DEFENSIVE PUBLICATION — TECHNICAL DISCLOSURE (TDCOMMONS DEPOSIT COPY) · 2026-06-25

Keywords: defensive-publication, prior-art, ai-agents, llm-gateway, fail-closed, multi-provider-routing, llm-as-judge, reliability

Per-Route Fail-Closed Multi-Provider LLM Routing

A Technical Defensive Publication establishing public prior art

Subtitle: One mandatory, provider-agnostic LLM bridge in which the fail-closed-versus-fail-open failure posture is bound to the *semantic route class* — governance/judge routes terminate on a model outage rather than silently degrading, while general routes fail over through a database-driven fallback chain that itself degrades to a hardcoded array.

Field	Value
Publisher / Copyright holder	Gus IT LLC (Florida, USA)
Author	Gustavo Assuncao, PhD
Publication date	2026-06-25
Version	1.0
Document type	Technical Defensive Publication (public prior art)
Classification	Public
License	AGPL-3.0-or-later (copyleft; commercial license available)
Deposit channel	To be assigned (IP.com / Zenodo / arXiv) — establishes a public, dated, citable prior-art record.
Field	Multi-provider LLM gateway routing for platforms with high-stakes governance decisions
Tags	LLM, routing, reliability, fail-closed, governance, LLM-as-judge, fallback

Abstract

Modern AI platforms funnel all inference through a single multi-provider gateway so that provider credentials, model allow-lists, cost telemetry, and policy resolve in one place. The default reliability posture of such a gateway is to **fail open**: if the requested model/provider is unreachable, the gateway silently substitutes a cheaper or different model so the call still returns. For general chat this is correct — availability beats fidelity. For **high-stakes governance decisions** (an LLM-as-judge that certifies mastery, scores an assessment, or gates a release) silently re-basing the decision onto an unvalidated, cheaper, or differently-aligned model is a **correctness/safety defect**, not graceful degradation.

This publication discloses a routing mechanism in which the **fail-closed-versus-fail-open decision is bound to the semantic *route class*, not to the requesting caller**. Each request bears a route identifier selecting a stored route descriptor that declares a provider, a default model, an allowed-model list, and a boolean fail-closed flag. Fail-closed routes construct a candidate chain consisting of **only** the requested provider+model, **bypass** any local-inference/GPU-queue path, **refuse** any result resolving to a *different* provider (provider-identity mismatch), and write an **immutable audit record** of every denial — terminating rather than substituting. Fail-open routes build an ordered, cross-provider fallback chain

read from a **database table** that **itself degrades** to an in-code hardcoded array when the database is unreachable — making the routing layer resilient to the failure of its own configuration store. The combination lets governance and general traffic share one provider-agnostic bridge with opposite, route-appropriate failure postures. This document is published defensively to keep the technique freely practiceable and unpatentable by others.

---

1. Executive Summary

1.1 Thesis

A single mandatory LLM bridge can — and for safety **must** — carry **two opposite failure postures at once**, selected per request by the *meaning* of the route rather than by the identity of the caller or by a global gateway setting. General traffic should **fail open** for availability; governance/judge traffic must **fail closed** so a high-stakes decision is never silently re-based onto an unvalidated model. The novel contribution is the *binding of that posture to a semantic route class*, combined with a configuration layer that is itself resilient (DB-driven fallback that degrades to a hardcoded array) and a denial path that is **immutable, auditable, and provider-identity-aware**.

1.2 Contributions

#	Contribution	Where
C1	Per-route boolean fail-closed flag **bound to a semantic route class** (governance/judge), not to the caller, on one shared bridge	§6, §7, §14
C2	Fail-closed routes construct a candidate chain of **only** the requested provider+model and **bypass the local-inference/GPU-queue path**	§7.3, §7.4
C3	**Provider-identity-mismatch refusal**: a result that resolves to a provider other than the one requested is refused even when "successful"	§7.5, §11
C4	**Immutable audit record** written on *every* fail-closed denial (both unreachable-tier and identity-mismatch)	§7.6, §8
C5	**DB-driven fallback chains that gracefully degrade to a hardcoded in-code array** when the configuration store is unavailable — config-availability resilience of the routing layer itself	§7.7, §8
C6	A request-time merge that always **places the requested attempt first**, then de-duplicated DB chain entries, preserving caller intent	§7.7
C7	An **old-model-id upgrade** rule that, on fail-closed and fail-open paths alike, upgrades a deprecated tier id to its sanctioned successor — never *downgrades*	§7.8

1.3 Headline claim

> *In a multi-provider LLM gateway through which all callers are funneled, the per-request decision of whether a model outage is terminal (fail closed) or recoverable by substitution (fail open) is determined by the **semantic class of the route** carried on the request — such that a governance/judge route is restricted to a single requested provider+model, bypasses local-inference paths, refuses provider-identity mismatches, and audits every denial, while a general route fails over through a database-driven chain that degrades to a hardcoded array.*

1.4 Scope — what it is / is NOT

It IS: a routing-layer mechanism inside one in-process LLM bridge; a per-route policy primitive; a failure-posture selector keyed to route semantics; an auditable denial path; a self-resilient configuration-read strategy.

It is NOT: a new model, a new judge/scoring algorithm, a RAG reranker, a prompt-composition system, a billing system, or a claim over LLM-as-judge itself. The *decision logic of any judge* and the *role→route mapping policy* are out of scope here (the latter is intentionally withheld as a trade secret — see §15). This publication is scoped strictly to the **multi-provider gateway routing mechanism**.

---

2. Introduction & Motivation

2.1 The concrete problem

A platform that mandates "**all** inference goes through one bridge" (so keys, allow-lists, telemetry, and policy live in one place) gains uniformity but inherits a single, global failure policy. The universal default in production gateways is **fail open**: on an outage, rate-limit, or timeout for the requested model, retry a *different* model/provider so the user still gets an answer. This is the right default for the 95% of traffic that is conversational.

But the same bridge also carries **governance** traffic — calls whose *output is a decision with consequences*:

- An **LLM-as-judge** that certifies whether a learner has achieved mastery and may unlock the next module.
- A **scoring judge** that assigns a formative score to each turn.
- A release-gate judge, a compliance judge, a fairness judge.

For these, "the answer still came back" is **not** success if it came back from the *wrong model*. A mastery certification produced by a cheap fallback model that was never validated for that decision is, at best, noise and, at worst, a silently-wrong high-stakes outcome that no one notices because the gateway "succeeded."

2.2 The "tax" of the naive approaches

Teams confronting this usually pick one of three unsatisfactory options:

1. **Global fail-closed.** Make the whole gateway terminal on outage. *Tax:* every chat message now errors during a provider blip; availability collapses; users churn.
2. **Caller-side guards.** Let each caller re-check which model answered and reject mismatches. *Tax:* the check is duplicated across dozens of call-sites, drifts, and is forgotten exactly where it matters; the gateway still *built* and *spent on* a fallback call before the caller rejected it; there is no central audit.
3. **Two separate gateways.** Run a "safe" gateway and a "fast" gateway. *Tax:* double the keys, double the telemetry, double the allow-list maintenance, and the policy of *which traffic is safe* now lives in routing/deployment config instead of in the request — easy to misroute.

2.3 Why existing gateways fall short

Production multi-provider gateways (LiteLLM, Portkey, Bifrost, Cloudflare AI Gateway, OpenRouter, Solo.io/Gloo AI Gateway) all support **per-route or per-config policy** and most support **ordered fallback**. Some support a *fail-closed/compliance suppression* mode. What none of them ships as a single integrated primitive is the **specific combination**:

- the fail-closed flag **bound to a semantic governance/judge route class** living **on the request**, *and*
- **bypassing the local-inference/GPU path** specifically because it would resolve to a non-requested provider, *and*
- a **provider-identity-mismatch refusal even on an otherwise-successful call**, *and*
- an **immutable audit row on every denial**, *and*
- a fallback-config store that **degrades to a hardcoded array** so the routing layer survives its own config DB being down.

2.4 Why now

Three trends make this acute in 2026: (a) the rise of **LLM-as-judge** for grading, certification, and autonomous gating; (b) **multi-provider** deployments where the "same" capability is served by Anthropic, OpenAI, Google, on-prem vLLM, and a GPU queue, each with materially different alignment; and (c) regulatory pressure (EU AI Act high-risk obligations, audit expectations) that makes a *silently substituted* high-stakes decision a compliance liability. The mechanism here is the minimal, route-local way to get availability where you want it and integrity where you need it on one bridge.

---

3. Problem Statement

3.1 Formal framing

Let a gateway `G` receive requests `q` each bearing a route identifier `r ∈ R`. A route descriptor `D(r) = ⟨provider, defaultModel, allowed, failClosed⟩` is resolved from a configuration source. Let `attempt(p, m)` denote an inference attempt against provider `p` with model `m`, returning either a result `res` (with an *actual* resolved provider `res.provider`) or an error.

We require a routing function `route(q, D(r))` such that:

- For `D(r).failClosed = true`, the *only* admissible successful outcome is `res` with `res.provider = D(r).provider` and `res.model ∈ allowed`. Any other resolution (unreachable, or `res.provider ≠ D(r).provider`) MUST yield a **terminal error** plus a durable denial record. No substitution, no local-inference path.
- For `D(r).failClosed = false`, `route` MAY substitute across an ordered chain `C(r)` to maximize availability; `C(r)` is read from a store `T` and, when `T` is unavailable, from an in-code default `C0(r)`.
- The choice between the two regimes is a pure function of `r` (the route class) — **independent of the caller's identity**.

3.2 Derived requirements (R1..R12)

Req	Requirement	Satisfied in
R1	All callers funnel through **one** in-process bridge; the route is selected per request by a route identifier.	§6, §7.1
R2	A route descriptor declares provider, default model, allowed-model list, and a boolean fail-closed flag, resolved from configuration.	§6, §8
R3	The fail-closed-vs-open decision is **bound to the route class**, not the caller.	§7.2, §14
R4	A fail-closed route's candidate chain is **exactly the requested provider+model** (no fallback entries).	§7.3
R5	A fail-closed route **bypasses** any local-inference / GPU-queue path.	§7.4
R6	On a fail-closed route, an *unreachable* requested tier yields a terminal error (no substitution).	§7.5
R7	On a fail-closed route, a result resolving to a **different provider** is detected and **refused** even if "successful".	§7.5
R8	**Every** fail-closed denial writes an **immutable audit record** (both failure modes).	§7.6, §8
R9	Fail-open routes read an **ordered cross-provider fallback chain from a database table**.	§7.7, §8
R10	When the database is unavailable, the chain **degrades to a hardcoded in-code array** — the routing layer survives its own config store failing.	§7.7, §8
R11	The requested attempt is always placed **first** in any chain; DB entries are appended de-duplicated.	§7.7
R12	A deprecated model id is **upgraded** to its sanctioned successor on all paths — never silently *downgraded*.	§7.8

4. Related Work & Prior Art

This mechanism is built deliberately *on top of* the established multi-provider-gateway art; it is a novel **combination**, not an ex-nihilo invention. The relevant prior sources:

- **LiteLLM (Router / Proxy).** Open-source multi-provider router with DB/config-backed model lists and **ordered fallbacks** (`fallbacks`, `context_window_fallbacks`). Adopted here as the closest analogue for *DB-backed config + ordered cross-provider fallback*. It does **not** bind a fail-closed posture to a *semantic governance route class*, nor refuse a provider-identity mismatch, nor degrade its config store to a hardcoded array as a first-class behavior.
- **Portkey Gateway (Configs / Conditional Routing).** Commercial gateway with **stored per-route configs**, conditional routing, and fallback targets. Adopted as the analogue for *stored per-route policy*. It does not ship *fail-closed-bound-to-judge-route + GPU-path bypass + identity-mismatch refusal + mandatory denial audit* as one primitive.
- **Bifrost (Maxim).** Gateway with a **plugin pipeline** that can suppress/short-circuit fallback ("fail-closed"/compliance plugins). Adopted as the analogue for *plugin-controlled fail-closed*. The fail-closed there is plugin/global, not bound to a request-borne semantic route class with the specific bypass/identity/audit trio.
- **Solo.io / Gloo AI Gateway.** Documents explicit **fail-open vs fail-closed** gateway semantics (e.g. for ext-proc/authz). Adopted as the analogue for *the fail-open/closed vocabulary itself*. It governs the proxy/authz layer, not a per-judge-route LLM model-substitution policy.

- **Cloudflare AI Gateway / OpenRouter.** Hosted multi-provider routing with fallback and provider preferences. Adopted as analogues for *provider-agnostic routing + fallback*. Neither binds fail-closed to a governance route class nor audits identity-mismatch denials.
- `oh-my-openagent` **(and similar role→model matchers).** Maps an agent **role to a model**. Adopted as the analogue for *role-selects-model*. It selects *which* model, not *whether substitution is permitted on outage* per semantic route.
- **Infobip US20250165752A1.** Patent-application art in adjacent LLM-routing space; cited for completeness of the routing landscape.
- **Circuit-breaker / bulkhead patterns (Nygard, Release It!).** Classical resilience patterns adopted conceptually for the fail-open chain's error handling; orthogonal to the route-class binding.

The synthesis: each constituent exists; the **integration of all of them, keyed to a semantic governance route class, with identity-mismatch refusal and a mandatory immutable denial audit, and a self-degrading config store**, is what this publication places into the public domain.

---

5. Prior-Art Delta

Per novel feature: what the closest prior source **has**, what it **lacks**, and what **this** adds.

Prior source	What it has	What it LACKS	What THIS adds
LiteLLM Router/Proxy	DB/config-backed model lists; ordered cross-provider `fallbacks`; retries	Fail-closed **bound to a semantic governance route class**; provider-identity-mismatch refusal; mandatory denial audit; config store that degrades to a hardcoded array	A fail-closed regime selected by *route meaning*; identity-mismatch refusal even on "success"; immutable denial audit on every denial; DB-chain → hardcoded-array graceful degradation
Portkey Gateway Configs / Conditional Routing	Stored per-route configs; conditional routing; fallback targets	Single integrated primitive that, for judge routes, restricts to one provider+model, **bypasses the local/GPU path**, and audits denials	The bypass-local-path + single-requested-attempt + audit trio as a route-class-bound unit
Bifrost plugin fail-closed/compliance suppression	Plugin pipeline can suppress fallback (fail-closed)	Binding to a **request-borne semantic route class** (governance/judge) rather than a plugin/global mode; identity-mismatch detection	The route descriptor's boolean flag *is* the fail-closed selector, per request, plus identity-mismatch refusal
Solo.io/Gloo fail-open-vs-fail-closed gateway semantics	The explicit fail-open/closed vocabulary at the proxy/authz layer	Application to **per-judge-route LLM model substitution** with model-tier integrity and audit	Lifts fail-open/closed semantics into LLM *model-substitution* policy, per route class, with denial audit
Cloudflare AI Gateway / OpenRouter	Provider-agnostic routing; provider preferences; fallback	Governance-route fail-closed; immutable denial audit; self-degrading config	A safety-first route class coexisting with availability-first routes on one bridge
`oh-my-openagent` role→model	Role selects which model to use	Whether *substitution on outage* is permitted, per route	Decouples "which model" (selection) from "may we substitute" (posture), and keys the latter to route class

6. System Architecture

6.1 Components

- **Bridge entrypoint** (`runChatCompletion`-style): the single function every caller invokes. Accepts `messages`, an optional explicit `model`, a `routeKey` (the route identifier), and an optional per-call `allowFallback` override.
- **Route descriptor store**: the configuration source mapping `routeKey → ⟨provider, defaultModel, allowed, fallback, allowFallback⟩`. Loaded with a deep-merge over a hardcoded default so an unset field always has a safe value.
- **Fail-closed evaluator**: derives the boolean posture from the per-call override OR the route descriptor (`allowFallback === false`).
- **Chain builder**: for fail-open, queries the **DB fallback-chains table**; on DB failure, uses the descriptor's hardcoded `fallback` array. For fail-closed, the chain is exactly `[requested]`.
- **GPU/local-inference path**: an optional fast/cheap local path, **skipped** for fail-closed routes.
- **Provider adapters**: per-provider call functions (OpenAI-compatible, Anthropic, Gemini, Mistral, etc.) returning a normalized result including the *actually resolved* provider/model.

- **Identity guard**: for fail-closed, asserts `resolved.provider === requested.provider`; refuses otherwise.
- **Telemetry & audit sink**: records every call's outcome and, for fail-closed, an **immutable denial row** with `status = fail-closed-denied`.

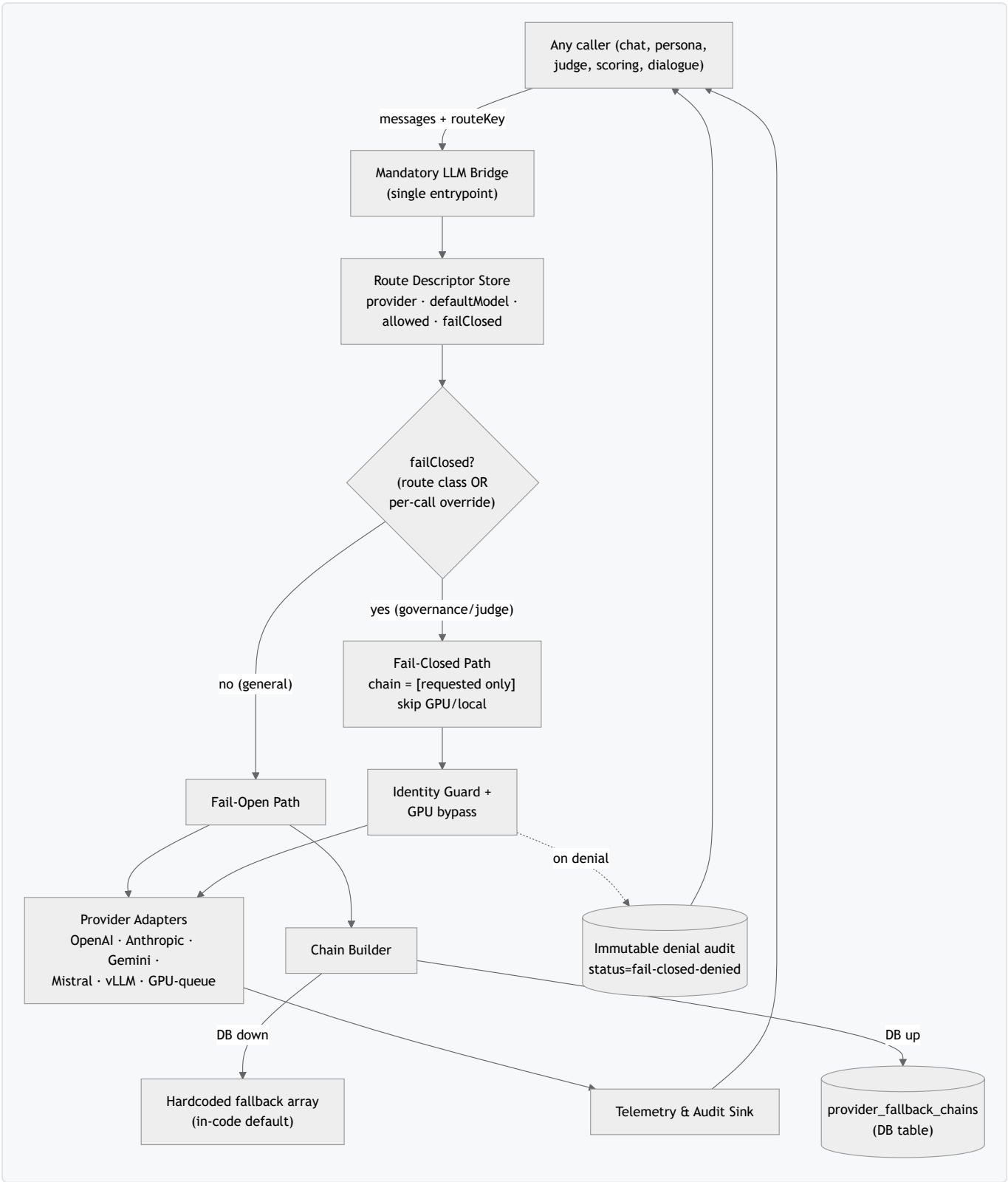

Figure 1 — System architecture. One bridge, two postures, selected by route class.

6.2 Cross-cutting properties

- **Single funnel.** No caller speaks to a provider SDK directly; the bridge owns keys, allow-lists, telemetry, and posture.
- **Provider-agnostic.** Callers consume a normalized result and never branch on which provider answered (except the gateway's own identity guard).
- **Configuration-resilient.** The fallback configuration is read from a DB but the bridge keeps functioning if that DB is down.
- **Auditable.** Every denial is durable and queryable alongside successful calls.
- **Default-safe-for-compatibility, explicit-safe-for-stakes.** The *default* posture is fail-open (so existing callers are unchanged) but governance routes are *pinned* fail-closed in their descriptor — opt-in safety where it matters.

---

7. Detailed Mechanics

7.1 Entry and descriptor resolution

The bridge entrypoint receives `{ messages, model?, persona?, routeKey = 'chat', allowFallback? }`. It loads the effective configuration (a deep-merge of the persisted config over a hardcoded default), then selects the route descriptor `chatCfg = config.routes[routeKey] || {}`.

7.2 Deriving the posture (route-class binding)

```
failClosed := (allowFallback === false) OR (chatCfg.allowFallback === false)
```

The posture is `true` if **either** the caller forced it for this call **or** the route descriptor pins it. Because governance routes (`mastery-judge`, `scoring-judge`, `dialogue`, etc.) carry `allowFallback: false` in their descriptor, **any** request on those routes is fail-closed *regardless of caller* — this is the route-class binding (R3). The default (both unset) is fail-open, preserving compatibility (R1).

7.3 Building the candidate chain

```
requested := { provider: chatCfg.provider, model: model || chatCfg.defaultModel }
providerChain := [ requested ]            // always requested-first (R11)
if NOT failClosed:
    providerChain := append DB/hardcoded fallback entries (de-duplicated)   // R9/R10
// if failClosed: chain stays exactly [ requested ]  (R4)
```

For a fail-closed route the chain has **exactly one** entry — there is nothing to degrade to.

7.4 GPU / local-inference bypass

```
gpuEnabled := (NOT failClosed) AND (env GPU_QUEUE_ENABLED != 'false')
```

The local-inference / GPU-queue fast path resolves to a *local* provider (e.g. `gpu-queue`/vLLM) which is, by definition, **not** the requested Claude/OpenAI tier. Taking it on a governance route *is* the silent degrade we forbid — so fail-closed routes **skip it entirely** (R5). This is a distinct, important step: even a route with a

single-entry chain could be quietly diverted to a local model before the chain is consulted, so the bypass must be explicit.

7.5 Attempt loop, unreachable handling, and identity guard

For each `{ provider, model }` in the chain, the bridge calls the matching provider adapter. Two fail-closed-specific guards apply:

1. **Identity guard (R7).** After a *successful* adapter call, if `failClosed AND resolved.provider !== requested.provider`, the bridge **refuses** the result: it writes a denial audit row (`reason = resolved-non-requested-provider`), throws a tagged `failClosedDenied` error, and does **not** return the substituted output.
2. **Unreachable handling (R6).** If the single requested attempt throws, the catch records an error telemetry row. A `failClosedDenied` error is *final* and re-thrown immediately (never try a next entry). For a fail-open route, the loop continues to the next chain entry.

After the loop, if `failClosed` and nothing succeeded, the bridge writes a denial audit row (`reason = requested-tier-unavailable`) and throws a terminal error — it **never** returns a non-requested result. For fail-open, it throws the last error only after the whole chain is exhausted.

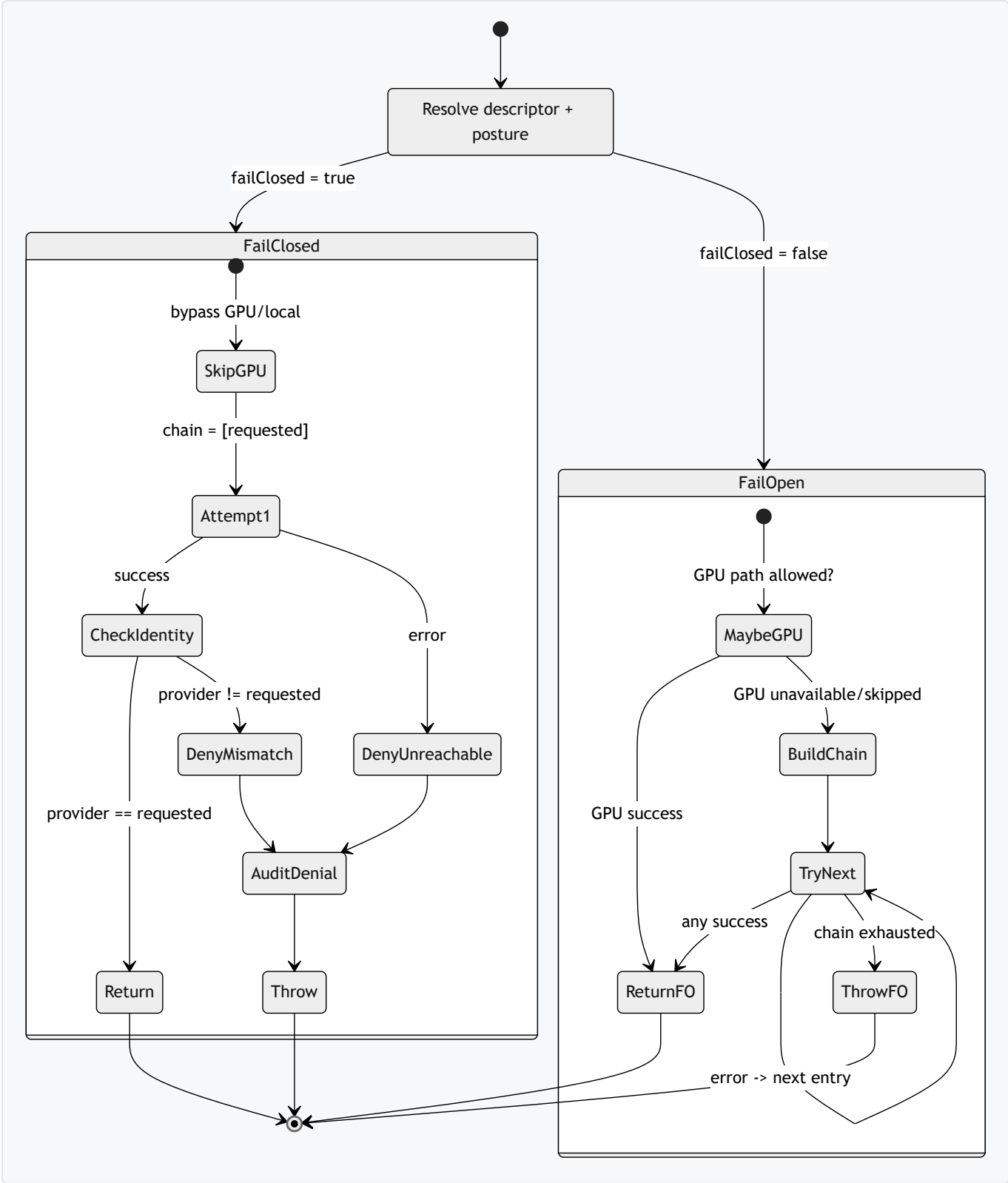

Figure 2 — Core state machine. Fail-closed terminates on outage or identity mismatch; fail-open walks the chain.

7.6 Denial audit

`recordFailClosedDenial({ routeKey, persona, requested, resolvedProvider, resolvedModel, reason, error })` writes a durable row with `status = 'fail-closed-denied'` capturing: the route, the requesting principal, the *requested* provider/model (the audit subject), the reason (`requested-tier-unavailable` or `resolved-non-`

`requested-provider`), and any resolved/cause detail. The function **swallows its own storage errors** — the caller still throws regardless, so a logging outage can never *suppress* a denial (it just degrades observability of it). The audit is **append-only** (immutable): denials are never updated or deleted (R8).

7.7 Fail-open chain construction and self-degradation

```
try:
    rows := DB.query(
        "SELECT providerId, model, priority
         FROM provider_fallback_chains fc
         JOIN providers p ON p.id = fc.providerId
         WHERE fc.capability = $1 AND fc.enabled AND p.enabled
         ORDER BY fc.priority", ['chat'])
    for r in rows (ascending priority):
        key := r.providerId + ':' + r.model
        if key not already seen: append { provider, model }; mark seen
catch (DB unavailable):
    for mdl in chatCfg.fallback:                // hardcoded in-code array
        if mdl != requested.model: append { provider: requested.provider, model: mdl }
```

The **requested attempt is always first** (R11) and DB entries are appended *de-duplicated* against it. Crucially, the `catch` is not an error path that fails the request — it is a **first-class graceful degradation** of the routing layer's own configuration (R10): if the DB that holds the fallback chains is itself down, routing still proceeds using the hardcoded array baked into the descriptor.

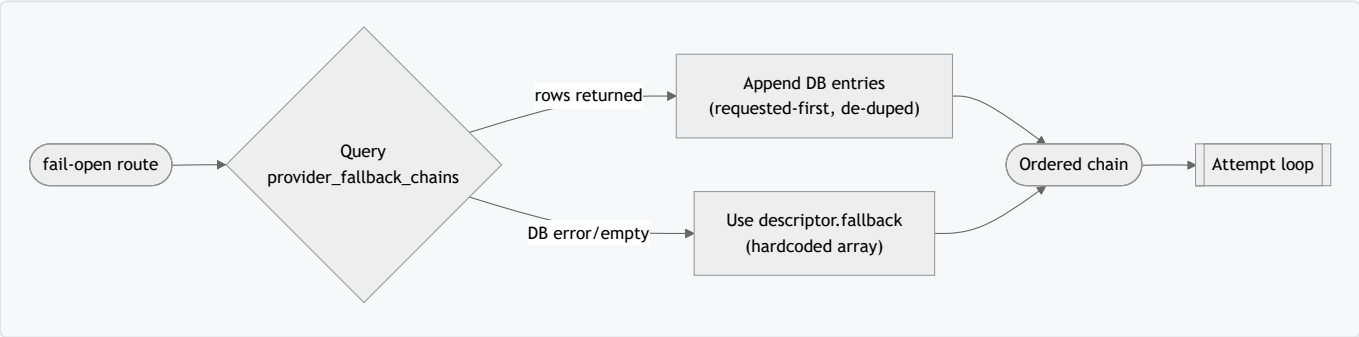

Figure 3 — Fail-open chain construction with self-degradation to a hardcoded array.

7.8 Old-model-id upgrade (never downgrade)

A deprecated model id (e.g. a retired top-tier id) is mapped to its sanctioned successor on every path — explicit param, app-config override, DB-loaded default, and DB-chain entry. This is an **upgrade** (to the current top tier), never a *downgrade* to a cheaper model — so it cannot be used to smuggle a silent degrade onto a fail-closed route (R12).

7.9 Error handling summary

Condition	Fail-closed route	Fail-open route
Requested tier unreachable	Audit denial + terminal throw	Try next chain entry
Resolved provider ≠ requested	Audit denial + terminal throw (identity guard)	N/A (substitution is the point)
GPU/local path available	Skipped entirely	Used first when up
Config DB down	Irrelevant (chain is `[requested]`)	Degrade to hardcoded array
Chain exhausted	Audit denial + terminal throw	Throw last error

---

8. Data Model

The mechanism touches three persisted structures plus the in-code default descriptor.

8.1 Route descriptor (configuration)

A `routeKey → descriptor` map, persisted as JSON config and deep-merged over a hardcoded default. Per descriptor: `provider` (string), `defaultModel` (string), `allowed` (string[]), `fallback` (string[] — the hardcoded array used when the DB is down), `allowFallback` (boolean, default unset ⇒ fail-open).

8.2 `provider_fallback_chains` (DB table)

Ordered cross-provider fallback for fail-open routes.

Column	Type	Notes
`id`	PK	
`capability`	text	e.g. `chat`, `embeddings`
`providerId`	FK → `providers.id`	the provider to fall back to
`model`	text	the model on that provider
`priority`	int	ascending = tried sooner
`enabled`	bool/int	only enabled rows participate

8.3 `providers` (DB table)

Column	Type	Notes
`id`	PK	provider identifier (e.g. `openai`, `anthropic`)
`enabled`	bool/int	disabled providers are excluded from chains

8.4 Telemetry / audit sink (append-only)

Column	Type	Notes
`id`	uuid	generated per row
`route`	text	route label / routeKey
`persona`	text	requesting principal (no PII required)
`provider`	text	for a denial, the *requested* provider (audit subject)
`model`	text	for a denial, the *requested* model
`latencyMs`, `promptTokens`, `completionTokens`	numeric	null on denials
`status`	enum-ish	`success` | `error` | `fail-closed-denied`
`error`	text	for denials: `[fail-closed:<route>] reason=… resolved=… cause=…`

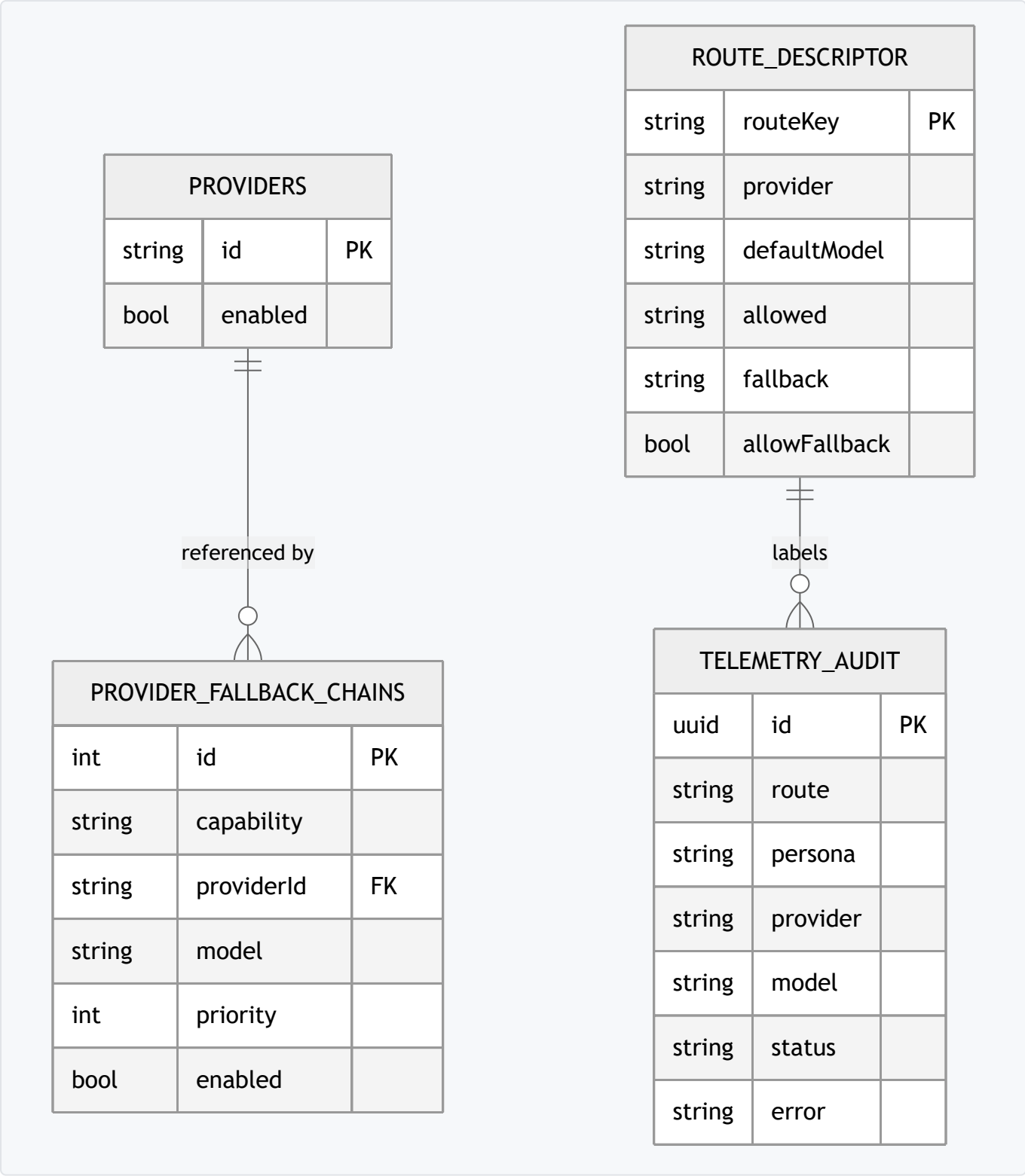

Figure 4 — Data model. The descriptor pins posture; the DB table supplies fail-open chains; the audit sink records denials immutably.

9. Reference Implementation & Enablement

The `src/` directory contains an **original, clean-room, dependency-light** Node.js implementation (`gateway.js`) that reduces the invention to practice. It is illustrative, not production code — it uses in-memory stub provider adapters and an in-memory "DB" so it runs with zero external services.

What `src/gateway.js` demonstrates (mapped to requirements):

- A single `runChatCompletion`-style entrypoint (R1) selecting a route descriptor by `routeKey` (R2).
- Posture derivation `failClosed = (allowFallback === false) || descriptor.allowFallback === false` bound to the route class (R3).
- A fail-closed chain of exactly `[requested]` (R4) and an explicit GPU-path bypass (R5).
- Terminal throw on unreachable requested tier (R6) and the **identity guard** that refuses a resolved-non-requested provider (R7).
- An immutable in-memory `auditLog` written on **both** denial reasons (R8).
- A fail-open chain read from a stub `fallbackChainsTable` (R9) that **throws on demand** to demonstrate degradation to the hardcoded `descriptor.fallback` array (R10), always requested-first and de-duplicated (R11).
- An `upgradeOldModelId` step that upgrades a deprecated id (R12).

A built-in self-check (`npm start` / `node src/gateway.js`) runs five scenarios and asserts the expected outcome of each (see §10), printing PASS/FAIL. See src/README.md for run instructions and configuration points.

Configuration points an integrator would expose: the route-descriptor map; the `provider_fallback_chains` table contents; the per-call `allowFallback` override; and the GPU-path enable flag. None of these are hardcoded business identifiers.

---

10. Worked Example / Scenario

Setup. Two routes share one bridge:

- `chat` — `{ provider: openai, defaultModel: gpt-x, fallback: [gpt-x-mini], allowFallback: unset }` ⇒ **fail-open**.
- `mastery-judge` — `{ provider: anthropic, defaultModel: claude-opus, allowed: [claude-opus], allowFallback: false }` ⇒ **fail-closed**.

Scenario A — general chat during an OpenAI blip. A user message arrives on `chat`. OpenAI times out. The bridge builds the chain `[openai/gpt-x, (DB entries…), openai/gpt-x-mini]`, the first attempt errors, the loop tries the next entry, a fallback model answers, and the user gets a reply. *Outcome: availability preserved.*

Scenario B — mastery certification during an Anthropic outage. A certification request arrives on `mastery-judge`. The chain is exactly `[anthropic/claude-opus]`; the GPU path is skipped. The single attempt errors. The bridge writes a `fail-closed-denied` audit row (`reason = requested-tier-unavailable`) and **throws**. The caller (the judge) catches the throw and leaves the mastery decision **un-made** (the gate stays closed on missing evidence) rather than certifying off a substitute model. *Outcome: integrity preserved — no silent wrong certification.*

Scenario C — identity mismatch. Same `mastery-judge` route, but a misconfiguration causes resolution to return an OpenAI result. The identity guard sees `resolved.provider (openai) ≠ requested.provider (anthropic)`, writes a `fail-closed-denied` audit row (`reason = resolved-non-requested-provider`), and **throws** — refusing the substituted (even though "successful") answer. *Outcome: provider-identity integrity preserved.*

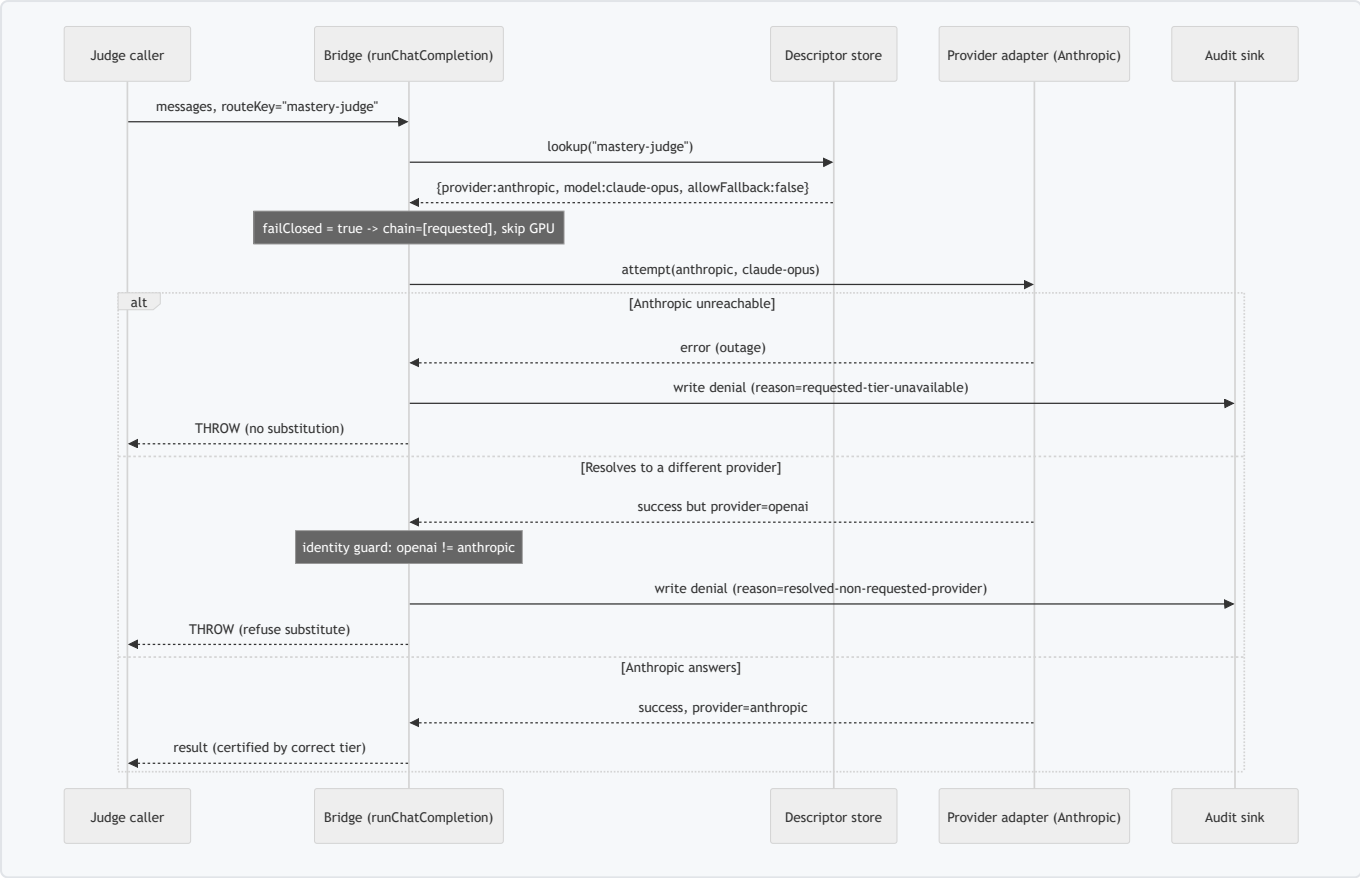

Figure 5 — Sequence for the fail-closed governance route across its three outcomes.

---

11. Security, Safety & Failure Modes

11.1 Posture and threat model

The central safety property is **fail-closed for high-stakes routes**: the *absence* of the correct model is treated as a *denial*, not as a license to substitute. The central availability property is **fail-open for general routes**. The asymmetry is the point.

11.2 Failure-mode / STRIDE-style table

Threat / failure	Vector	Posture & mitigation
Silent downgrade of a high-stakes decision	Provider outage triggers fallback to a cheaper/unvalidated model	Fail-closed: chain = `[requested]`, GPU bypass, terminal throw + audit (R4–R8)
Provider-identity spoof / misroute	Resolution returns a different provider that "succeeds"	Identity guard refuses + audits (R7, R8)
Tampering — suppress a denial	Disable/mute the audit logger	Denial throw is independent of audit success; logger swallows its own errors but the caller still throws (R8)
Repudiation — "the judge never denied"	No durable record of denial	Immutable, append-only `fail-closed-denied` rows are queryable alongside calls (R8)
Config-store outage cascading to an availability outage	DB holding fallback chains is down	Fail-open degrades to hardcoded array; routing continues (R10)
Compatibility regression	Hardening flips existing chat to terminal	Default posture is fail-open; only descriptors that *opt in* (`allowFallback:false`) are fail-closed (R3)
Deprecated-id smuggling	Pass a retired top-tier id to dodge allow-list	Old id is *upgraded* to current top tier, never downgraded (R12)
Denial-of-service via forced fail-closed	Caller passes `allowFallback:false` to force errors	Posture only *restricts substitution*; it cannot widen access; impact is the caller's own request erroring

11.3 Fail-open vs fail-closed — the design rule

> *Fail open when the cost of a wrong/late answer is bounded and recoverable (chat). Fail closed when the cost of a silently-wrong answer is unbounded or unrecoverable (certification, scoring, release gating).* The mechanism makes this rule a per-route data flag rather than a code fork.

---

12. Standards & Framework Mapping

This is **semantic alignment**, not a claim of certified compliance.

Framework / standard	Relevant clause(s)	How this mechanism aligns
EU AI Act (high-risk systems)	Art. 12 (record-keeping/logging), Art. 14 (human oversight), Art. 15 (accuracy/robustness)	Immutable denial audit supports record-keeping; fail-closed prevents a high-risk decision from silently degrading; terminal throw surfaces to a human/owner rather than auto-substituting
NIST AI RMF 1.0	MEASURE (2.x reliability), MANAGE (4.x), GOVERN (audit trails)	Route-class posture + denial audit operationalize "manage and document failure modes of AI components"
ISO/IEC 42001 (AI management systems)	Operational control & monitoring of AI components	Per-route control of substitution policy; durable telemetry/audit
ISO/IEC 27001 / SOC 2	Logging, integrity, change control	Append-only denial records; configuration-driven, auditable routing
Reliability engineering (Nygard, Release It!)	Circuit breaker, bulkhead, fail fast	Fail-open chain = bulkheaded retries; fail-closed = fail-fast for integrity-critical paths
NIST SP 800-92 (log management)	Audit log integrity	`fail-closed-denied` rows are append-only and queryable

No certification is asserted; the table records where the mechanism is *semantically consistent* with each framework's intent.

13. Evaluation Methodology

All figures below are **illustrative** placeholders to define *how* one would measure, not measured results.

13.1 Dimensions and metrics

Dimension	Metric	How to measure	Interpretation
Integrity (fail-closed correctness)	Silent-substitution rate on governance routes	Inject provider outages; count any governance result whose `resolved.provider ≠ requested.provider`	Target **0** (any non-zero is a defect)
Auditability	Denial-capture rate	Force N denials; count `fail-closed-denied` rows	Target **100%**
Availability (fail-open)	Successful-completion rate under fault	Inject first-provider faults; measure % of chat requests that still return	Higher is better (illustrative: 99.x%)
Config resilience	Routing-continuity under config-DB outage	Take the chains DB down; measure % of fail-open requests still routed via hardcoded array	Target **100%**
Overhead	Added latency of posture + identity check	Compare p50/p95 with/without the guards	Target negligible (µs-scale, illustrative)
Compatibility	Behavior drift for existing fail-open callers	Replay a baseline trace; diff outcomes	Target **0** changed outcomes for unflagged routes

13.2 Experimental protocol (sketch)

1. Define two routes (one fail-open, one fail-closed) and a stub provider that can be toggled to outage/mismatch.
2. Run a fault-injection matrix (healthy / outage / identity-mismatch / config-DB-down) × (fail-open / fail-closed).
3. Assert the §10 expected outcomes; count audit rows; record latency deltas.
4. The `src/gateway.js` self-check is a minimal instance of steps 1–3.

---

14. Novelty & Inventive Claims

Presented as prose for clarity; one independent claim and twelve dependent claims, each narrowing on a distinct novel feature.

Claim 1 (independent). A method, in a multi-provider LLM gateway through which all inference callers are funneled, comprising: receiving a request bearing a route identifier; selecting, by the route identifier, a stored route descriptor declaring a provider, a default model, an allowed-model list, and a boolean fail-closed flag; for a route whose fail-closed flag is set — a governance/LLM-judge route class — constructing a candidate chain consisting only of the single requested provider+model and bypassing any local-inference/GPU-queue path, such that (i) if the requested provider/model cannot be reached the gateway terminates the request with an error rather than substituting any other model, and (ii) if resolution returns a provider other than the requested provider the gateway detects the provider-identity mismatch, refuses the substituted result, and in both cases writes an immutable audit record of the denial; and for a route whose fail-closed flag is unset, constructing an ordered cross-provider fallback chain read from a database table and, only when that table is unavailable, degrading to an in-code hardcoded fallback array — wherein the per-request fail-closed-versus-open determination is bound to the semantic route class rather than to the requesting caller.

Claim 2. The method of claim 1, wherein the fail-closed flag is set by either a per-call override carried on the request OR a value pinned in the route descriptor, such that every request on a governance route is fail-closed independent of the requesting caller's identity.

Claim 3. The method of claim 1, wherein the default determination when neither the per-call override nor the descriptor specifies a posture is fail-open, preserving the unchanged behavior of pre-existing callers.

Claim 4. The method of claim 1, wherein bypassing the local-inference/GPU-queue path for a fail-closed route comprises disabling that path specifically because it resolves to a local provider that is, by construction, not the requested provider tier.

Claim 5. The method of claim 1, wherein the provider-identity-mismatch refusal of clause (ii) is performed after an otherwise-successful adapter call by comparing the actually-resolved provider against the requested provider and discarding the result if they differ.

Claim 6. The method of claim 1, wherein the immutable audit record distinguishes at least two denial reasons — an unreachable-requested-tier reason and a resolved-non-requested-provider reason — and records the requested provider and model as the audit subject.

Claim 7. The method of claim 6, wherein writing the audit record is performed by a routine that swallows its own storage errors while the gateway still terminates the request, such that a logging outage cannot suppress the denial itself.

Claim 8. The method of claim 1, wherein constructing the ordered cross-provider fallback chain comprises querying a database table that joins per-capability fallback entries to a providers table, filtering to enabled providers and enabled entries, and ordering by an ascending priority column.

Claim 9. The method of claim 8, wherein, upon failure or unavailability of said database table, the gateway constructs the chain from a hardcoded fallback array declared in the route descriptor, thereby keeping the routing layer operational despite the failure of its own configuration store.

Claim 10. The method of claim 8, wherein the requested provider+model is always placed first in the chain and the database-derived entries are appended after de-duplication against the requested attempt and against each other.

Claim 11. The method of claim 1, wherein a fail-closed denial encountered mid-chain is tagged as final and re-thrown immediately without attempting any further chain entry, whereas a non-final error on a fail-open route advances to the next chain entry.

Claim 12. The method of claim 1, further comprising upgrading a deprecated model identifier appearing on any path — explicit parameter, per-application override, descriptor default, or database chain entry — to a sanctioned successor identifier, the mapping being defined only in the upgrade direction so that it cannot effect a silent downgrade on a fail-closed route.

Claim 13. The method of claim 1, wherein the gateway records, for every completed call as well as every denial, a telemetry row including route, requesting principal, provider, model, and status drawn from a set including a fail-closed-denied status, such that denials are queryable alongside successful calls.

Claim 14. The method of claim 1, wherein the route descriptor is obtained by deep-merging a persisted configuration over a hardcoded default descriptor so that any unspecified field of the descriptor resolves to a safe default value.

---

15. Limitations & Threats to Validity

- **Patentability is intentionally low.** Independent screening rated US ~22/100 and DE/EPO ~15/100: each constituent is anticipated, and a claim narrow enough to clear §101 is trivially designed around (flip the default, drop the audit row). This document is therefore a *defensive* disclosure, not a filing basis.
- **Prior art is close.** LiteLLM (DB config + ordered fallback), Portkey (stored per-route policy), Bifrost (plugin fail-closed), and Solo.io/Gloo (fail-open/closed vocabulary) each cover *part* of the combination. The novelty is the integration, not any single element.
- **Intentionally withheld.** The specific **persona-role → route mapping** and the **model-tier policy** (which role gets which tier) are retained as trade secret and are deliberately *not* disclosed here; this publication is scoped to the routing mechanism only.
- **Scope boundary.** The *judge/scoring decision logic itself* and any RAG reranking are out of scope and covered, if at all, by separate disclosures.

- **Illustrative numbers.** All evaluation figures are methodology placeholders, not measured benchmarks.

---

16. Future Work & Open-Source Reference App

A minimal open-source **reference gateway** implementing this mechanism is planned (see docs/OPEN-SOURCE-APP.md): a small HTTP service exposing `POST /chat` with a `routeKey`, a route-descriptor config file, a `provider_fallback_chains` table, pluggable provider adapters, and the denial-audit sink — deployable to a generic Kubernetes/AKS cluster via standard manifests/Helm. Roadmap: (1) the clean-room core (this repo's `src/`); (2) the HTTP service + adapters; (3) the Postgres-backed chains table + migration; (4) a denial-audit dashboard; (5) conformance tests asserting the §13 metrics.

---

17. Conclusion

Funneling all inference through one bridge is the right architecture, but a *single global failure posture* is wrong for a platform that carries both conversational and governance traffic. Binding the **fail-closed-versus-fail-open posture to the semantic route class** — and equipping the fail-closed path with a single-requested-attempt chain, a GPU-path bypass, a provider-identity-mismatch refusal, and an immutable denial audit, while the fail-open path uses a DB-driven chain that degrades to a hardcoded array — lets one provider-agnostic bridge deliver availability where it is wanted and integrity where it is required. This publication places that combination into the public record so it remains free to practice.

Appendix A — Prior-Art Landscape (well-trodden vs candidate-novel)

Well-trodden (treat as known art):

- Multi-provider LLM gateways with a single funnel (LiteLLM, Portkey, Cloudflare AI Gateway, OpenRouter, Bifrost).
- Ordered, cross-provider fallback chains (LiteLLM `fallbacks`, Portkey targets).
- DB/config-backed model lists and per-route configs.
- Fail-open vs fail-closed *vocabulary* at the proxy/authz layer (Solo.io/Gloo, ext-proc patterns).
- Role→model selection (`oh-my-openagent` and similar).
- Circuit-breaker / bulkhead resilience patterns.

Candidate-novel (the disclosed combination):

- Fail-closed posture **bound to a semantic governance/judge route class** carried on the request.
- **GPU/local-inference bypass** justified specifically by non-requested-provider resolution.
- **Provider-identity-mismatch refusal** of an otherwise-successful call.
- **Immutable denial audit on every denial**, with distinct reasons, independent of audit-store success.
- **DB fallback chain that gracefully degrades to a hardcoded in-code array** — config-availability resilience of the routing layer itself.

- The **coexistence of opposite postures on one bridge**, default-safe-for-compat and explicit-safe-for-stakes.

Honesty attestation. The prior-art searches behind this document are **directional, not exhaustive**. They reflect a reasonable practitioner's review of public gateways, documentation, and patent-adjacent material as of the publication date; they are **not** a professional freedom-to-operate or invalidity search. No representation is made that every relevant reference has been found. This disclosure's purpose is defensive — to add to the public prior-art record — not to assert validity of any claim.

Appendix B — Glossary

- **Bridge / gateway** — the single in-process function all callers use for inference.
- **Route / routeKey** — the identifier on a request selecting a route descriptor.
- **Route descriptor** — `{ provider, defaultModel, allowed, fallback, allowFallback }`.
- **Fail-closed** — a posture where an inability to use the *requested* model is terminal (no substitution).
- **Fail-open** — a posture where substitution across a fallback chain is permitted for availability.
- **Governance / judge route** — a route whose output is a high-stakes decision (certification, scoring, gating).
- **Identity guard** — the check that the *resolved* provider equals the *requested* provider on a fail-closed route.
- **Fallback chain** — the ordered list of `{provider, model}` attempts for a fail-open route.
- **Denial audit** — the immutable `fail-closed-denied` record written when a fail-closed route refuses to substitute.
- **Graceful degradation (of config)** — using a hardcoded array when the DB holding the chains is unavailable.

Appendix C — Reference-Implementation Index

File	Purpose
`src/gateway.js`	Clean-room, dependency-light core: descriptor resolution, posture derivation, fail-closed chain + GPU bypass + identity guard + denial audit, fail-open DB chain with hardcoded degradation, model-id upgrade, and a five-scenario self-check.
`src/README.md`	What `src/` shows, how to run it, configuration points, and the clean-room/illustrative disclaimer.

Appendix D — Defensive-Publication Deposit & Timestamp

- **Publication date:** 2026-06-25.
- **Publisher:** Gus IT LLC (Florida, USA). **Author:** Gustavo Assuncao, PhD.
- **Deposit channel:** to be assigned (IP.com / Zenodo / arXiv) — establishes a public, dated, citable prior-art record. No DOI is asserted at the time of writing.
- **Intent:** *This disclosure is intentionally made public to bar later patenting of the described combination by others.* By publishing the full enabling description, the technique is placed in the prior art as of the date above and remains free for anyone to practice. Licensed under AGPL-3.0-or-later with its express patent grant.