thought is silently dropped and quality nosedives.

6. What Is Currently Limiting Capabilities (consolidated)

#	Limiter	Layer	Fix
1	Single global config; no role/provider profiles	Admin	\$2/\$3 profiles
2	Flat <code>max_tokens</code> ; builds truncate	Client + Admin	per-persona + per-model caps ≥ 16384
3	<code>tool_calls</code> can be stripped by response middleware	Admin	<code>force_tool_use: "auto"</code> , no transformer filtering
4	<code>reasoning_content</code> / provider fields dropped	Admin	preserve provider-specific fields
5	Multimodal <code>content</code> arrays collapsed to string	Admin	preserve arrays

#	Limiter	Layer	Fix
6	Video endpoint not exposed on the key	Admin	assign a video model; verify <code>GET /v1/models</code>
7	Destructive client-side keyword guardrail destroys valid output	Client	non-destructive banner + server-side (pre_call) enforcement
8	<code>drop_params</code> default drops legitimate OpenAI params	Admin	<code>drop_params: false</code>
9	Errors swallowed (partial bodies)	Admin	relay full upstream error JSON

7. GRILL-ME — Plan Interrogation (adapted)

Protocol per `GRILL-ME.md` : challenge the plan, surface hidden assumptions, resolve dependencies one-by-one, give a recommended answer for each. Adapted from the DeckerGUI layer model to the KD gated-proxy context.

Q1 — What problem does this solve, and who benefits?

Recommended: It converts a *minimum viable* gateway into a *profile-aware* gateway. Beneficiaries: **User** (higher first-pass quality, fewer tokens), **Admin/owner** (predictable cost, governance, audit), **KD** (an evidence-backed agentic product tier). Without it, every advanced workflow regresses to plain chat.

Q2 — Which layer does this live in?

Recommended: Multi-layer, matching the whitepaper's split —

- **Gateway policy (Admin):** `drop_params` , `force_tool_use` , per-model caps, response passthrough, routing, budget.
- **Agent harness (user-local):** tool loop, safe-parse, context reduction, capture.
- **Execution boundary (owner):** authorization, isolation, audit — **not** lexical filters. Keep guardrails as a layered control chain, never a single regex.

Q3 — What's the data flow?

Recommended: `User → KD Agent/Harness → LiteLLM Proxy (routing · budget · policy · log) → Model → tool_call → Policy/Authorization → MCP/Sandbox → Tool Result → Validate → Answer` . Each hop has an owner; configuration is the independent variable.

Q4 — What can fail, and what's the graceful path?

Recommended failure→degradation map:

- Truncated tool args → safe-parse longest prefix, re-request once.
- Model name wrong → `GET /v1/models` , exact match.
- Chat model sent to image/video → route by capability table.

- Budget exceeded → does not auto-reset; surface to admin, don't tight-loop.
- Video unassigned → report "no video model on this key", don't retry.
- Proxy strips fields → D. upstream-reference comparison detects gateway transformations.

Q5 — What's the cost model?

Recommended: Measure **cost per verified successful task**, not per call. Simulated planning targets (explicitly *not* measured): A Normal 45–55% / \$0.70–1.00 / 170–220K tok; B Standard Agentic 52–62% / \$0.60–0.90 / 145–190K; C Universal 60–70% / \$0.45–0.75 / 105–155K. Iterate image/video cheap, render final once.

Q6 — How do we measure success?

Recommended: Pass@1/reward, partial score, cost/task, tokens/task (reasoning vs tool separated), agent steps, tool-success rate, human intervention, latency, failure taxonomy. External benchmark **DeepSWE v1.1** (113 long-horizon SE tasks); internal 10-task smoke suite first. Freeze model IDs, proxy version, sampling and token accounting **before** measuring.

Q7 — What's the rollback plan?

Recommended: Configuration is versioned (Person-B config as a versioned artifact). Rollback = flip to the frozen baseline config; no data migration. Keep the D. upstream-reference path available to prove the gateway change caused any delta. Feature-flag per profile so a bad profile doesn't take down all keys.

Residual open questions

1. Exact pinned dated IDs for all models under test (DeepSeek V4 Flash, GPT-5.5/5.4-mini/5.4-nano/5.6-sol/5.6-terra/Astra).
2. Whether a video model will be assigned to the key, or video stays out of scope.
3. Which fields the proxy's middleware currently transforms (only a D. direct-upstream diff settles this).
4. Whether guardrails move fully to `pre_call`, or a narrow lexical first-line signal is retained.
5. **Simulated numbers must never be published as measured** — governance rule.

Resolved decision tree (GRILL-ME output format)

- **Problem Statement:** One minimal config limits a heterogeneous, role- and provider-varied workload.
- **Layer Assignment:** Multi-layer (gateway policy · agent harness · execution boundary).
- **Data Flow:** `User → Harness → LiteLLM → Model → tool_call → Policy → Sandbox → Validate → Answer`.
- **Failure Modes:** truncation, wrong model, wrong endpoint class, budget cap, missing video model, field stripping — each with a defined degradation path.
- **Cost Estimate:** track cost per verified success; iterate cheap, render final.
- **Success Metrics:** DeepSWE Pass@1 + cost/token/latency/tool/failure metrics; internal smoke suite first.
- **Rollback Plan:** versioned configs; flip to baseline; upstream-reference diff; per-profile flags.

- **Open Questions:** pinned IDs, video scope, middleware transforms, guardrail placement, no-simulated-claims rule.

8. Recommendations to the Proxy Admin (checklist)

- ☐ Introduce **role profiles** (User / Admin / Owner) instead of one global default.
- ☐ Ship **per-provider SDK config profiles** (OpenAI-compatible, DeepSeek, Anthropic, Azure) — include the Anthropic-native `tool_use` loop already drafted in `CLAUDE SDK CONFIG (AGENTIC).pdf`.
- ☐ `force_tool_use: "auto"` — never `"none"`.
- ☐ Per-model `max_tokens ≥ 16384` for code-generation models.
- ☐ Prefix/suffix output budget control **per model class** (chat, build, reasoning, image, video).
- ☐ Preserve `message.tool_calls`, `provider_specific_fields.reasoning_content`, and multimodal `content` arrays.
- ☐ `drop_params: false`; relay full upstream error bodies.
- ☐ Expose `/v1/images/generations`, `/v1/images/edits`, `/v1/videos` (assign a video model first).
- ☐ Run guardrails at `pre_call`, not as destructive output filters.
- ☐ Enforce authorization/isolation at the **execution boundary** (sandbox/micro-VM), not via regex.
- ☐ Adopt the A/B → D. upstream-reference → DeepSWE methodology; publish **measured only**.

9. Worked Examples — How ChatGPT, Claude & Grok Gate the Same Proxy

The pattern KD needs already runs at scale. OpenAI (ChatGPT), Anthropic (Claude) and xAI (Grok) each run **one backend behind a gated proxy whose config varies by tier** — free → paid → max → team → enterprise. The examples below are illustrative of the *stable gating pattern* (not live price/limit figures — those change and must be pinned at time of use).

9.1 OpenAI — ChatGPT (Free · Plus · Pro · Team · Enterprise)

Tier	What the gate changes
Free	Base model, short context, tight message rate limits, standard output cap, limited tool set
Plus	Higher-capability models, higher message caps, advanced tools (deep research, image gen, voice)
Pro	Top models + extended reasoning budget ("pro" mode = more compute/thinking), highest message caps, early access
Team	Shared workspace + admin, data excluded from training, collaboration controls
Enterprise / Edu	SSO/SCIM, admin console + audit logs, data residency, custom retention, connectors/DLP, compliance

The gate varies: model access · reasoning budget · rate limits · tool availability · governance — the same axes as KD §2.

9.2 Anthropic — Claude (Free · Pro · Max · Team · Enterprise)

Tier	What the gate changes
Free	Base model, lowest usage
Pro	All current models, higher usage, Projects, integrations, Claude Code
Max (5× / 20×)	The gate is a usage multiplier — more requests, priority, higher effective throughput
Team	Shared workspace, admin, higher per-seat allowance
Enterprise	SSO/SAML, SCIM, audit + compliance API, custom retention/data residency, expanded context, purchase via Bedrock/Vertex
API (separate gate)	Workspace-scoped keys, tiered rate limits, extended thinking, MCP connectors

The gate varies: a *multiplier* on budget/rate plus enterprise governance — the KD "Universal" tier analogue.

9.3 xAI — Grok (Free · Premium · SuperGrok · Heavy · API/Enterprise)

Tier	What the gate changes
Free / X Premium	Grok access gated by X subscription level; base quotas
SuperGrok / higher	Better models, higher quotas, DeepSearch, Think/reasoning
Grok Heavy	Multi-agent orchestration gated as a premium capability
API / Enterprise	Tiered keys, function calling, structured outputs, live search, reasoning; enterprise terms

The gate varies: capability-tier gating (reasoning, multi-agent) — the model for KD **Agentic vs Universal**.

9.4 What all three share (the reusable pattern)

- 1. **One gateway, many profiles** — same backend, config differs by tier.
- 2. **Model access is gated by tier** (cheapest model on the cheap tier).
- 3. **Reasoning/effort is a gated budget** — Pro / Max / Heavy = more compute.
- 4. **Output & rate limits scale with tier.**
- 5. **Tools/agents are tier-gated** (deep research, Claude Code, Grok Heavy).
- 6. **Enterprise = governance, not more raw capability** (SSO, SCIM, audit, residency, retention).
- 7. **The API is a separate gate** with scoped keys and usage tiers.

9.5 Map to KD

Provider ladder	KD profile / tier
Free / basic	KD Standard (User profile)
Plus / Pro / Max	KD Agentic, KD Universal
Team	KD Universal (team)

Provider ladder	KD profile / tier
Enterprise	KD Enterprise (isolation, audit, custom MCP)
Provider API	KD gated proxy (User / Admin / Owner roles — \$2)

Lesson: don't invent a new ladder — mirror the proven one. The gated proxy's job is to enforce exactly these seven gates, **per role** (\$2) and **per provider SDK** (\$3). That is what turns "two settings and it works" into a real, tiered product.

10. References

1. KD Agentic Execution Stack — LiteLLM, Tool Calling, Harness Optimization & DeepSWE (ctaxnagomi, v1.0, 16 Sep 2026).
2. KD-ADE Kit Architecture Reference (ctaxnagomi et al., 23 Sep 2026).
3. KRACKED DEVS ADE KIT — Bug Report: LiteLLM Integration Audit (BUG-1...BUG-8).
4. CLAUDE SDK CONFIG (AGENTIC) — Anthropic SDK sample.
5. LiteLLM docs — supported_endpoints · function_call · drop_params · input (tool_choice/max_tokens) · reasoning_content · image_generation · image_edits · videos · guardrails.
6. DataCurve AI — DeepSWE v1.1 (113 long-horizon SE tasks).
7. **SKILLS.md** — KD gated-proxy agent skill (attach to any agent session).

Prepared by ctaxnagomi · Applied Agentic AI Engineer · Olympic Dancer 1963 · KrackedDevs / KD AI Infrastructure. Suggestion only — production authority remains with the LiteLLM/KD proxy owner.