**MCPX‑KendoBridge — System Context & Architecture**

**Document:** docs/04\_context.docx  
**Project Code:** MCPX‑KendoBridge  
**Version:** 1.0.0 (Draft)  
**Last Updated:** 2025‑09‑23  
**Document Owner:** DoSE (Accountable) — DocFactory (Responsible)

**Purpose.** Provide a complete, implementation‑ready architecture reference for MCPX‑KendoBridge: context, containers, components, deployment and data‑flow views; trust boundaries and STRIDE threat model; runtime sequences; and traceability to FR/NFR, CI/CD gates, and database policies (**No‑Hard‑Coding**, **SP‑only**, add‑only). This aligns with Technijian’s GitHub‑first SDLC and evidence expectations.

**Document Control**

**Revision History**

| **Version** | **Date** | **Author/Role** | **Summary of Changes** |
| --- | --- | --- | --- |
| 1.0.0‑D | 2025‑09‑23 | DocFactory (R) | Initial system context, C4 views, DFD, STRIDE, sequences |

**Approvals**

| **Name / Role** | **Responsibility** | **Signature / Date** |
| --- | --- | --- |
| Director of Software Engineering | Accountable (A) | \_\_\_\_ / \_\_\_\_ |
| Systems Architect (T‑Arch) | Consulted (C) | \_\_\_\_ / \_\_\_\_ |
| QA Lead | Consulted (C) | \_\_\_\_ / \_\_\_\_ |

**Table of Contents**

1. Scope & Constraints
2. Architecture Narrative (Executive Overview)
3. Context View (C4‑L1)
4. Container View (C4‑L2)
5. Component View (C4‑L3)
6. Deployment View (Environments & Promotion)
7. Data‑Flow Diagrams & Trust Boundaries (DFD‑L0/L1)
8. STRIDE Threat Model & Mitigations
9. Runtime Sequences (Happy‑Path & Exceptions)
10. Interfaces & Contracts (API, Headers, DB, Config)
11. Observability (Health, Readiness, Metrics, Logs)
12. Performance, Capacity & Scalability
13. Fault Model & Resilience Patterns
14. Compliance & Evidence (ASVS mapping, CI/CD gates)
15. Architecture Decisions (ADR Index)
16. Traceability (FR/NFR ↔ Views)
17. Assumptions
18. Next Steps
19. Appendices (Mermaid sources)

**1) Scope & Constraints**

**In Scope (MVP).**  
HTTP/Streamable‑HTTP transport for MCP; POST /mcp (JSON or SSE), GET /mcp (SSE); session management using **Mcp‑Session‑Id**; one **child STDIO** process per session; origin allow‑list from DB; /healthz, /ready, /config/effective; structured logs and minimal metrics.

**Constraints (hard rules).**

* **No‑Hard‑Coding.** All dynamic values from DB via SPs or vault. No ad‑hoc SQL.
* **SP‑only DAL** & **Add‑only migrations.**
* **Four environments:** **Alpha → Beta → RTM (validates on Prod DB) → Prod**.
* **GitHub‑first:** branch protections, merge queue, required checks (Build/Tests, CodeQL, Dependency Review, Secret Scanning), SBOM & ≥1‑year artifact retention.

**Out of Scope.** Persist/transform MCP payloads; storing secrets in DB; full admin console beyond read‑only ops UI (optional).

**2) Architecture Narrative (Executive Overview)**

MCPX‑KendoBridge is a stateless **.NET 8** Web API that **spawns and supervises** one **KendoReact MCP** child process per session, forwarding JSON‑RPC over STDIO and returning either **JSON** or **SSE** streams. Configuration (child command/args, SSE keep‑alive, Origin allow‑list, timeouts) is pulled via **stored procedures** from SQL Server. The system exposes liveness/readiness and a **redacted** effective configuration endpoint to simplify ops. Promotion is GitHub‑driven across **Alpha → Beta → RTM → Prod**, with RTM validating on **Prod DB**. Secrets (SQL connection, Telerik license) are configured **only** in GitHub Environments or vendor portals.

**3) Context View (C4‑Level 1)**

**Scope.** External actors and high‑level dependencies, including database and optional Ops UI.

**Diagram (Mermaid source).**

flowchart LR

Client[Remote MCP Client\n(ChatGPT/MyGPT Connector)] -- HTTP/Streamable-HTTP + SSE --> Proxy[Our MCP Proxy API\n(.NET 8)]

Legacy[Legacy MCP Client\n(optional)] -- HTTP+SSE (flag) --> Proxy

Proxy -- spawn via STDIO --> Kendo[@progress/kendo-react-mcp\nchild process]

Proxy -- SP-only DAL --> SQL[(SQL Server)]

Admin[Ops Admin UI (optional)\nKendoReact Fluent v12] -- HTTPS (read-only) --> Proxy

**Narrative.**

* **Remote MCP Client**: initiates requests/streams.
* **Proxy API**: isolates sessions; brokers requests; enforces Origin allow‑list; emits health/metrics.
* **Kendo MCP child**: STDIO‑based black‑box.
* **SQL Server**: authoritative config & feature flags via SPs; **no secrets**.
* **Ops UI (optional)**: read‑only health/metrics/config using KendoReact (Fluent v12 + ThemeBuilder).

**DB COMPLIANCE Banner.**  
Add‑only schema. **Stored‑procedure‑only** access. **No‑Hard‑Coding**. Config via DB (AppConfig, FeatureFlag, Lookup) and SPs (sp\_Config\_GetValue, sp\_Feature\_IsEnabled, sp\_Lookup\_Get).

**4) Container View (C4‑Level 2)**

**Containers.**

* **API (.NET 8 Web API)**: Transport controllers, session registry, STDIO bridge, config provider, observability.
* **Child Process (@progress/kendo‑react‑mcp)**: JSON‑RPC over STDIO.
* **SQL Server**: SP‑backed config & flags.
* **Optional Ops UI (React + KendoReact)**: Read‑only dashboards.

**Diagram (Mermaid source).**

flowchart TB

subgraph API[.NET 8 Web API]

Cfg[Config Provider\n(SP-only)]

Sess[Session Manager]

Brg[STDIO Bridge]

Tx[Transport: /mcp, /messages, /sse]

Obs[Logs/Health/Metrics]

end

Cfg --> SQL[(SQL Server)]

Sess --> Brg --> MCP[@progress/kendo-react-mcp\n(child)]

Tx --> Sess

AdminUI[Ops UI (optional)] --> API

**Key responsibilities.**

* **Transport** negotiates JSON vs SSE; validates headers (Origin, Mcp‑Session‑Id, Accept).
* **Session Manager** ensures one child per session; lifecycle & graceful drain.
* **STDIO Bridge** writes requests to stdin, parses stdout for SSE/message framing.
* **Config Provider** retrieves **all** dynamic values via SPs; caches with TTL for resilience.
* **Observability** provides /healthz, /ready, metrics, structured logs.

**5) Component View (C4‑Level 3 — API internals)**

**Diagram (Mermaid source).**

flowchart LR

Ctrl[/MCP Controller/] --> Transport

Transport --> SessionRegistry

SessionRegistry --> StdioBridge

StdioBridge --> ChildProc

Ctrl --> Healthz[/Health/Ready/Config/]

ConfigProvider --> SQL[(SPs)]

subgraph Observability

Logging --> JSONLogs

Metrics --> Counters

end

**Component responsibilities.**

* **MCP Controller.** Implements POST /mcp (JSON vs SSE) and GET /mcp (SSE subscribe).
* **Transport Module.** Frames SSE events (event: message, id, data), heartbeats every Network:SseKeepAliveSeconds.
* **Session Registry.** Maps Mcp‑Session‑Id → child PID; enforces one‑to‑one; handles cleanup.
* **STDIO Bridge.** Async read/write with backpressure; request‑scoped cancellation.
* **Health/Ready/Config.** Liveness; readiness includes DB reachability and optional child spawn; config returns **redacted** key/values from DB.
* **Config Provider.** Strict **SP‑only** DAL; 30‑second command timeout; no inline SQL.

**6) Deployment View (Environments & Promotion)**

**Environments.** **Alpha → Beta → RTM → Prod** with **RTM validating on Prod DB** (read‑only). OpenAPI servers list all four; GitHub Environments hold secrets and approvals.

**Promotion gates (merge‑queue aware).** Build & tests, **CodeQL**, **Dependency Review**, **Secret Scanning**, **SBOM**, OpenAPI lint/diff, evidence attachments (retention ≥1 year).

**Ingress expectations.** SSE passthrough for text/event-stream; disable proxy buffering; send keep‑alive comments at configured interval.

**7) Data‑Flow Diagrams & Trust Boundaries**

**7.1 DFD‑Level 0 (System as a whole)**

**Mermaid source.**

flowchart LR

subgraph Internet

Client[Remote MCP Client]

end

subgraph DC[Technijian Cloud / VPC]

API[Proxy API]

DB[(SQL Server)]

K[@progress/kendo-react-mcp child]

end

Client -- HTTP/SSE --> API

API -- SP calls --> DB

API -- STDIO --> K

K -- stdout notifications --> API

API -- SSE --> Client

**Trust boundaries.**

* **Boundary A (Internet ↔ API).** Enforce CORS/Origin and authentication (bearer at gateway/app).
* **Boundary B (API ↔ DB).** Controlled network; SP‑only access.
* **Boundary C (API ↔ Child Process).** Local host/process boundary; sanitize logs; supervise PIDs.

**7.2 DFD‑Level 1 (Detailed flows)**

* **F1 — Request (JSON).** Client → POST /mcp (JSON) → API validates Origin → SP lookups (timeouts, child launch cfg) → API → child stdin → child stdout → API → JSON response.
* **F2 — Request (SSE).** Client → POST /mcp with Accept: text/event-stream → API streams SSE; heartbeats every Network:SseKeepAliveSeconds.
* **F3 — Background Notifications.** Client → GET /mcp SSE subscribe → child notifications → API broadcasts SSE message events.
* **F4 — Health/Ready.** Client/Monitor → /healthz & /ready; readiness verifies DB and (optionally) child spawn.
* **F5 — Effective Config.** Ops → /config/effective → redacted view from DB (sp\_Config\_GetAll).
* **F6 — Legacy Transport.** (optional) /messages and /sse gated by EnableLegacyHttpSse.

**8) STRIDE Threat Model & Mitigations**

| **STRIDE** | **Threat Example (by boundary/flow)** | **Impact** | **Mitigations / Controls** |
| --- | --- | --- | --- |
| **S**poofing | Client identity spoofing on /mcp | Unauthorized use | Bearer auth (platform) + CORS Origin allow‑list from DB; reject on 403; correlation IDs in logs. |
| **T**ampering | Altered SSE frames by intermediaries | Corrupted stream | TLS termination at trusted edges; no caching; event framing with id sequencing; logs for anomalies. |
| **R**epudiation | Client denies sending a request | Audit gaps | JSON logs with requestId, sessionId, childPid; evidence retention ≥1 year. |
| **I**nformation Disclosure | Secrets leak in logs or /config/effective | Credential exposure | Zero‑secret logging; redaction; secrets only in GitHub Environments/vendor portals (never DB). |
| **D**enial of Service | Abusive streams or many sessions | Exhaustion | Per‑replica session caps; timeouts (Network:RequestTimeoutSeconds); HPA scale‑out; graceful shutdown. |
| **E**levation of Privilege | Using legacy endpoints to bypass controls | Policy bypass | Gate /messages and /sse with sp\_Feature\_IsEnabled; default off; change via audited migration. |

**DB COMPLIANCE:** Add‑only schema; **Stored‑procedure‑only** access; **No‑Hard‑Coding**. Place this banner in code/docs reviews.

**9) Runtime Sequences (Happy‑Path & Exceptions)**

**9.1 Initialize (happy path)**

sequenceDiagram

participant C as Client

participant P as Proxy API

participant DB as SQL (SPs)

participant K as Kendo MCP (child)

C->>P: POST /mcp (initialize)

P->>DB: sp\_Config\_GetValue / IsEnabled

P->>K: spawn child (command/args from DB)

K-->>P: ready via STDIO

P-->>C: 200 + Mcp-Session-Id

**9.2 Tool call streaming**

sequenceDiagram

participant C as Client

participant P as Proxy API

participant K as Child (STDIO)

C->>P: POST /mcp (Accept: text/event-stream)

P->>K: write JSON-RPC over STDIO

K-->>P: stdout lines/chunks

P-->>C: SSE events (message, id:1..N) + heartbeats

P-->>C: SSE end on final response

**9.3 Background notification**

sequenceDiagram

participant C as Client

participant P as Proxy API

participant K as Child (STDIO)

C->>P: GET /mcp (SSE subscribe; Mcp-Session-Id)

K-->>P: notification on STDIO

P-->>C: SSE "message" to session subscribers

**9.4 Origin denied (exception)**

sequenceDiagram

participant C as Client (disallowed Origin)

participant P as Proxy API

participant DB as SQL (SPs)

C->>P: POST /mcp (Origin: https://evil.example)

P->>DB: sp\_Config\_GetValue("Security:AllowedOrigins")

P-->>C: 403 { code: "origin\_forbidden" }

**9.5 Graceful shutdown (drain)**

* Receive termination signal; stop new requests.
* Continue SSE heartbeats; finish in‑flight streams.
* Terminate child processes; emit final metrics; exit.

**10) Interfaces & Contracts (API, Headers, DB, Config)**

**API (OpenAPI 3.1).** api/openapi/mcp-proxy.yaml defines:

* **Servers:** alpha/beta/rtm/prod; bearer auth; standard error envelope.
* **Transport:** POST /mcp (JSON vs SSE), GET /mcp (SSE), legacy /messages + /sse (flagged).
* **Ops:** /healthz, /ready, /config/effective (redacted).

Note: OpenAPI lint/diff is a required check; servers reflect 4 environments; **No‑Hard‑Coding** annotation included.

**Canonical headers.** Mcp‑Session‑Id (in/out), MCP‑Protocol‑Version (optional), Accept, Origin, Content‑Type.

**DB Contracts (SP‑only).**

* **Tables:** AppConfig([Key] PK, [Value], [UpdatedAt]), FeatureFlag([Name] PK, [Enabled], [UpdatedAt]).
* **SPs:** sp\_Config\_GetValue(@Key), sp\_Config\_GetAll(), sp\_Feature\_IsEnabled(@Name), sp\_Lookup\_Get(@Type,@Key) (reserved).
* **Seeds:** Mcp:ChildCommand=npx, Mcp:ChildArgs=-y @progress/kendo-react-mcp@latest, Mcp:ChildCwd="", Security:AllowedOrigins=https://chat.openai.com,https://platform.openai.com, Network:SseKeepAliveSeconds=15, Network:RequestTimeoutSeconds=120.

**Error envelope.** { code: string; message: string; requestId?: string } (stable across endpoints).

**11) Observability (Health, Readiness, Metrics, Logs)**

* **Health:** /healthz returns ok/fail, uptime, session/child counts.
* **Readiness:** /ready validates DB SP reachability and (optionally) child spawn.
* **Metrics:** session\_count, child\_up, child\_restart\_count, latency summaries (p50/p95).
* **Logs:** Structured JSON with requestId, sessionId, childPid; **no secrets/PII**.

**Evidence.** Monitoring snapshots and logs are attached to Releases; retain ≥1 year.

**12) Performance, Capacity & Scalability**

* **Latency budgets:** Non‑streaming P50 ≤ 300 ms, P95 ≤ 800 ms (intra‑VPC). **Streaming TTFB ≤ 200 ms**.
* **Concurrency:** Target ≥ 200 concurrent sessions per replica; the system is **CPU‑bound before memory**.
* **Scale‑out:** Horizontal via HPA; verify SSE stability and ingress config.
* **Caching:** Config values cached with short TTL to mitigate DB blips (do not cache secrets).

**13) Fault Model & Resilience Patterns**

| **Failure** | **Detection** | **Handling** | **Evidence** |
| --- | --- | --- | --- |
| DB unavailable | /ready fails | Backoff; serve 503 for sensitive ops; alert Ops | Readiness logs/metrics |
| Child spawn failure | Error envelope | Retry with jitter; mark session unhealthy | Error logs with childPid null |
| SSE cut by proxy | Client disconnects | Heartbeats + reconnect logic on client side | Stream termination logs |
| Session leak | Session age/idle | Reap via TTL; drain streams | Session gauges & alerts |
| High latency | p95 monitors | Scale‑out; profile CPU | Monitoring snapshot (release evidence) |

**14) Compliance & Evidence (ASVS, CI/CD)**

* **ASVS highlights:** V2 (Auth), V4 (Access Control via Origin), V5 (Input Validation), V9 (Transport), V14 (Config).
* **CI/CD required checks:** Build/Tests, **CodeQL**, **Dependency Review**, **Secret Scanning**, **SBOM** generation; OpenAPI lint/diff.
* **Evidence Pack:** Test results, CodeQL SARIF, secret‑scan summary, SBOM, OpenAPI diff, monitoring snapshot; **retain ≥1 year**.

**15) Architecture Decisions (ADR Index)**

* **ADR‑0001 — Transport Choice:** Streamable‑HTTP + SSE.
* **ADR‑0002 — Legacy Endpoints Flag:** /messages and /sse behind EnableLegacyHttpSse.
* **ADR‑0003 — Session Model:** One child per Mcp‑Session‑Id.
* **ADR‑0004 — No‑Hard‑Coding & SP‑Only DAL:** All dynamic values from DB SPs.
* **ADR‑0005 — RTM Validates on Prod DB:** Prevent drift pre‑Prod.

**16) Traceability (FR/NFR ↔ Views)**

| **FR/NFR** | **View(s) Proving Compliance** |
| --- | --- |
| FR‑001 (Transport) | Context, Container, Component, Sequences 9.2/9.3 |
| FR‑004/005 (Child spawn/bridge) | Container, Component, Sequences 9.1 |
| FR‑006 (Sessioning) | Component (Session Registry), Sequences |
| FR‑007 (Origin allow‑list) | DFD Boundaries, Sequences 9.4, Threat model |
| FR‑008/009 (Health/Config) | Observability §11, Interfaces §10 |
| FR‑011 (Error envelope) | Interfaces §10, Sequences |
| NFR‑001/002 (Perf) | §12 budgets, Sequences |
| NFR‑003 (Availability) | §13 fault model; readiness |
| NFR‑006 (Security) | Threat model §8; DB compliance banners |
| CI/CD gates | §14 compliance & evidence |

**17) Assumptions**

1. Ingress supports **SSE** without buffering; text/event-stream is passed through.
2. SQL Server is reachable from all environments; migrations run in pipeline; **SP‑only** enforced.
3. Authentication via platform bearer tokens (gateway or app level).
4. Optional Ops UI is **read‑only** and can be delivered after backend GA.
5. RTM uses **Prod DB** (read‑only) to validate configuration parity.

**18) Next Steps**

1. Confirm environment hostnames and update OpenAPI servers (alpha/beta/rtm/prod).
2. Implement session manager, STDIO bridge, SSE writer with keep‑alives; wire **SP‑only** config provider.
3. Finalize health/readiness/config endpoints; instrument metrics and structured logs.
4. Enable GitHub checks (CodeQL, Dependency Review, Secret Scanning) and SBOM publishing; configure merge queue.
5. Run acceptance and perf tests in Alpha/Beta; validate RTM against Prod DB; assemble Evidence Pack.

**19) Appendices (Mermaid Sources)**

**A. Context (C4‑L1)**

flowchart LR

Client[Remote MCP Client] --> Proxy[.NET 8 Proxy API]

Legacy[Legacy MCP Client] --> Proxy

Proxy --> Kendo[@progress/kendo-react-mcp]

Proxy --> SQL[(SQL Server)]

Admin[Ops UI (optional)] --> Proxy

**B. Container (C4‑L2)**

flowchart TB

subgraph API

Cfg[Config Provider]

Sess[Session Manager]

Brg[STDIO Bridge]

Tx[Transport]

Obs[Observability]

end

Cfg --> SQL[(SQL Server)]

Sess --> Brg --> MCP[@progress/kendo-react-mcp]

Tx --> Sess

**C. Component (C4‑L3)**

flowchart LR

Ctrl[/MCP Controller/] --> Transport

Transport --> SessionRegistry

SessionRegistry --> StdioBridge

StdioBridge --> ChildProc

Ctrl --> Healthz

ConfigProvider --> SQL[(SPs)]

**D. Sequences (Initialize / Streaming / Notification / Origin Denied)**

sequenceDiagram

participant C as Client

participant P as Proxy API

participant DB as SQL

participant K as Kendo MCP

C->>P: POST /mcp

P->>DB: sp\_Config\_\*

P->>K: spawn & write

K-->>P: ready/stdout

P-->>C: 200 + Mcp-Session-Id (or SSE)

**Footer (optional for Word header/footer):**  
*MCPX‑KendoBridge • System Context & Architecture • Version 1.0.0 (Draft) • 2025‑09‑23 • Confidential — Technijian Internal*