**ADR‑0004 — No‑Hard‑Coding, SP‑Only Data Access, and Add‑Only Migrations**

**Document:** docs/adr/0004-no-hard-coding-sp-only-db.docx  
**Status:** **Accepted**  
**Date:** 2025‑09‑27  
**Project:** MCPX‑KendoBridge  
**Deciders:** DoSE (Accountable), DBA Lead, SRE Lead, Dev Lead, Security Lead  
**Consulted:** T‑Arch, QA Lead, DocFactory  
**Tags:** database, configuration, migrations, security, compliance, DAL, SP‑only

**Guardrails (non‑negotiable):** **GitHub‑first** SDLC; four environments **Alpha → Beta → RTM (validates on Prod DB read‑only) → Prod**; **Add‑only** schema; **Stored‑procedure‑only** data access; **No‑Hard‑Coding** of dynamic values; **secrets only** in GitHub Environments or platform vaults. These rules are the DocFactory defaults for Technijian projects.

**1) Context**

MCPX‑KendoBridge requires deterministic, auditable runtime behavior across environments while preventing configuration drift and secret leakage. The proxy must read **all dynamic behavior** (child command/args/cwd, request timeouts, SSE keep‑alive cadence, Origin allow‑list, feature flags) from **SQL Server via stored procedures**, never from literals in code or environment files. DB changes must be **forward‑compatible (add‑only)** to support rolling deploys and RTM validation on **Prod DB (read‑only)**.

**2) Decision**

Adopt the following **binding policies** for the service and repository:

1. **No‑Hard‑Coding**
   * Runtime‑varying values **must** originate from DB **AppConfig/FeatureFlag/Lookup** tables via SPs:  
     sp\_Config\_GetValue, sp\_Config\_GetAll, sp\_Feature\_IsEnabled, sp\_Lookup\_Get.
   * Source code **must not** contain literals for child process settings, network timeouts, SSE keep‑alive, or origin allow‑list.
   * Secrets (SQL connection strings, Telerik license) **never** in code/DB/docs/logs; they are **only** configured in **GitHub Environments**.
2. **Stored‑Procedure‑Only (SP‑only) DAL**
   * The application principal has **EXECUTE‑only** on whitelisted SPs; **no** table/view CRUD rights.
   * All reads come from SPs; **no inline SQL** in application code.
   * Ownership chaining preserved by keeping SPs/tables under dbo.
3. **Add‑Only Migrations**
   * Schema and SP evolution is **forward‑additive** (add columns/SPs/nullable params) and **non‑destructive**.
   * Backward‑incompatible or destructive changes are prohibited in regular releases; deprecation requires a plan and major version window.
   * RTM must validate against **Prod DB (read‑only)**; migrations may not be applied in RTM.

These decisions are enforced by CODEOWNERS, PR templates, CI checks, and the DB grants script.

**3) Scope & Non‑Goals**

* **Scope:** API service configuration plane, DAL, migrations, SP grants, and operational evidence.
* **Non‑Goals:** Introducing runtime secret stores in DB; supporting destructive DDL; writing business data (the Admin Portal is read‑only).

**4) Options Considered**

| **Option** | **Pros** | **Cons** |
| --- | --- | --- |
| **No‑Hard‑Coding + SP‑only + Add‑only (chosen)** | Predictable, auditable; safe rolling deploys; RTM parity; minimal blast radius | Requires discipline and CI enforcement; slightly more boilerplate |
| Inline SQL in code | Quick to prototype | Fragile, hard to audit; risks injection/drift; fails RTM parity |
| App reads config from files/env | Simple when small | Drifts across envs; hard to audit; secrets risk |
| Destructive migrations | “Clean” schema | Breaks rolling upgrades; RTM parity impossible |

**5) Rationale**

* **Auditability & Evidence.** Centralizing dynamic behavior in DB (non‑secret) with SP‑only access is easily snapshotted via /config/effective and retained ≥ 1 year in the Evidence Pack.
* **Safety.** Add‑only ensures rolling deploys and avoids “half‑migrated” outages; SP‑only ensures least‑privilege and stable contracts.
* **Consistency.** RTM parity checks against Prod DB (read‑only) catch drift **before** production.

**6) Policy Details (authoritative)**

**6.1 Data model & SP contracts (non‑secret)**

* **Tables** *(add‑only)*
  + AppConfig([Key] PK, [Value], [UpdatedAt])
  + FeatureFlag([Name] PK, [Enabled] BIT, [UpdatedAt])
* **Stored Procedures**
* sp\_Config\_GetValue(@Key NVARCHAR(200)) → NVARCHAR(MAX)
* sp\_Config\_GetAll() → TABLE ([Key],[Value],[UpdatedAt])
* sp\_Feature\_IsEnabled(@Name NVARCHAR(200)) → BIT
* sp\_Lookup\_Get(@Type NVARCHAR(100), @Key NVARCHAR(200)) → NVARCHAR(MAX) -- optional
* **Seed (non‑secret) keys** include Mcp:ChildCommand, Mcp:ChildArgs, Mcp:ChildCwd,  
  Security:AllowedOrigins, Network:SseKeepAliveSeconds, Network:RequestTimeoutSeconds,  
  EnableLegacyHttpSse. (Secrets never seeded.)

**6.2 Grants & permissions**

* App user belongs **only** to role app\_sp\_execute; role has **EXECUTE** on approved SPs.
* App user **must not** have SELECT/INSERT/UPDATE/DELETE on tables/views.
* Grant script is idempotent and **add‑only** (see §8).

**6.3 DAL usage pattern (authoritative)**

* **.NET 8** ADO.NET with **CommandType.StoredProcedure**, **async**, **CancellationToken**, default 30s timeout.
* **No inline SQL** strings.
* **Configuration Provider** reads at startup and refresh intervals (if designed), caching non‑secret values.

**C# sketch (non‑secret; copy/paste):**

public sealed class DbConfigProvider

{

private readonly string \_cs;

public DbConfigProvider(IConfiguration cfg) => \_cs = cfg.GetConnectionString("AppDb")!;

public async Task<string?> GetValueAsync(string key, CancellationToken ct)

{

await using var con = new SqlConnection(\_cs);

await con.OpenAsync(ct);

await using var cmd = new SqlCommand("sp\_Config\_GetValue", con)

{ CommandType = CommandType.StoredProcedure, CommandTimeout = 30 };

cmd.Parameters.Add(new SqlParameter("@Key", SqlDbType.NVarChar, 200) { Value = key });

var result = await cmd.ExecuteScalarAsync(ct);

return result == DBNull.Value ? null : (string?)result;

}

}

**7) CI/CD & Repository Enforcement**

**a) CODEOWNERS (already committed)**

* db/migrations/\*\* and db/stored\_procedures/\*\* require **DBA** + **Security** review.
* .github/workflows/\*\* requires **CI/CD** + **Security** owners.

**b) PR Template (required checks)**

* No‑Hard‑Coding checkbox, SP‑only, add‑only, secrets policy, OpenAPI lint/diff, CodeQL, Dependency Review, Secret Scanning, SBOM — all must be green before merge.

**c) Simple static guard (heuristic)**

# Fail CI if obvious inline SQL detected in application folders (heuristic; safe-list tests/tools)

grep -RIn --include='\*.cs' -E 'SELECT |INSERT |UPDATE |DELETE |CREATE TABLE|ALTER TABLE' src/ api/ \

| grep -v 'db/' && { echo "Inline SQL detected. Use SPs."; exit 1; } || true

**d) Migration naming**

* db/migrations/VYYYYMMDDHHMM\_\_<snake\_case>.sql (monotonic, UTC).
* Never modify prior files; new migrations are append‑only.

**8) Reference SQL (copy/paste)**

**A) Initial schema (add‑only; excerpt)**

-- V202509230900\_\_init\_schema.sql

CREATE TABLE dbo.AppConfig (

[Key] NVARCHAR(200) NOT NULL CONSTRAINT PK\_AppConfig PRIMARY KEY,

[Value] NVARCHAR(MAX) NULL,

[UpdatedAt] DATETIME2 NOT NULL CONSTRAINT DF\_AppConfig\_UpdatedAt DEFAULT SYSUTCDATETIME()

);

CREATE TABLE dbo.FeatureFlag (

[Name] NVARCHAR(200) NOT NULL CONSTRAINT PK\_FeatureFlag PRIMARY KEY,

[Enabled] BIT NOT NULL,

[UpdatedAt] DATETIME2 NOT NULL CONSTRAINT DF\_FeatureFlag\_UpdatedAt DEFAULT SYSUTCDATETIME()

);

**B) Non‑secret seeds**

-- V202509230905\_\_seed\_appconfig\_featureflag.sql

MERGE dbo.AppConfig AS t

USING (VALUES

(N'Mcp:ChildCommand', N'npx', SYSUTCDATETIME()),

(N'Mcp:ChildArgs', N'-y @progress/kendo-react-mcp@latest', SYSUTCDATETIME()),

(N'Mcp:ChildCwd', N'', SYSUTCDATETIME()),

(N'Security:AllowedOrigins', N'https://chat.openai.com,https://platform.openai.com', SYSUTCDATETIME()),

(N'Network:SseKeepAliveSeconds', N'15', SYSUTCDATETIME()),

(N'Network:RequestTimeoutSeconds', N'120', SYSUTCDATETIME())

) AS s([Key],[Value],[UpdatedAt])

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

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

WHEN NOT MATCHED THEN INSERT([Key],[Value],[UpdatedAt]) VALUES(s.[Key],s.[Value],s.[UpdatedAt]);

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 [Enabled]=s.[Enabled],[UpdatedAt]=s.[UpdatedAt]

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

**C) SP bodies**

-- db/stored\_procedures/sp\_Config\_GetValue.sql

CREATE OR ALTER PROCEDURE dbo.sp\_Config\_GetValue

@Key NVARCHAR(200)

AS

BEGIN

SET NOCOUNT ON;

SELECT [Value] FROM dbo.AppConfig WITH (READCOMMITTED) WHERE [Key]=@Key;

END

GO

-- db/stored\_procedures/sp\_Config\_GetAll.sql

CREATE OR ALTER PROCEDURE dbo.sp\_Config\_GetAll

AS

BEGIN

SET NOCOUNT ON;

SELECT [Key],[Value],[UpdatedAt] FROM dbo.AppConfig WITH (READCOMMITTED) ORDER BY [Key];

END

GO

-- db/stored\_procedures/sp\_Feature\_IsEnabled.sql

CREATE OR ALTER PROCEDURE dbo.sp\_Feature\_IsEnabled

@Name NVARCHAR(200)

AS

BEGIN

SET NOCOUNT ON;

SELECT CAST(COALESCE([Enabled],0) AS BIT) FROM dbo.FeatureFlag WITH (READCOMMITTED) WHERE [Name]=@Name;

END

GO

-- db/stored\_procedures/sp\_Lookup\_Get.sql

CREATE OR ALTER PROCEDURE dbo.sp\_Lookup\_Get

@Type NVARCHAR(100),

@Key NVARCHAR(200)

AS

BEGIN

SET NOCOUNT ON;

-- Placeholder for future typed lookups; return NULL if not implemented

SELECT CAST(NULL AS NVARCHAR(MAX)) AS [Value];

END

GO

**D) Grants (role + EXECUTE‑only)** — see V202509271100\_\_grants\_app\_execute\_only.sql from the DB appendix; apply per environment.

**9) Verification & Evidence**

* **Readiness gating:** /ready fails if SPs are unreachable; RTM points to **Prod DB (read‑only)**.
* **Config snapshot:** /config/effective returns non‑secret values; attach per‑env snapshots to the Evidence Pack.
* **CI artifacts:** TRX/coverage, OpenAPI lint/diff, CodeQL SARIF, Dependency Review, Secret‑scan summary, SBOM, migration list.
* **Quarterly audits:** permission query results proving **EXECUTE‑only**; attach outputs to Evidence (retention ≥ 1 year).

**SQL: check role permissions**

SELECT r.name AS role\_name, p.permission\_name, p.state\_desc, o.name AS object\_name, o.type\_desc

FROM sys.database\_permissions p

JOIN sys.database\_principals r ON p.grantee\_principal\_id = r.principal\_id

LEFT JOIN sys.objects o ON p.major\_id = o.object\_id

WHERE r.name = N'app\_sp\_execute'

ORDER BY p.permission\_name, o.name;

**10) Deprecation & Evolution (pattern)**

1. **Add** new nullable column/SP parameter; **do not** drop existing schema.
2. **Backfill** if needed via out‑of‑band job or additive migration.
3. **Update** app & SP to coalesce old/new; ship; verify.
4. **Schedule removal** as a separate **major** with sunset notice; never break RTM/Prod parity.

**11) Risks & Mitigations**

| **Risk** | **Impact** | **Mitigation** |
| --- | --- | --- |
| Developer adds inline SQL | Security & parity regression | CI heuristic; CODEOWNERS/PR checklist; code review |
| Secret committed to repo/DB | Compliance breach | Secret Scanning; repo protections; incident & rotation runbooks; secrets only in Environments |
| Destructive migration | Outage / rollback complexity | Add‑only rule; DBA review; PR checks; ADR scope |
| Config drift | Behavior differs per env | Centralize in DB; /config/effective evidence; RTM parity gates |

**12) Alignment with Other Artifacts**

* **FR/NFR:** FR config surfaces; NFR config‑fetch SLI (p95 ≤ 200 ms), availability, **restart‑to‑ready ≤ 30 s**.
* **OpenAPI 3.1:** /config/effective (non‑secret), error envelope.
* **Runbooks:** deploy/rollback/incident/scale‑out; grants & parity steps.
* **Error Catalog:** canonical codes; **no payload bodies** in logs.
* **DB Grants Appendix:** docs/07a\_db\_grants\_sp\_signatures.docx.
* **Evidence Pack:** artifacts & retention.

**13) Acceptance Criteria**

* No source literals for dynamic values; all read via SPs.
* App principal has **EXECUTE‑only** on approved SPs; **no** table CRUD rights.
* All migrations are **add‑only**; naming follows VYYYYMMDDHHMM\_\_\*.sql.
* /config/effective shows expected non‑secret values per env.
* RTM successfully validates against **Prod DB (read‑only)** without writes.

**14) Backout Plan**

If a migration or SP change introduces regression:

1. **Stop rollout**; keep RTM stable on **Prod DB (read‑only)**.
2. **Revert application image** (no DB rollback) and **Config Rollback** if needed.
3. Ship a **new add‑only** migration/SP version; never destroy or modify prior contracts.
4. Capture incident evidence and post‑mortem.

**15) Appendices**

**A) PR checklist (DB excerpt — copy into PR body)**

* **Add‑only** migration(s); no destructive DDL.
* SPs updated by **adding** optional params only; original signatures preserved.
* App reads values via SPs; **no inline SQL**.
* No secrets in code/DB/evidence; secrets in **GitHub Environments** only.

**B) Environment mapping (principals)**

| **Env** | **DB user (example)** | **Role** |
| --- | --- | --- |
| Alpha | mcp\_proxy\_alpha | app\_sp\_execute |
| Beta | mcp\_proxy\_beta | app\_sp\_execute |
| RTM | mcp\_proxy\_rtm | app\_sp\_execute (connects to **Prod DB RO**) |
| Prod | mcp\_proxy\_prod | app\_sp\_execute |

**Decision record maintained by DocFactory. Changes to these policies require synchronized updates to Data Contracts, DB Grants, PR templates, CI, runbooks, and OpenAPI.**