**MCPX‑KendoBridge — DB Grants & SP Signatures Appendix**

**Document:** docs/07a\_db\_grants\_sp\_signatures.docx  
**Project Code:** MCPX‑KendoBridge  
**Version:** 2.0.0  
**Last Updated:** 2025‑09‑27  
**Owner:** DBA Lead (Responsible) — DoSE (Accountable) — DocFactory (Author) — SecLead, SRE, Dev Lead (Consulted)

**Purpose.** Provide an **audit‑ready** specification of database **principals/roles/permissions** and the **stored‑procedure contracts** that the MCPX‑KendoBridge API is allowed to execute. This appendix operationalizes the Technijian guardrails: **Add‑only schema**, **Stored‑procedure‑only (SP‑only) access**, **No‑Hard‑Coding** of dynamic values (all read via SPs), **four environments** (**Alpha → Beta → RTM → Prod**), and **secrets only** in environment stores.

**DB COMPLIANCE (reiterated):**

* **Add‑only** migrations; never destructive DDL.
* Application principal has **EXECUTE‑only** on whitelisted SPs; **no** table/view CRUD rights.
* **No‑Hard‑Coding** of dynamic behavior: values are read from **AppConfig**/**FeatureFlag** via **sp\_Config\_\***, **sp\_Feature\_IsEnabled**, **sp\_Lookup\_Get**.
* **Secrets** (connection strings, vendor licenses) **never** stored in DB or code; they live only in **GitHub Environments** or platform vaults.

**1) Scope & Environments**

**Scope.** SQL Server database used by MCPX‑KendoBridge for **non‑secret** runtime configuration and feature flags.  
**Environments.** **Alpha → Beta → RTM → Prod**. **RTM** validates against the **Prod DB (read‑only)** to detect drift prior to Prod promotion; this doc and the grants apply identically across environments (principals differ by name).

**2) Principals, Roles & Permissions Model**

**2.1 Roles**

* **app\_sp\_execute (database role):** The only role the application user belongs to. Carries **EXECUTE** grants on the **approved SP list** below. No membership in db\_datareader or db\_datawriter.
* **dbo (owner):** Owns the SPs and the underlying tables to preserve **ownership chaining** and avoid exposing table privileges to the app principal.

**2.2 Application principals (per environment)**

| **Environment** | **Database user (example)** | **Source of password/secret** | **Role membership** |
| --- | --- | --- | --- |
| Alpha | mcp\_proxy\_alpha | **GitHub Environment: alpha** (connection string) | app\_sp\_execute |
| Beta | mcp\_proxy\_beta | **GitHub Environment: beta** | app\_sp\_execute |
| RTM | mcp\_proxy\_rtm | **GitHub Environment: rtm** (uses **Prod DB read‑only**) | app\_sp\_execute |
| Prod | mcp\_proxy\_prod | **GitHub Environment: prod** | app\_sp\_execute |

**Note:** Creation of **logins/users** and injection of credentials are handled **outside** this repo via platform automation; **do not** commit secrets or login scripts.

**3) Whitelisted Stored Procedures & Signatures**

The API can **only** execute the following SPs. Contracts are **stable** and must be versioned via **add‑only** changes (e.g., new optional parameters, new SP names; never change/remove existing parameters).

**3.1 sp\_Config\_GetValue**

* **Purpose:** Return the **non‑secret** value for a configuration key.
* **Signature:**
* sp\_Config\_GetValue(
* @Key NVARCHAR(200)
* ) → NVARCHAR(MAX) -- nullable if key not found
* **Behavior:** Case‑insensitive key match; returns NULL if missing.
* **Callers:** API configuration provider; Admin Portal read‑only viewer.
* **Notes:** Keys are **non‑secret** and may include: Mcp:ChildCommand, Mcp:ChildArgs, Mcp:ChildCwd, Security:AllowedOrigins, Network:SseKeepAliveSeconds, Network:RequestTimeoutSeconds.

**3.2 sp\_Config\_GetAll**

* **Purpose:** List **all non‑secret** configuration pairs.
* **Signature:**
* sp\_Config\_GetAll() → TABLE [Key] NVARCHAR(200), [Value] NVARCHAR(MAX), [UpdatedAt] DATETIME2
* **Behavior:** Returns a complete snapshot; used for /config/effective.
* **Notes:** Must never include secret categories; redaction happens server‑side.

**3.3 sp\_Feature\_IsEnabled**

* **Purpose:** Return whether a feature flag is enabled.
* **Signature:**
* sp\_Feature\_IsEnabled(
* @Name NVARCHAR(200)
* ) → BIT
* **Known flags:** EnableLegacyHttpSse (default **false**). Controls legacy endpoints /messages and /sse (403 feature\_disabled when off).

**3.4 sp\_Lookup\_Get (optional, for future‑proofing)**

* **Purpose:** Fetch typed lookup entries without ad‑hoc SQL.
* **Signature:**
* sp\_Lookup\_Get(
* @Type NVARCHAR(100),
* @Key NVARCHAR(200)
* ) → NVARCHAR(MAX) -- serialization of a lookup value
* **Behavior:** Type‑scoped namespace for generic lookups; implement only when needed.

**4) Effective Permissions (Required State)**

* Application user belongs **only** to app\_sp\_execute.
* Application user has **EXECUTE** on: sp\_Config\_GetValue, sp\_Config\_GetAll, sp\_Feature\_IsEnabled, sp\_Lookup\_Get (if present).
* Application user has **no** SELECT/INSERT/UPDATE/DELETE permissions on any tables or views.
* Ownership chaining remains intact (dbo owns SPs and base tables) so SPs may read tables without granting table rights to the app user.
* No GRANT of CONTROL, ALTER, VIEW DEFINITION on schema objects to the app user.

**5) Migration & Grants Script (Add‑Only)**

**File:** db/migrations/V202509271100\_\_grants\_app\_execute\_only.sql (example timestamp).  
Apply via your standard migration tool. This script is **idempotent** and **add‑only**.

/\* V202509271100\_\_grants\_app\_execute\_only.sql

Purpose: Create EXECUTE-only role and grant SP execution to app role.

Policy: Add-only; no secrets; no destructive DDL.

\*/

SET NOCOUNT ON;

-- 1) Create role if not exists

IF NOT EXISTS (SELECT 1 FROM sys.database\_principals WHERE name = N'app\_sp\_execute' AND type = 'R')

BEGIN

CREATE ROLE [app\_sp\_execute] AUTHORIZATION [dbo];

END

GO

-- 2) Grant EXECUTE on approved SPs to role (idempotent)

DECLARE @sps TABLE(name SYSNAME);

INSERT INTO @sps(name) VALUES

(N'sp\_Config\_GetValue'),

(N'sp\_Config\_GetAll'),

(N'sp\_Feature\_IsEnabled'),

(N'sp\_Lookup\_Get'); -- keep even if not yet created; grant will be retried later

DECLARE @name SYSNAME, @sql NVARCHAR(MAX);

DECLARE c CURSOR LOCAL FAST\_FORWARD FOR SELECT name FROM @sps;

OPEN c;

FETCH NEXT FROM c INTO @name;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF OBJECT\_ID(QUOTENAME('dbo') + '.' + QUOTENAME(@name), 'P') IS NOT NULL

BEGIN

SET @sql = N'GRANT EXECUTE ON OBJECT::dbo.' + QUOTENAME(@name) + N' TO [app\_sp\_execute];';

EXEC sp\_executesql @sql;

END

-- If SP absent, a later migration will create it; re-run grants as needed.

FETCH NEXT FROM c INTO @name;

END

CLOSE c; DEALLOCATE c;

GO

-- 3) (Optional) Associate environment-specific app users to the role

-- DO NOT hard-code secrets; bind existing database users to role.

-- Uncomment and replace with your environment-specific user names.

-- IF EXISTS (SELECT 1 FROM sys.database\_principals WHERE name = N'mcp\_proxy\_alpha' AND type IN ('S','U'))

-- AND NOT EXISTS (SELECT 1 FROM sys.database\_role\_members m

-- JOIN sys.database\_principals r ON m.role\_principal\_id=r.principal\_id

-- JOIN sys.database\_principals u ON m.member\_principal\_id=u.principal\_id

-- WHERE r.name=N'app\_sp\_execute' AND u.name=N'mcp\_proxy\_alpha')

-- BEGIN

-- EXEC sp\_addrolemember @rolename = N'app\_sp\_execute', @membername = N'mcp\_proxy\_alpha';

-- END

-- GO

-- 4) Safety checks (no table CRUD granted to app role)

-- Note: This SELECT is for auditor review; it does not change state.

SELECT r.name AS role\_name, p.permission\_name, 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;

GO

**Auditor note:** The optional sp\_addrolemember statements are **commented** intentionally. Bind environment‑specific users to the role via separate, environment‑scoped automation that uses **GitHub Environment secrets** for connection/auth.

**6) Verification Queries (Copy/Paste)**

**A. Confirm only EXECUTE rights for app role**

SELECT 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;

**Expected:** Rows show **EXECUTE** on approved SPs; **no** SELECT/INSERT/UPDATE/DELETE on tables/views.

**B. Confirm app user has no implicit table rights**

-- Replace with your env user (e.g., mcp\_proxy\_beta)

DECLARE @user SYSNAME = N'mcp\_proxy\_beta';

SELECT pr.name AS principal, pe.permission\_name, pe.state\_desc, obj.name AS object\_name, obj.type\_desc

FROM sys.database\_permissions pe

LEFT JOIN sys.objects obj ON pe.major\_id = obj.object\_id

JOIN sys.database\_principals pr ON pe.grantee\_principal\_id = pr.principal\_id

WHERE pr.name = @user

ORDER BY pe.permission\_name, obj.name;

**Expected:** No table/view CRUD permissions granted directly to the user.

**7) Change Management & Versioning**

* **Add‑only:** Introduce new SPs or optional parameters via new migrations; do **not** change existing signatures.
* **Approvals:** All DB migrations reviewed via **merge queue** with required checks (Build/Tests, CodeQL, Dependency Review, Secret Scanning, SBOM).
* **Evidence:** Attach migration files, grant scripts, and verification outputs to the Release and retain **≥ 1 year**.

**8) Security Considerations**

* **Least privilege:** Application user membership limited to app\_sp\_execute; no db\_datareader/db\_datawriter.
* **Ownership chaining:** Keep SPs and base tables owned by dbo; avoid explicit DENY on base tables (not required when no GRANT exists).
* **Secrets:** Never store connection strings or vendor licenses in the DB; rotate secrets in **GitHub Environments** (see runbook).

**9) Mapping to Specs & Tests**

* **FR/NFR:** SP‑only data access; non‑secret config via SPs; RTM parity on Prod DB (read‑only).
* **OpenAPI:** /config/effective and behavior rely on sp\_Config\_\* and sp\_Feature\_IsEnabled.
* **Tests:** Integration tests must call config surfaces only; **no inline SQL**.
* **Monitoring:** Track config\_fetch\_duration\_ms p95 ≤ 200 ms; alert on sustained regressions.

**10) RACI (DB Grants & Contracts)**

| **Activity** | **A** | **R** | **C** | **I** |
| --- | --- | --- | --- | --- |
| Define DB access model | DoSE | DBA Lead | SecLead, Dev Lead | SRE, QA |
| Author migrations & grants | DoSE | DBA | Dev Lead | QA |
| Validate permissions (per env) | DoSE | DBA + SRE | SecLead | QA |
| Evidence capture (verification outputs) | DoSE | DocFactory | DBA | All |

**11) Assumptions**

1. The application connects with a **single database user per environment** whose password/secret is injected at runtime from **GitHub Environments** (never from DB).
2. **RTM** connects to **Prod DB (read‑only)** to validate parity, invoking only SPs listed here.
3. The Admin Portal is **read‑only** and never writes to the DB.

**12) Next Steps**

* Apply V202509271100\_\_grants\_app\_execute\_only.sql in **Alpha**, verify via §6; promote to **Beta → RTM → Prod**.
* Automate environment‑specific **role membership binding** in deployment tooling (not in repo).
* Schedule quarterly **permissions audits** and attach outputs to the Evidence Pack.

**Footer (optional for Word header/footer):**  
*MCPX‑KendoBridge • DB Grants & SP Signatures • v2.0.0 • 2025‑09‑27 • Confidential — Technijian Internal*