**Document: 07a – DB Grants & SP Signature**

**Project:** MCPX-KendoBridge Admin Portal  
**Document ID:** TJ-MCPX-DOC-07a  
**Version:** 1.0.0  
**Status:** Draft  
**Date:** 2025-09-27  
**Owner:** Database Architect (Technijian)  
**Confidentiality:** Technijian Internal

**Document Control**

| **Version** | **Date** | **Author** | **Change Summary** | **Status** |
| --- | --- | --- | --- | --- |
| 1.0.0 | 2025-09-27 | Database Architect | Initial SP grants matrix, signature policy, evidence SQL | Draft |

**Approvals**

| **Role/Team** | **Name** | **Signature/Date** | **Comment** |
| --- | --- | --- | --- |
| Director of Software Eng. |  |  |  |
| Systems Architect |  |  |  |
| Security/Compliance |  |  |  |
| DBA Lead |  |  |  |

**1. Purpose**

Define **exact SQL permissions** for the API principal (**EXECUTE-only on whitelisted SPs**) and a **signature regime** for stored procedures that lets us (a) detect **breaking changes**, (b) **version** contracts, and (c) export proof into the **Evidence Pack** on every release.

**2. Scope**

* SQL Server 2022; schema dbo.
* SPs in scope (current & planned) for the Admin API:
  + **Present (read/eval):** sp\_Config\_GetAll, sp\_Config\_GetValue, sp\_Feature\_IsEnabled, sp\_Lookup\_Get
  + **Planned (mutations/audit):** sp\_Config\_SetValue, sp\_Feature\_Set, sp\_Lookup\_Upsert, sp\_Audit\_Write
  + **Optional:** sp\_Security\_Origins\_Set (CORS allow-list), sp\_Config\_Diff (parity report)

**3. Principle of Least Privilege (PLP)**

* Application login **app\_mcpx** → database user **app\_mcpx**.
* **DENY** table DML to app\_mcpx.
* **GRANT EXECUTE** only on the SP allow-list below.
* All changes to the allow-list require a **DBA PR** + **Security approval** and create an **AuditEvent**.

**4. Grants Matrix (authoritative)**

| **Stored Procedure** | **Purpose** | **API Access** | **Notes** |
| --- | --- | --- | --- |
| dbo.sp\_Config\_GetAll | Non-secret effective config | ✅ EXEC | Read |
| dbo.sp\_Config\_GetValue | Get config value by key | ✅ EXEC | Read |
| dbo.sp\_Feature\_IsEnabled | Evaluate feature flag | ✅ EXEC | Read |
| dbo.sp\_Lookup\_Get | Read lookup value | ✅ EXEC | Read |
| dbo.sp\_Config\_SetValue | Upsert config key (+ audit) | ✅ EXEC | Write |
| dbo.sp\_Feature\_Set | Set/enable/disable flag (+ audit) | ✅ EXEC | Write |
| dbo.sp\_Lookup\_Upsert | Upsert lookup row (+ audit) | ✅ EXEC | Write |
| dbo.sp\_Audit\_Write | Centralized audit write | ✅ EXEC | Internal |
| dbo.sp\_Security\_Origins\_Set *(opt)* | Manage CORS allow-list | ✅ EXEC | Write |
| dbo.sp\_Config\_Diff *(opt)* | Compute parity diff (RTM↔Prod) | ✅ EXEC | Read |

**Never** grant table-level SELECT/INSERT/UPDATE/DELETE to app\_mcpx.

**5. Provisioning Script (repeatable)**

-- Create app user (login exists separately)

IF NOT EXISTS (SELECT 1 FROM sys.database\_principals WHERE name = 'app\_mcpx')

CREATE USER app\_mcpx FOR LOGIN app\_mcpx;

-- Remove any accidental broad rights

DENY SELECT, INSERT, UPDATE, DELETE, ALTER, CONTROL TO app\_mcpx;

-- Whitelist EXECUTE on approved procedures

DECLARE @procs TABLE (p sysname);

INSERT INTO @procs(p) VALUES

('dbo.sp\_Config\_GetAll'),

('dbo.sp\_Config\_GetValue'),

('dbo.sp\_Feature\_IsEnabled'),

('dbo.sp\_Lookup\_Get'),

('dbo.sp\_Config\_SetValue'),

('dbo.sp\_Feature\_Set'),

('dbo.sp\_Lookup\_Upsert'),

('dbo.sp\_Audit\_Write'),

('dbo.sp\_Security\_Origins\_Set'), -- optional

('dbo.sp\_Config\_Diff'); -- optional

DECLARE @sql nvarchar(max) = N'';

SELECT @sql = STRING\_AGG(N'GRANT EXECUTE ON OBJECT::' + QUOTENAME(p) + N' TO app\_mcpx;', CHAR(10))

FROM @procs WHERE OBJECT\_ID(p) IS NOT NULL;

EXEC sp\_executesql @sql;

Keep this file in /db/grants/VYYYYMMDDHHmm\_\_grant\_exec\_app\_mcpx.sql and include it in release migrations.

**6. SP Signature Policy (contract discipline)**

**6.1 Signature definition**

For each SP, the **signature** consists of:

* **Schema + name** (e.g., dbo.sp\_Config\_SetValue)
* **Ordered parameter list**: @name type (nullability, default)
* **Result sets**: column names/types/order for each result set
* **Semantics hash**: SHA-256 of the concatenated items above

**6.2 Allowed changes (non-breaking)**

* Add **new optional** parameter **at the end** with a default.
* Add **new output column** **to the end** of a result set **when the API does not bind by ordinal** (our API binds by **name**, so this is allowed).
* Internal logic changes that **don’t** alter signature or semantics.

**6.3 Breaking changes (require vNext)**

* Rename SP or parameter; change param **type/order/nullability**; remove param.
* Remove or rename existing output columns; change type/meaning.
* In these cases, **create a new SP** (e.g., sp\_Config\_SetValue\_v2), deprecate old via ADR + roadmap.

**6.4 Evidence requirement**

On every release, export the **signature table** and attach to the **Evidence Pack**. Any **diff** vs prior release must be **approved** (Security + Systems Architect).

**7. Signature Capture Objects**

**7.1 Metadata table**

CREATE TABLE IF NOT EXISTS dbo.SpSignature (

SpName sysname NOT NULL,

SpSchema sysname NOT NULL DEFAULT 'dbo',

ParamList nvarchar(max) NOT NULL,

ResultShape nvarchar(max) NOT NULL,

SemanticsHash varbinary(32) NOT NULL, -- SHA-256

CapturedAt datetime2 NOT NULL DEFAULT sysutcdatetime(),

CONSTRAINT PK\_SpSignature PRIMARY KEY (SpSchema, SpName, CapturedAt)

);

**7.2 Collector (idempotent)**

CREATE OR ALTER PROCEDURE dbo.sp\_Signature\_Collect

AS

BEGIN

SET NOCOUNT ON;

;WITH ProcList AS (

SELECT p.[object\_id], s.name AS [schema], p.name

FROM sys.procedures p

JOIN sys.schemas s ON s.schema\_id = p.schema\_id

WHERE s.name = 'dbo' AND p.name LIKE 'sp[\_]%'

),

Params AS (

SELECT pl.[schema], pl.name,

STRING\_AGG(CONCAT('@', pr.name, ' ',

t.name,

CASE WHEN pr.max\_length IN (-1,0) OR t.name NOT IN('varchar','nvarchar','varbinary')

THEN ''

ELSE CONCAT('(', CASE WHEN t.name LIKE 'nvar%' THEN pr.max\_length/2 ELSE pr.max\_length END, ')') END,

CASE WHEN pr.is\_output = 1 THEN ' OUTPUT' ELSE '' END,

CASE WHEN pr.has\_default\_value = 1 THEN ' = default' ELSE '' END),

', ') WITHIN GROUP (ORDER BY pr.parameter\_id) AS ParamList

FROM ProcList pl

JOIN sys.parameters pr ON pr.object\_id = pl.object\_id

JOIN sys.types t ON t.user\_type\_id = pr.user\_type\_id

GROUP BY pl.[schema], pl.name

)

INSERT INTO dbo.SpSignature(SpName, SpSchema, ParamList, ResultShape, SemanticsHash)

SELECT

pl.name, pl.[schema],

ISNULL(pa.ParamList, ''),

-- Result sets (best-effort: uses sys.dm\_exec\_describe\_first\_result\_set)

CAST(rs.result\_shape AS nvarchar(max)) AS ResultShape,

HASHBYTES('SHA2\_256',

CONCAT(pl.[schema], '.', pl.name, '|', ISNULL(pa.ParamList,''), '|', CAST(rs.result\_shape AS nvarchar(max)))

) AS SemanticsHash

FROM ProcList pl

OUTER APPLY sys.dm\_exec\_describe\_first\_result\_set

(N'SELECT \* FROM ' + QUOTENAME(pl.[schema]) + N'.' + QUOTENAME(pl.name) + N' ' , NULL, NULL) rs

LEFT JOIN Params pa ON pa.[schema]=pl.[schema] AND pa.name=pl.name;

END

Note: sys.dm\_exec\_describe\_first\_result\_set gives a precise description for most SPs that return a single result set. For multi-set SPs, keep a curated **ResultShape** note in the SP header comment (see §8.3) to augment the collector.

**8. SP Contract Snippets (normative)**

**8.1 Header template (paste atop every SP)**

/\*

Contract: sp\_Config\_SetValue

Params:

@ConfigKey nvarchar(128) NOT NULL

@Value nvarchar(max) NOT NULL

@ValueType nvarchar(32) NOT NULL

@Scope nvarchar(32) NOT NULL

@Description nvarchar(256) NULL

@Tags nvarchar(256) NULL

@Actor nvarchar(128) NOT NULL

@RequestId nvarchar(64) NOT NULL

Result Sets:

RS1: echo row: [key nvarchar(128)], [value nvarchar(max)], [type nvarchar(32)], [scope nvarchar(32)],

[tags nvarchar(256)], [updatedBy nvarchar(128)], [updatedAt datetime2]

Notes:

- Add-only schema rule enforced

- Writes AuditEvent via sp\_Audit\_Write

Breaking changes require a new SP name (…\_v2) and ADR.

\*/

**8.2 Example: read SP (no secrets)**

CREATE OR ALTER PROCEDURE dbo.sp\_Config\_GetValue

@ConfigKey nvarchar(128),

@Scope nvarchar(32) = NULL

AS

BEGIN

SET NOCOUNT ON;

-- NEVER return secrets

SELECT TOP 1

c.Value AS [value],

c.ValueType AS [type]

FROM dbo.AppConfig c

WHERE c.ConfigKey = @ConfigKey

AND c.IsActive = 1

AND (@Scope IS NULL OR c.Scope IN (@Scope, 'Global'))

ORDER BY CASE WHEN c.Scope = @Scope THEN 0 ELSE 1 END;

END

**8.3 Example: mutation SP (with audit)**

CREATE OR ALTER PROCEDURE dbo.sp\_Feature\_Set

@FlagKey nvarchar(128),

@IsEnabled bit,

@Scope nvarchar(32),

@TargetRole nvarchar(64) = NULL,

@Description nvarchar(256) = NULL,

@Actor nvarchar(128),

@RequestId nvarchar(64)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @before nvarchar(max) =

(SELECT \* FROM dbo.FeatureFlag WHERE FlagKey=@FlagKey FOR JSON PATH, WITHOUT\_ARRAY\_WRAPPER);

MERGE dbo.FeatureFlag AS t

USING (SELECT @FlagKey AS FlagKey) AS s

ON (t.FlagKey = s.FlagKey)

WHEN MATCHED THEN UPDATE SET

IsEnabled=@IsEnabled, Scope=@Scope, TargetRole=@TargetRole, Description=@Description,

UpdatedBy=@Actor, UpdatedAt=sysutcdatetime()

WHEN NOT MATCHED THEN INSERT (FlagKey,IsEnabled,Scope,TargetRole,Description,UpdatedBy,UpdatedAt)

VALUES (@FlagKey,@IsEnabled,@Scope,@TargetRole,@Description,@Actor,sysutcdatetime());

DECLARE @after nvarchar(max) =

(SELECT \* FROM dbo.FeatureFlag WHERE FlagKey=@FlagKey FOR JSON PATH, WITHOUT\_ARRAY\_WRAPPER);

EXEC dbo.sp\_Audit\_Write

@Entity='FeatureFlag', @EntityKey=@FlagKey, @Action='Upsert',

@BeforeJson=@before, @AfterJson=@after, @Actor=@Actor, @ActorId=NULL, @RequestId=@RequestId;

-- Echo row for API

SELECT FlagKey AS [flag], IsEnabled, Scope, TargetRole, Description, UpdatedBy, UpdatedAt

FROM dbo.FeatureFlag WHERE FlagKey=@FlagKey;

END

**9. Evidence Pack Hooks (automation)**

Add a deploy-time step (CI) to **collect signatures** and attach to the release:

-- In RTM/Prod staging before deploy:

EXEC dbo.sp\_Signature\_Collect;

-- Export latest snapshot:

-- (Use sqlcmd/bcp to dump this query to CSV/JSON for the Evidence Pack)

SELECT TOP 1 WITH TIES

SpSchema + '.' + SpName AS Proc,

ParamList,

ResultShape,

CONVERT(varchar(64), SemanticsHash, 1) AS HashHex,

CapturedAt

FROM dbo.SpSignature

ORDER BY ROW\_NUMBER() OVER (PARTITION BY SpSchema, SpName ORDER BY CapturedAt DESC);

**Gate rule:** if any **SemanticsHash** changed vs last release **without** an approved ADR and version plan, **block promotion**.

**10. Validation & Monitoring**

* **Permission test:** from the API connection, assert:
  + EXEC dbo.sp\_Config\_GetAll **succeeds**
  + Direct SELECT on tables **fails** (expected)
* **Performance SLOs (DB):** p95 ≤ 75 ms (reads), ≤ 150 ms (mutations incl. audit).
* **Audit coverage:** every mutation SP path writes an AuditEvent row.
* **Signature drift alert (optional):** nightly job compares latest SpSignature hashes to the committed baseline and opens a ticket on drift.

**11. Acceptance Criteria**

1. app\_mcpx has **only EXECUTE** on the allow-list SPs; no table DML.
2. All SPs include a **header contract** (§8.1); collector runs and snapshots signatures.
3. Evidence Pack contains a **current signature snapshot**; release gate blocks on **unauthorized signature changes**.
4. Mutation SPs write **AuditEvent** with BeforeJson/AfterJson, Actor, RequestId.
5. DB perf targets met under CI smoke (k6) with Query Store baselines captured.

**12. Open Issues**

* Decide if sp\_Config\_Diff should live in DB or be computed in API from /config/effective.
* If multiple result sets are required, extend the signature collector to list **all** sets (or annotate in headers).
* Confirm whether CORS allow-list is modeled in Lookup or a dedicated SecurityAllowedOrigins table.

**End of Document — TJ-MCPX-DOC-07a v1.0.0**