**ADR‑0002 — Legacy Endpoints Feature Flag (EnableLegacyHttpSse)**

**Document:** docs/adr/0002-legacy-endpoints-feature-flag.docx  
**Status:** **Accepted**  
**Date:** 2025‑09‑27  
**Project:** MCPX‑KendoBridge (Project Code: MCPX‑KendoBridge)  
**Deciders:** DoSE (Accountable), SRE Lead, Dev Lead, Security Lead  
**Consulted:** T‑Arch, DBA, QA Lead, DocFactory  
**Tags:** feature‑flag, transport, compatibility, policy, compliance

**Guardrails (non‑negotiable):** GitHub‑first SDLC with merge‑queue and required checks; four environments **Alpha → Beta → RTM (validates on Prod DB read‑only) → Prod**; **Add‑only** schema; **Stored‑procedure‑only** DB access; **No‑Hard‑Coding** of dynamic values (config/flags from SQL via sp\_Config\_\*, sp\_Feature\_IsEnabled, sp\_Lookup\_Get); secrets only in **GitHub Environments**.

**1) Context**

Our **primary transport** is Streamable‑HTTP with **SSE** exposed by POST /mcp and GET /mcp. Some early/legacy MCP clients may only speak a simpler **HTTP+SSE** pattern that expects:

* POST /messages — JSON (non‑streaming) request/response, and
* GET /sse — SSE event stream tied to a session.

We must **optionally** support those compatibility endpoints **without** weakening our default policy, security posture, or SDLC gates. The decision here governs how those endpoints are exposed, controlled, audited, and eventually removed. (Primary transport rationale is recorded in **ADR‑0001**.)

**2) Decision**

Introduce a **DB‑sourced feature flag** **EnableLegacyHttpSse** that **gates the legacy endpoints**:

* When **EnableLegacyHttpSse = false** (default), calls to **POST /messages** and **GET /sse** return **403** with the canonical envelope { code: "feature\_disabled", message, requestId? }.
* When **EnableLegacyHttpSse = true**, those endpoints are served as **compatibility shims** that bridge to the same child process/session model used by /mcp.
* The flag value is read via **sp\_Feature\_IsEnabled('EnableLegacyHttpSse')**; **no literals** are embedded in application code; state is **environment‑specific** via DB seeds/migrations. **Secrets never live in the DB.**

This behavior is **contracted** in **OpenAPI 3.1** (legacy paths present, with examples for the 403 feature\_disabled envelope) and verified by tests and monitoring.

**3) Options Considered**

| **Option** | **Pros** | **Cons** |
| --- | --- | --- |
| **DB‑flag‑gated legacy endpoints (chosen)** | Centralized policy; observable and auditable; environment‑specific; zero redeploy to toggle | Small code surface to keep; must test off‑by‑default path |
| Always enabled legacy endpoints | Max compatibility | Larger attack surface; policy drift; confusing client guidance |
| Remove legacy endpoints entirely | Clean interface, smaller surface | Breaks existing clients; harder adoption path |
| Build client adapters/gateways | Keeps API pure | More moving parts to operate and support |

**4) Rationale**

* **Policy control & auditability.** A **DB‑driven** flag is visible in /config/effective (non‑secret) and can be proven in the **Evidence Pack** without code changes.
* **Security by default.** Keeping the flag **OFF** preserves the **smallest surface area**; attempts hit a clear **403 feature\_disabled** envelope captured in dashboards.
* **Operational simplicity.** Same session model and child process are used; when enabled, legacy endpoints simply **forward** to the existing bridge.
* **Compliance.** Upholds **No‑Hard‑Coding**, **SP‑only**, **add‑only** rules: the flag is in FeatureFlag and read via **sp\_Feature\_IsEnabled**.

**5) Implications & Constraints**

1. **Default OFF** in all environments; enabling requires change control (PR + approvals) and **Evidence**.
2. **Error envelope** must be returned consistently: { code: "feature\_disabled", message, requestId? } with **403** status.
3. **Monitoring** should track usage of legacy endpoints and **spikes** in feature\_disabled to detect misconfigurations or client drift.
4. **Docs** (FR/NFR, OpenAPI, runbooks, Error Catalog) must remain in sync with this gating.

**6) API Surface (authoritative summary)**

* **Legacy endpoints (feature‑flagged):**
  + POST /messages — JSON‑RPC (opaque JSON)
  + GET /sse — SSE notifications (requires Mcp-Session-Id)
* **Disabled response (default):**
* { "code": "feature\_disabled", "message": "Endpoint disabled by feature flag", "requestId": "req-456" }
* **OpenAPI 3.1** includes both endpoints with the above example and the note that they are **flag‑gated**. (See api/openapi/mcp-proxy.yaml.)

**7) Data & DB Contracts**

* **Table:** FeatureFlag([Name] PK, [Enabled] BIT, [UpdatedAt] DATETIME2)
* **Seed:** EnableLegacyHttpSse = 0 (all envs by default)
* **SP:**
* sp\_Feature\_IsEnabled(@Name NVARCHAR(200)) → BIT
* **DAL Policy:** **Stored‑procedure‑only**, **EXECUTE‑only** grants, **no table CRUD** permissions for the app principal. **Add‑only** migrations maintain forward‑compatibility.

**8) Operations & Runbooks Alignment**

* **Deploy:** Confirm flag state via /config/effective; keep **OFF** unless an approved exception exists.
* **Rollback/Incident:** If legacy endpoints create instability, toggle **OFF** first (Config Rollback) before code rollback.
* **Scale‑out:** Legacy traffic, if enabled, follows the same **SSE pass‑through** and **sticky routing** by Mcp‑Session‑Id.
* **License/Secrets:** No change; secrets (SQL, Telerik license) remain only in **GitHub Environments**.

**9) Testing & Evidence**

**Tests**

* **Unit:** gate check returns 403 feature\_disabled when sp\_Feature\_IsEnabled('EnableLegacyHttpSse') = 0.
* **Contract:** OpenAPI response examples for legacy endpoints; CI **lint/diff** enforced.
* **E2E (Gherkin):**
  + **04\_origin\_denied.feature** remains unaffected (policy).
  + **New:** “Legacy flag off denies endpoints” scenario (see Gherkin sketch below).

**Gherkin sketch (add to /tests/gherkin/05\_legacy\_flag.feature)**

Feature: Legacy endpoints are gated by feature flag

Scenario: Legacy is disabled by default

Given the feature flag "EnableLegacyHttpSse" is false

When I POST "/messages" with a valid JSON-RPC request

Then the response status is 403

And the error envelope code is "feature\_disabled"

Scenario: Legacy is enabled by ops

Given the feature flag "EnableLegacyHttpSse" is true

When I GET "/sse" with "Mcp-Session-Id: test"

Then I receive an SSE stream with heartbeats

**Evidence Pack**

* Screenshot of /config/effective showing EnableLegacyHttpSse=false.
* OpenAPI lint/diff output containing legacy endpoints and examples.
* Monitoring snapshot: counts of feature\_disabled over the release window. **Retention ≥ 1 year.**

**10) Monitoring & Alerts**

* **Counters:** errors\_total{code="feature\_disabled"}; http\_requests\_total{path=~"/messages|/sse"}.
* **Panels:** Usage of legacy endpoints, error code distribution, correlation with **TTFB** and readiness.
* **Alerts:** Notify on **unexpected spikes** in /messages or /sse while flag is OFF (possible client misconfig).

**11) Security & Compliance**

* **Default deny** model: flag OFF → 403; origin allow‑list enforced as usual.
* **No‑Hard‑Coding:** gating strictly via **DB flag**; never by code constants.
* **SP‑only & Add‑only:** enforced across migrations and grants.
* **Secrets policy:** unchanged — secrets never in DB/code/logs; only in **GitHub Environments**.

**12) Backout Plan**

* If enabling legacy endpoints degrades service, **toggle the flag OFF** via DB (Config Rollback), confirm /messages and /sse return 403, then proceed with the standard **Rollback Runbook** if required. Evidence (before/after snapshots) must be attached.

**13) Decommission Path (sunset criteria)**

* Publish deprecation notice in release notes.
* Track zero usage of legacy endpoints for **two consecutive releases**.
* Remove legacy routes from OpenAPI and code behind a major version bump, with this ADR marked **Superseded**.

**14) Related & Derived Artifacts**

* **ADR‑0001:** Transport choice (Streamable‑HTTP + SSE).
* **OpenAPI 3.1:** api/openapi/mcp-proxy.yaml (legacy endpoints documented as flag‑gated).
* **Error Catalog:** feature\_disabled (403) envelope.
* **FR/NFR:** transport, gating, and budgets.
* **Runbooks:** deploy/rollback/incident/scale\_out; config rollback steps.
* **Evidence Pack:** config snapshots, OpenAPI lint/diff, monitoring.

**15) Appendices**

**A) Example seeds (add‑only)**

-- VYYYYMMDDHHMM\_\_seed\_featureflag.sql

MERGE dbo.FeatureFlag AS t

USING (VALUES (N'EnableLegacyHttpSse', CAST(0 AS BIT), SYSUTCDATETIME())) AS s([Name],[Enabled],[UpdatedAt])

ON (t.[Name] = s.[Name])

WHEN MATCHED THEN UPDATE SET t.[Enabled] = s.[Enabled], t.[UpdatedAt] = s.[UpdatedAt]

WHEN NOT MATCHED THEN INSERT([Name],[Enabled],[UpdatedAt]) VALUES(s.[Name],s.[Enabled],s.[UpdatedAt]);

**B) OpenAPI excerpt (responses)**

paths:

/messages:

post:

responses:

'403':

description: Legacy endpoints disabled by feature flag

content:

application/json:

schema: { $ref: '#/components/schemas/ErrorEnvelope' }

examples:

featureDisabled:

value: { code: "feature\_disabled", message: "Endpoint disabled by feature flag", requestId: "req-456" }

**Record maintained by DocFactory. Changes to gating semantics require synchronized updates to OpenAPI, Error Catalog, tests, runbooks, and evidence procedures.**