**MCPX‑KendoBridge — Actors & Use‑Cases**

**Document:** docs/03\_actors\_usecases.docx  
**Project Code:** MCPX‑KendoBridge  
**Version:** 1.0.0 (Draft)  
**Last Updated:** 2025‑09‑23  
**Document Owner:** DoSE (Accountable) — DocFactory (Responsible)

**Purpose.** Define the **actors**, **use‑case catalog**, and **detailed specifications** (flows, pre/post‑conditions, exceptions, Gherkin acceptance, and sequence diagrams) that drive implementation and testing of the MCPX‑KendoBridge service. This document aligns with Technijian’s **GitHub‑first SDLC**, **four environments (Alpha → Beta → RTM → Prod)**, **No‑Hard‑Coding**, and **SP‑only** database policy.

**Document Control**

**Revision History**

| **Version** | **Date** | **Author/Role** | **Summary of Changes** |
| --- | --- | --- | --- |
| 1.0.0‑D | 2025‑09‑23 | DocFactory (R) | Initial actors, catalog, specs, Gherkin, 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 & System Boundary
2. Actor Definitions
3. Use‑Case Catalog (Overview)
4. Detailed Use‑Case Specifications
5. Gherkin Acceptance Scenarios
6. Sequence Diagrams (Mermaid source)
7. Security, Compliance & Guardrails
8. Traceability Matrix (UC ↔ FR/NFR/Endpoints/SPs)
9. Assumptions
10. Next Steps

**1. Scope & System Boundary**

**In scope (MVP).** Transport endpoints (POST /mcp, GET /mcp), sessioning via Mcp-Session-Id, STDIO child spawn/bridge, SSE streaming with keep‑alives, origin allow‑list enforcement, health/readiness, effective configuration (redacted), structured logs and minimal metrics. **DB is SP‑only; schema changes are add‑only; dynamic values via SPs; no secrets in DB.**

**Out of scope.** Persisting/translating MCP payloads; full admin console beyond read‑only ops; storing license keys in DB; modifying Kendo MCP behavior (treated as a black‑box STDIO server).

**System boundary.** The proxy **does not** interpret JSON‑RPC business semantics—it **brokers** messages/streams between HTTP clients and the child STDIO process, applying security/operability controls.

**2. Actor Definitions**

| **Actor** | **Description** | **Responsibilities** | **Triggers** |
| --- | --- | --- | --- |
| **Remote MCP Client** | Cloud assistant or connector (e.g., ChatGPT/MyGPT) that calls HTTP/Streamable‑HTTP endpoints | Initiate sessions; send JSON‑RPC; request streaming; subscribe to background notifications | Send POST /mcp; open GET /mcp (SSE) |
| **Legacy MCP Client** (optional) | Compatibility client using HTTP+SSE legacy semantics | Uses /messages (POST) and /sse (GET) when feature flag enabled | Feature EnableLegacyHttpSse is true |
| **Kendo MCP Child Process** | The spawned @progress/kendo-react-mcp STDIO server | Produce JSON‑RPC responses and notifications on stdout; read requests from stdin | Spawn on first request or explicit initialize |
| **Ops Admin** | Read‑only operational persona | Observe health/readiness/metrics; read effective (redacted) config; approve promotions Alpha → Prod | Manual checks or CI/CD promotion gates |

**Policy reminders (apply to all actors):** **No‑Hard‑Coding**; dynamic values from DB via SPs; **SP‑only** DAL; secrets configured **only** in GitHub Environments/vendor portals; **RTM validates on Prod DB**.

**3. Use‑Case Catalog (Overview)**

| **ID** | **Title** | **Primary Actor** | **Summary** |
| --- | --- | --- | --- |
| **UC‑01** | Establish Session & Receive Mcp-Session-Id | Remote MCP Client | First POST /mcp spawns session child (if needed) and returns/echoes Mcp-Session-Id. |
| **UC‑02** | Streamed Tool Call (SSE) | Remote MCP Client | Client requests Accept: text/event-stream; proxy streams child stdout as SSE events. |
| **UC‑03** | Background Notification via SSE | Remote MCP Client | Client opens GET /mcp SSE; receives child notifications for the session. |
| **UC‑04** | Deny Disallowed Origin | Proxy | Requests with Origin not in Security:AllowedOrigins are rejected with 403 and error envelope. |
| **UC‑05** | Read Health & Readiness | Ops Admin | Query /healthz and /ready for liveness/dependency checks. |
| **UC‑06** | View Effective Config (Redacted) | Ops Admin | Read current non‑secret config via /config/effective (sourced from DB). |
| **UC‑07** | Graceful Shutdown & Drain | Proxy | Drain SSE, terminate child processes during shutdown. |
| **UC‑08** | Legacy HTTP+SSE (Feature‑Flagged) | Legacy Client | Use /messages and /sse when EnableLegacyHttpSse is true. |
| **UC‑09** | Promotion Alpha → Beta → RTM → Prod | Ops Admin | Promote via GitHub Environments with required checks and evidence. |
| **UC‑10** | Rotate Telerik License Secret | Ops Admin | Rotate TELERIK\_LICENSE\_PATH/TELERIK\_LICENSE in environment; verify readiness. |

**4. Detailed Use‑Case Specifications**

**UC‑01 — Establish Session & Receive Mcp-Session-Id**

**Primary Actor:** Remote MCP Client  
**Stakeholders:** Ops Admin (observability), Security (origin allow‑list)  
**Pre‑conditions:**

* Service up; DB reachable (for config SPs).
* Request Origin is in Security:AllowedOrigins.  
  **Post‑conditions:**
* A session exists with a dedicated child process; Mcp-Session-Id echoed/issued.  
  **Main Flow:**

1. Client sends POST /mcp with a valid JSON‑RPC 2.0 message.
2. Proxy loads config via sp\_Config\_GetValue and sp\_Feature\_IsEnabled.
3. If no session exists, proxy **spawns** child using DB‑sourced Mcp:ChildCommand, Mcp:ChildArgs, Mcp:ChildCwd.
4. Proxy forwards request to child (stdin).
5. Proxy returns JSON response (non‑streaming) with header Mcp-Session-Id.  
   **Alternate Flows:**

* **AF‑1:** Client supplies Mcp-Session-Id header → proxy binds to existing session.
* **AF‑2:** DB temporarily unavailable → readiness fails; return 503 with error envelope.  
  **Exceptions:**
* **E‑1 (Origin Forbidden):** Return 403 { code: "origin\_forbidden", ... }.
* **E‑2 (Child Spawn Failure):** Return 500 { code: "spawn\_failed", ... }.  
  **NFR References:** Latency P50≤300 ms/P95≤800 ms; availability 99.9%; zero secret logging.

**UC‑02 — Streamed Tool Call (SSE)**

**Primary Actor:** Remote MCP Client  
**Pre‑conditions:** Valid session exists; client sets Accept: text/event-stream.  
**Post‑conditions:** Stream closed after final response; heartbeats sent per keep‑alive config.  
**Main Flow:**

1. Client POST /mcp with Accept: text/event-stream.
2. Proxy forwards JSON‑RPC to child; begins **SSE** output with incremental id.
3. For each stdout line/chunk from child, proxy emits event: message, id: <n>, data: <chunk>.
4. Proxy emits heartbeat comments : <timestamp> every Network:SseKeepAliveSeconds.
5. On final child response, proxy sends terminal SSE event and closes stream.  
   **Alternate Flows:**

* **AF‑1:** Client cancels request; proxy closes stream and detaches from child.
* **AF‑2:** Backpressure at client → stream remains open; proxy flushes each event, no buffering beyond line/chunk.  
  **Exceptions:**
* **E‑1 (Timeout):** If processing exceeds Network:RequestTimeoutSeconds, proxy closes stream with { code: "timeout" } in trailing event (implementation specific).  
  **NFR References:** Streaming TTFB ≤200 ms; P95≤800 ms for non‑streaming; ≥200 concurrent sessions/replica.

**UC‑03 — Background Notification via SSE**

**Primary Actor:** Remote MCP Client  
**Pre‑conditions:** Session exists; client opens GET /mcp SSE with Mcp-Session-Id.  
**Post‑conditions:** Client receives asynchronous notifications emitted by the child process.  
**Main Flow:**

1. Client invokes GET /mcp with header Mcp-Session-Id.
2. Proxy subscribes the connection to the session’s notification bus.
3. When child writes notification to stdout, proxy broadcasts as SSE event: message to all subscribers of the session.
4. Heartbeats continue at configured interval.  
   **Alternate Flows:**

* **AF‑1:** Multiple subscribers per session supported (fan‑out).
* **AF‑2:** Client disconnects; proxy prunes subscriber.  
  **Exceptions:**
* **E‑1:** Missing/invalid session → 400 { code: "invalid\_session" }.

**UC‑04 — Deny Disallowed Origin**

**Primary Actor:** Proxy (enforcement); **External Actor:** Remote Client (violating)  
**Pre‑conditions:** Origin header not in Security:AllowedOrigins (DB).  
**Main Flow:**

1. Client request hits /mcp (or any endpoint).
2. Proxy loads/uses allow‑list from DB and denies request.
3. Response 403 with error envelope { code: "origin\_forbidden", message: "...", requestId }.  
   **Non‑Functional:** Decision latency negligible; ensure **no** secret leakage in logs.

**UC‑05 — Read Health & Readiness**

**Primary Actor:** Ops Admin  
**Main Flow:**

1. Call GET /healthz for liveness status (process up, minimal counters).
2. Call GET /ready to verify DB access (SP calls) and child spawn check (optional).  
   **Outputs:** Health object with status, uptimeSeconds, sessionCount, childProcesses.

**UC‑06 — View Effective Config (Redacted)**

**Primary Actor:** Ops Admin  
**Main Flow:**

1. Call GET /config/effective.
2. Server returns non‑secret key/value pairs merged from DB. Secrets are redacted/excluded.  
   **Notes:** Used to diagnose environment drift; **No‑Hard‑Coding**—values come from DB SPs.

**UC‑07 — Graceful Shutdown & Drain**

**Primary Actor:** Proxy (internal); **Stakeholder:** Remote Clients  
**Main Flow:**

1. Termination signal received.
2. Proxy stops accepting new requests; holds open existing **SSE** streams to drain.
3. After grace period, terminates child processes per session; reports metrics.  
   **Post‑conditions:** No orphan PIDs; consistent logs of shutdown sequence.

**UC‑08 — Legacy HTTP+SSE (Feature‑Flagged)**

**Primary Actor:** Legacy Client  
**Pre‑conditions:** Feature flag EnableLegacyHttpSse is true (DB).  
**Main Flow:**

1. Client uses POST /messages for JSON requests.
2. Client opens GET /sse for notifications.  
   **Exceptions:**

* Flag disabled → 403 { code: "feature\_disabled" }.

**UC‑09 — Promotion Alpha → Beta → RTM → Prod**

**Primary Actor:** Ops Admin  
**Main Flow:**

1. Approve deployment in GitHub Environment alpha; validate /ready.
2. Promote to beta; review metrics.
3. Promote to **RTM** (validates on **Prod DB**); confirm parity.
4. Approve **Prod** and execute 24‑hour post‑release checks.  
   **Evidence:** Test reports, CodeQL SARIF, Dependency Review, Secret Scanning summary, SBOM, OpenAPI diff, monitoring snapshot (retain ≥1 year).

**UC‑10 — Rotate Telerik License Secret**

**Primary Actor:** Ops Admin  
**Main Flow:**

1. Update TELERIK\_LICENSE\_PATH/TELERIK\_LICENSE in target GitHub Environment or vendor portal.
2. Roll pods; verify /ready and a basic MCP round‑trip.  
   **Compliance:** Secrets **never** stored in DB or code.

**5. Gherkin Acceptance Scenarios**

The following scenarios are **copy‑ready** for /tests/gherkin/ and map to the MVP use‑cases.

**5.1 UC‑01 — Establish Session**

Feature: Establish MCP session

Scenario: Client receives Mcp-Session-Id on first request

Given a Remote MCP Client without a session id

When it POSTs to /mcp with a valid JSON-RPC message

Then the response status is 200

And the response header "Mcp-Session-Id" is present

And the body is a valid JSON-RPC 2.0 response

**5.2 UC‑02 — Streamed Tool Call**

Feature: Streamed tool call response

Scenario: Client requests SSE streaming

Given an existing Mcp-Session-Id

When the client POSTs to /mcp with Accept "text/event-stream"

Then the server responds with "text/event-stream"

And SSE events named "message" are received with incremental ids

And heartbeat comments are emitted every configured keepalive seconds

**5.3 UC‑03 — Background Notification**

Feature: Background notifications

Scenario: Client subscribes to notifications

Given an existing Mcp-Session-Id

And a GET /mcp SSE channel is open

When the child process emits a notification

Then the client receives an SSE "message" with the notification payload

**5.4 UC‑04 — Origin Denied**

Feature: Disallowed Origin is rejected

Scenario: Request from an Origin not in the allow-list

Given the Origin header is "https://evil.example"

When the client POSTs to /mcp

Then the response status is 403

And the error envelope contains code "origin\_forbidden"

**5.5 UC‑05 — Health/Ready**

Feature: Health and readiness

Scenario: Ready when DB and child are OK

When a client GETs /ready

Then the status is 200

And the payload includes fields "status", "sessionCount", "childProcesses"

**5.6 UC‑06 — Effective Config (Redacted)**

Feature: Effective configuration

Scenario: Ops reads redacted configuration

When a client GETs /config/effective

Then the status is 200

And the payload contains known non-secret keys

And no secret-looking values are present

**5.7 UC‑08 — Legacy Endpoints Feature Flag**

Feature: Legacy HTTP+SSE endpoints are gated

Scenario: Legacy endpoints disabled by default

When a client GETs /sse

Then the response status is 403

And the error envelope contains code "feature\_disabled"

**6. Sequence Diagrams (Mermaid source)**

Paste these code blocks into a diagram tool that supports Mermaid, or include the code snippets in the repo for reference.

**6.1 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 (JSON-RPC init)

P->>DB: sp\_Config\_GetValue / sp\_Feature\_IsEnabled

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

K-->>P: ready via STDIO

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

**6.2 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

**6.3 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 all session subscribers

**6.4 Origin Denied**

sequenceDiagram

participant C as Client (evil.example)

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", ... }

**7. Security, Compliance & Guardrails**

* **CORS/Origin allow‑list** enforced from DB key Security:AllowedOrigins; violations return 403 with standard error envelope.
* **Secrets:** Never placed in code/docs/DB. Configure only in **GitHub Environments** or vendor portals (e.g., TELERIK\_LICENSE\_PATH, TELERIK\_LICENSE, SQL connection).
* **DB COMPLIANCE:** **Add‑only** schema, **Stored‑procedure‑only** access (sp\_Config\_\*, sp\_Feature\_IsEnabled, sp\_Lookup\_Get), **No‑Hard‑Coding** (dynamic values come from DB or vault).
* **Evidence Retention:** CI/CD artifacts (tests, CodeQL SARIF, Dependency Review, Secret Scanning, SBOM, OpenAPI diff, monitoring snapshot) retained ≥ **1 year** per Release.

**8. Traceability Matrix (UC ↔ FR/NFR/Endpoints/SPs)**

| **UC ID** | **Primary Endpoints** | **Key Headers** | **SPs / Tables** | **FR Mapping** | **NFR Mapping** |
| --- | --- | --- | --- | --- | --- |
| UC‑01 | POST /mcp | Mcp-Session-Id, Origin | sp\_Config\_GetValue, sp\_Feature\_IsEnabled; AppConfig, FeatureFlag | FR‑001, FR‑004, FR‑005, FR‑006, FR‑007, FR‑011 | NFR‑001, NFR‑003, NFR‑005, NFR‑006 |
| UC‑02 | POST /mcp (SSE) | Accept: text/event-stream | sp\_Config\_GetValue | FR‑001, FR‑005, FR‑006, FR‑010 | NFR‑001, NFR‑002, NFR‑005 |
| UC‑03 | GET /mcp (SSE) | Mcp-Session-Id | sp\_Config\_GetValue | FR‑001, FR‑006, FR‑010 | NFR‑002, NFR‑005 |
| UC‑04 | any | Origin | sp\_Config\_GetValue (Security:AllowedOrigins) | FR‑007, FR‑011 | NFR‑006 |
| UC‑05 | /healthz, /ready | — | sp\_Config\_GetValue (optional), child probe | FR‑008 | NFR‑007 |
| UC‑06 | /config/effective | — | sp\_Config\_GetAll | FR‑009 | NFR‑007, NFR‑009 |
| UC‑07 | — | — | — | FR‑012 | NFR‑003 |
| UC‑08 | /messages, /sse | Mcp-Session-Id | sp\_Feature\_IsEnabled | FR‑003 | NFR‑006 |
| UC‑09 | (CI/CD) | — | — | FR‑014 | NFR‑009 |
| UC‑10 | — | — | — | FR‑014 (secrets governance) | NFR‑006, NFR‑009 |

FR/NFR numbering aligns with the project’s FR/NFR specification and CI/CD guardrails.

**9. Assumptions**

1. Ingress supports **SSE** and streaming without buffering lines (proper text/event-stream handling).
2. SQL Server reachable from all environments; migrations run in CI/CD; SP‑only DAL (SqlCommand + CommandType.StoredProcedure).
3. Authentication is provided by the platform (bearer); our endpoints assume bearer unless fronted by gateway.
4. **RTM** environment validates against **Prod DB** in read‑only mode to catch config drift.
5. Optional Ops UI is read‑only and **non‑blocking** for backend release.

**10. Next Steps**

1. Implement session manager and STDIO bridge; wire config provider to sp\_Config\_\* and flags to sp\_Feature\_IsEnabled.
2. Implement /mcp POST/GET with content negotiation, SSE writer with keep‑alives, and graceful shutdown hooks.
3. Add origin enforcement from Security:AllowedOrigins; standardize error envelope.
4. Complete health/readiness and effective config endpoints; instrument metrics.
5. Finalize acceptance tests (Gherkin above) and perf tests against NFR budgets; attach evidence to releases.

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