**ADR‑0001 — Transport Choice: Streamable‑HTTP + SSE for MCP Bridging**

**Document:** docs/adr/0001-transport-choice-streamable-http-sse.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:** transport, streaming, SSE, sessioning, ingress, KendoReact MCP, JSON‑RPC

**Guardrails (non‑negotiable):** GitHub‑first SDLC; 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 (all runtime config from SQL AppConfig/FeatureFlag/Lookup via sp\_Config\_\*, sp\_Feature\_IsEnabled, sp\_Lookup\_Get); secrets only in **GitHub Environments**.

**1) Context**

We must expose a **remote MCP** server over HTTP so Remote MCP clients (e.g., ChatGPT/MyGPT connectors) can call the **Telerik KendoReact MCP** tool process (spawned via **STDIO**) and receive **streamed** tool output. Requirements:

* **Primary transport** supports **incremental streaming** to clients and **simple operation through commodity ingress**.
* **Compatibility** with existing HTTP infrastructure and corporate proxies (TLS termination, WAF, L7 LB) without special tunneling.
* **Clear sessioning**: one child process per **Mcp‑Session‑Id**; stable **error envelope**; **origin allow‑list** policy.
* **Operational fit** with DocFactory gates: OpenAPI 3.1, monitoring/SLOs (incl. **SSE TTFB p95 ≤ 200 ms**), and Evidence Pack.

**2) Decision**

Adopt **Streamable‑HTTP with Server‑Sent Events (SSE)** as the **primary transport**:

* **POST /mcp** accepts a single JSON‑RPC message.
  + If Accept: application/json → returns JSON once ready.
  + If Accept: text/event-stream → returns **SSE** stream (event: message, incrementing id), with : **heartbeat comments** every Network:SseKeepAliveSeconds (DB‑sourced).
* **GET /mcp** opens an **SSE** channel for **background notifications** within the same Mcp‑Session‑Id.
* **Legacy compatibility** endpoints **POST /messages** and **GET /sse** are retained **behind a feature flag** EnableLegacyHttpSse (default **OFF**).
* Configuration (timeouts, keep‑alive cadence, allowed origins, child spawn command/args/cwd) is **DB‑driven** via SPs; **no literals in code**.

This decision is reflected in the **OpenAPI 3.1** contract, monitoring/NFR budgets, runbooks (deploy/rollback/incident/scale‑out), and the error catalog.

**3) Options Considered**

| **Option** | **Pros** | **Cons** |
| --- | --- | --- |
| **SSE over HTTP (chosen)** | Works with standard HTTP ingress; easy to observe/log; minimal handshake; great for **server→client** incremental output; simple to bridge from child STDIO (line/frame model). | **One‑way streaming** (server→client) — client uploads request once; background notifications require a separate GET stream; limited binary (needs base64); some intermediaries require **buffering off**. |
| **WebSockets** | Full‑duplex; lower framing overhead for binary. | Operationally heavier (sticky upgrades, special LB support); more failure modes on enterprise proxies; more complex back‑pressure; not needed for our request→stream pattern. |
| **gRPC streaming** | Strong typing; bidi streaming cores. | Requires gRPC‑aware ingress; adds heavy toolchain; not a fit for JSON‑RPC passthrough and existing clients. |
| **Long‑polling/Chunked JSON** | Simple to implement server‑side. | Inefficient; jittery first‑byte times; poor UX; harder to standardize heartbeat semantics. |

**4) Rationale**

* **Operational simplicity.** SSE uses **text/event-stream** over plain HTTP and is well‑supported by reverse proxies/load balancers when buffering is disabled; it fits our **GitHub‑first**, evidence‑heavy SDLC with minimal moving parts.
* **Match to workload.** The MCP proxy pattern is **request→(server‑stream)**, with occasional background notifications. SSE maps cleanly: **POST /mcp** for streamed tool responses; **GET /mcp** for background events.
* **Observability.** Line‑oriented frames map to structured logs and metrics (TTFB, heartbeat gaps) without logging payload bodies (compliance).
* **Security and policy.** It’s straightforward to enforce **origin allow‑list**, bearer (at gateway), and a **stable error envelope** across all endpoints.

**5) Implications & Constraints**

**5.1 Non‑Functional budgets (must meet)**

* **SSE TTFB p95 ≤ 200 ms**; **JSON latency p50 ≤ 300 ms / p95 ≤ 800 ms**; **restart‑to‑ready ≤ 30 s**; availability ≥ 99.9%; **≥ 200 concurrent sessions/replica** (CPU‑bound first). These are codified in **NFR** and **Monitoring/SLOs**.

**5.2 Ingress requirements**

* **No buffering** for text/event-stream.
* Adequate **read/idle timeouts** to permit heartbeat cadence.
* **Sticky routing** keyed by **Mcp‑Session‑Id** so a session’s requests land on the same replica. (Examples and settings are embedded in runbooks.)

**5.3 Sessioning & state**

* **One child process per Mcp‑Session‑Id**; request‑scoped streams and session‑scoped notifications.
* **Graceful drain** on shutdown/rollout to avoid truncating streams. (Runbooks cover readiness flip → drain → terminate child.)

**5.4 Security & compliance**

* **Origin allow‑list** (DB key Security:AllowedOrigins), enforced at the API; disallowed origins receive 403 origin\_forbidden.
* **Secrets** never in responses/logs/DB; **Telerik license** is build‑time only via GitHub Environments.
* **DB policy:** **Add‑only**, **SP‑only**, **No‑Hard‑Coding**; app has **EXECUTE‑only** on the whitelisted SPs.

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

* **Endpoints:**
  + POST /mcp → JSON (default) or **SSE** (when Accept: text/event-stream)
  + GET /mcp → **SSE** notifications (requires Mcp‑Session‑Id)
  + POST /messages, GET /sse → **legacy** (gated by EnableLegacyHttpSse)
  + GET /ready, GET /healthz, GET /config/effective (non‑secret)
* **Headers:** Mcp-Session-Id (in/out), MCP-Protocol-Version (future).
* **Error envelope:** { code, message, requestId? } — mandatory on any non‑2xx.
* **OpenAPI 3.1:** api/openapi/mcp-proxy.yaml is the contract of record.

**7) Operations & Runbooks Alignment**

* **Deploy:** verify ingress SSE settings; canary → full rollout; capture /ready, /config/effective, **TTFB** smoke; attach to Evidence.
* **Rollback:** **graceful SSE drain**, image flip to LKG; re‑check **TTFB**, heartbeats, availability.
* **Incident:** P1 if availability <99% (10‑min), /ready red >2 min, or **TTFB p95 > 200 ms** for 15 min; triage ingress buffering, sticky routing, DB SP reachability.
* **Scale‑out:** HPA on CPU (+ optional session\_count); **PDB** to avoid mass eviction; verify no stream truncation on scale‑in.  
  All procedures are documented in the runbooks with explicit commands and checks.

**8) Testing & Evidence**

**Acceptance tests (E2E Gherkin):**

1. **Session establish & Mcp‑Session‑Id issuance** via POST /mcp.
2. **Streamed tool call** via POST /mcp with Accept: text/event-stream (monotonic id, final frame).
3. **Background notification** via GET /mcp SSE.
4. **Origin denied** returns 403 origin\_forbidden envelope.

**Evidence Pack (per release; retain ≥ 1 year):**

* OpenAPI spec + lint/diff; CodeQL SARIF; Dependency Review; Secret‑scan summary; SBOM; /ready and /config/effective; **TTFB** and heartbeat results; monitoring screenshots (post‑deploy and 24‑hour).

**9) Consequences**

* **Positive:** Simpler ops than WebSockets; observability built‑in; minimal client changes; straightforward policy enforcement and error handling.
* **Trade‑offs:** No true client→server streaming (one request per call); heartbeat discipline and ingress conformance are critical; large binary data requires base64 (not currently needed).
* **Mitigations:** Separate **GET SSE** for notifications; heartbeat metrics/alerts; ingress conformance checks in CI; explicit error taxonomy and runbooks.

**10) Backout Plan**

If SSE proves unstable in an environment:

1. **Fall back** to JSON responses on POST /mcp while investigating ingress.
2. Optionally **enable legacy** /messages + /sse via feature flag for affected clients (document in Evidence).
3. Revert ingress changes and/or roll back image per rollback runbook.

**11) Related & Derived Artifacts**

* **OpenAPI:** api/openapi/mcp-proxy.yaml (streaming examples, errors).
* **NFR:** docs/06\_nfr.docx (budgets: JSON latency, **SSE TTFB**, readiness).
* **Monitoring:** docs/11\_monitoring.docx (SLIs/SLOs, alerts, dashboards).
* **Runbooks:** deploy / rollback / incident / scale\_out / rotate\_telerik\_license.
* **Error Catalog:** docs/error\_catalog.docx.
* **DB Grants & SP Signatures:** docs/07a\_db\_grants\_sp\_signatures.docx.

**12) Appendices**

**A) SSE framing (illustrative)**

: 2025-09-27T12:00:00Z # heartbeat (every Network:SseKeepAliveSeconds)

event: message

id: 1

data: {"jsonrpc":"2.0","id":"9","result":{"partial":true}}

event: message

id: 2

data: {"jsonrpc":"2.0","id":"9","result":{"final":true}}

**B) Ingress knobs (examples)**

* **NGINX:** proxy\_buffering off; proxy-read-timeout: 3600; upstream-hash-by: "$http\_mcp\_session\_id".
* **Envoy:** stream\_idle\_timeout: 0s.  
  (See runbooks for full snippets and validation steps.)