**Document: 07 – Data Model & DB Contracts**

**Project:** MCPX-KendoBridge Admin Portal  
**Document ID:** TJ-MCPX-DOC-07  
**Version:** 1.1.0  
**Status:** Draft  
**Date:** 2025-09-27  
**Owner:** Database Architect (Technijian)  
**Confidentiality:** Technijian Internal

**Document Control**

| **Version** | **Date** | **Author** | **Change Summary** | **Status** |
| --- | --- | --- | --- | --- |
| 1.0.0 | 2025-09-26 | Doc-Factory | Initial entities and read SPs | Draft |
| 1.1.0 | 2025-09-27 | Database Architect | Aligned to repo SPs; added mutation SP specs, audit schema, indexes, grants & SLOs | Draft |

**Approvals**

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

**1. Purpose**

Define the **relational data model** and **stored-procedure contracts** for the Admin API. The API MUST access SQL Server **exclusively via stored procedures** (SP-only DAL). Schema evolution follows **add-only** rules (no destructive DDL).

**2. Scope**

* Core configuration tables: **AppConfig**, **FeatureFlag**, **Lookup**, **AuditEvent**.
* Optional job persistence: **Job** (if server needs durable job state).
* Read and mutation stored procedures (present + to-be-added).
* Grants model and performance expectations (NFR alignment).

**Secrets Policy:** Secrets are **not** stored or returned by the DB layer. /config/effective exposes **non-secret** keys only. Secrets live in a secret store (e.g., Key Vault).

**3. Logical Entities**

**3.1 AppConfig**

Holds **non-secret** configuration keys consumed by API & Web.

| **Column** | **Type** | **Notes** |
| --- | --- | --- |
| ConfigKey (PK) | nvarchar(128) | Case-insensitive collation recommended |
| Value | nvarchar(max) | Serialized string; typed at the edge |
| ValueType | nvarchar(32) | `string |
| Scope | nvarchar(32) | e.g., `Alpha |
| Description | nvarchar(256) | Human readable |
| Tags | nvarchar(256) | Comma/JSON list (e.g., ui, network) |
| IsActive | bit default 1 | Soft disable support |
| UpdatedBy | nvarchar(128) | Display name or UPN |
| UpdatedAt | datetime2 default sysutcdatetime() |  |
| RowVer | rowversion | Concurrency token |

**Indexes:** PK on ConfigKey; nonclustered on (Scope, ConfigKey).

**3.2 FeatureFlag**

Controls progressive rollout.

| **Column** | **Type** | **Notes** |
| --- | --- | --- |
| FlagKey (PK) | nvarchar(128) |  |
| IsEnabled | bit |  |
| Scope | nvarchar(32) | Env scoping |
| TargetRole | nvarchar(64) | Optional (e.g., Portal.Admin) |
| Description | nvarchar(256) |  |
| UpdatedBy/At, RowVer | as above |  |

**Indexes:** PK on FlagKey; nonclustered on (Scope, FlagKey).

**3.3 Lookup**

Lightweight reference data.

| **Column** | **Type** | **Notes** |
| --- | --- | --- |
| Domain | nvarchar(64) | e.g., JobType, OriginWhitelist |
| LookupKey | nvarchar(128) |  |
| LookupValue | nvarchar(256) |  |
| IsDeprecated | bit default 0 | Add-only policy → deprecate instead of delete |
| UpdatedBy/At, RowVer |  |  |

**PK:** (Domain, LookupKey)  
**Indexes:** nonclustered on (Domain, IsDeprecated).

**3.4 AuditEvent**

Immutable trail of admin mutations and critical reads.

| **Column** | **Type** | **Notes** |
| --- | --- | --- |
| AuditId (PK) | bigint identity |  |
| Entity | nvarchar(64) | `AppConfig |
| EntityKey | nvarchar(256) | e.g., ConfigKey, FlagKey, Domain:Key |
| Action | nvarchar(32) | `Create |
| BeforeJson | nvarchar(max) | Serialized snapshot (nullable) |
| AfterJson | nvarchar(max) | Serialized snapshot |
| Actor | nvarchar(128) | Display name/UPN |
| ActorId | nvarchar(128) | ObjectId/UPN |
| RequestId | nvarchar(64) | Correlation |
| CreatedAt | datetime2 | default sysutcdatetime() |

**Indexes:** clustered on (CreatedAt desc); nonclustered on (Entity, EntityKey).

**3.5 (Optional) Job**

Persist job metadata if required for history/resume.

| **Column** | **Type** | **Notes** |
| --- | --- | --- |
| JobId (PK) | uniqueidentifier |  |
| JobType | nvarchar(64) |  |
| Status | nvarchar(16) | `queued |
| ParamsJson | nvarchar(max) |  |
| ResultJson | nvarchar(max) |  |
| ErrorCode | nvarchar(64) |  |
| ErrorMessage | nvarchar(512) |  |
| CreatedAt/StartedAt/CompletedAt | datetime2 |  |

**4. Stored Procedure Contracts**

**4.1 Present in repo (Read/Eval)**

| **SP Name** | **Purpose** | **Inputs** | **Output** | **Perf SLO** |
| --- | --- | --- | --- | --- |
| sp\_Config\_GetAll | Return **non-secret** effective config | @Scope nvarchar(32) nullable | Key/Value/Type/Scope/Tags | p95 ≤ 75ms |
| sp\_Config\_GetValue | Return value for key (non-secret) | @ConfigKey nvarchar(128), @Scope nvarchar(32) | Value, Type | p95 ≤ 50ms |
| sp\_Feature\_IsEnabled | Evaluate flag (considering scope/role) | @FlagKey nvarchar(128), @Scope nvarchar(32), @Role nvarchar(64) null | bit | p95 ≤ 25ms |
| sp\_Lookup\_Get | Get lookup value | @Domain nvarchar(64), @LookupKey nvarchar(128) | LookupValue | p95 ≤ 25ms |

**Contract rule:** Read SPs **never** return secrets; if a key is marked secret in policy, the SP MUST omit it or return NULL.

**4.2 Required (to-be-added) Mutation SPs**

All mutations MUST write an **AuditEvent** row with BeforeJson/AfterJson, Actor, and RequestId.

| **SP Name** | **Purpose** | **Inputs (min)** | **Output** | **Notes** |
| --- | --- | --- | --- | --- |
| sp\_Config\_SetValue | Create/Update AppConfig key | @ConfigKey, @Value, @ValueType, @Scope, @Description?, @Tags?, @Actor, @RequestId | rows affected / echo row | Upsert semantics; enforce add-only on columns |
| sp\_Feature\_Set | Enable/Disable/Update Flag | @FlagKey, @IsEnabled, @Scope, @TargetRole?, @Description?, @Actor, @RequestId | rows affected / echo row |  |
| sp\_Lookup\_Upsert | Add/Update Lookup row | @Domain, @LookupKey, @LookupValue, @IsDeprecated bit, @Actor, @RequestId | rows affected / echo row |  |
| sp\_Audit\_Write | Centralized audit writer | @Entity, @EntityKey, @Action, @BeforeJson, @AfterJson, @Actor, @ActorId?, @RequestId | AuditId | Called by other SPs |
| sp\_Security\_Origins\_Set | Manage CORS allow-list (optional) | @Origin, @Action (`Add | Remove`), @Scope, @Actor, @RequestId | rows affected |

**Error handling:** use THROW with an application error number and stable message; API maps to the **error envelope** (code, message, requestId).

**5. API ↔ DB Contracts (Shapes)**

**5.1 Effective Config (GET /config/effective)**

**DB row shape → API DTO**

{

key: string, // ConfigKey

value: string, // Value (stringified)

type: "string|int|bool|json|uri|seconds|enum",

scope: "Alpha|Beta|RTM|Prod|Global",

tags?: string[],

updatedAt: string, // ISO 8601

updatedBy: string

}

**5.2 Feature Evaluation**

* API calls sp\_Feature\_IsEnabled(@FlagKey, @Scope, @Role) → bit.
* API MAY cache evaluations per env/role; cache TTL configurable.

**5.3 Lookup**

* API calls sp\_Lookup\_Get(@Domain, @LookupKey) → LookupValue.

**6. Schema Evolution (Add-Only Rules)**

1. **Never** drop/rename columns in place. Add new columns; deprecate old ones at boundaries.
2. **Never** change column types to incompatible types. Add …\_New then backfill.
3. Create **views** to provide stable outward shapes if refactors occur.
4. Migrations are named VYYYYMMDDHHmm\_\_description.sql and are **idempotent** where possible.

**Example migration skeleton**

-- V202509271300\_\_add\_lookup\_desc.sql

IF COL\_LENGTH('dbo.Lookup', 'Description') IS NULL

ALTER TABLE dbo.Lookup ADD Description nvarchar(256) NULL;

**7. Grants & Security (SP-Only Access)**

* Create a **least-privilege** SQL user for the API, e.g., app\_mcpx.
* DENY SELECT/INSERT/UPDATE/DELETE on tables to app\_mcpx.
* GRANT EXECUTE on **whitelisted** SPs only:
  + sp\_Config\_GetAll, sp\_Config\_GetValue, sp\_Feature\_IsEnabled, sp\_Lookup\_Get
  + sp\_Config\_SetValue, sp\_Feature\_Set, sp\_Lookup\_Upsert, sp\_Audit\_Write (when created)

**Grant script (example)**

CREATE USER app\_mcpx FOR LOGIN app\_mcpx;

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

GRANT EXECUTE ON OBJECT::dbo.sp\_Config\_GetAll TO app\_mcpx;

GRANT EXECUTE ON OBJECT::dbo.sp\_Config\_GetValue TO app\_mcpx;

GRANT EXECUTE ON OBJECT::dbo.sp\_Feature\_IsEnabled TO app\_mcpx;

GRANT EXECUTE ON OBJECT::dbo.sp\_Lookup\_Get TO app\_mcpx;

-- Add mutation SPs when deployed

**8. Performance Expectations (DB NFRs)**

* **Read SPs:** p95 ≤ **75 ms** (Config\_GetAll ≤ 75 ms; flag/lookup ≤ 25–50 ms).
* **Mutation SPs:** p95 ≤ **150 ms** (including audit write).
* **Concurrency:** ≥ **200** concurrent sessions across API; DB remains below 70% DTU/CPU during smoke loads.
* **Indexing:** Review monthly; Query Store tracked; add missing indexes via add-only migrations.
* **SARGability:** Avoid wrapping indexed columns in functions; seekable predicates.

**9. Sample SP Definitions (Normative Sketches)**

These are **reference** bodies to guide implementation; exact SQL can differ while preserving contracts.

**sp\_Config\_GetAll**

CREATE OR ALTER PROCEDURE dbo.sp\_Config\_GetAll

@Scope nvarchar(32) = NULL

AS

BEGIN

SET NOCOUNT ON;

SELECT c.ConfigKey AS [key],

c.Value AS [value],

c.ValueType AS [type],

ISNULL(c.Scope, 'Global') AS [scope],

c.Tags,

c.UpdatedBy,

c.UpdatedAt

FROM dbo.AppConfig c

WHERE c.IsActive = 1

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

END

**sp\_Feature\_IsEnabled**

CREATE OR ALTER PROCEDURE dbo.sp\_Feature\_IsEnabled

@FlagKey nvarchar(128),

@Scope nvarchar(32),

@Role nvarchar(64) = NULL

AS

BEGIN

SET NOCOUNT ON;

DECLARE @enabled bit =

(SELECT TOP 1 IsEnabled

FROM dbo.FeatureFlag

WHERE FlagKey = @FlagKey

AND (Scope = @Scope OR Scope = 'Global')

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

SELECT ISNULL(@enabled, 0) AS IsEnabled;

END

**sp\_Config\_SetValue** *(writes AuditEvent)*

CREATE OR ALTER PROCEDURE dbo.sp\_Config\_SetValue

@ConfigKey nvarchar(128),

@Value nvarchar(max),

@ValueType nvarchar(32),

@Scope nvarchar(32),

@Description nvarchar(256) = NULL,

@Tags nvarchar(256) = NULL,

@Actor nvarchar(128),

@RequestId nvarchar(64)

AS

BEGIN

SET NOCOUNT ON;

DECLARE @before nvarchar(max) = (

SELECT \* FROM dbo.AppConfig WHERE ConfigKey=@ConfigKey FOR JSON PATH, WITHOUT\_ARRAY\_WRAPPER

);

MERGE dbo.AppConfig AS t

USING (SELECT @ConfigKey AS ConfigKey) AS s

ON (t.ConfigKey = s.ConfigKey)

WHEN MATCHED THEN UPDATE SET

Value=@Value, ValueType=@ValueType, Scope=@Scope, Description=@Description, Tags=@Tags,

UpdatedBy=@Actor, UpdatedAt=sysutcdatetime()

WHEN NOT MATCHED THEN INSERT (ConfigKey,Value,ValueType,Scope,Description,Tags,IsActive,UpdatedBy,UpdatedAt)

VALUES (@ConfigKey,@Value,@ValueType,@Scope,@Description,@Tags,1,@Actor,sysutcdatetime());

DECLARE @after nvarchar(max) = (

SELECT \* FROM dbo.AppConfig WHERE ConfigKey=@ConfigKey FOR JSON PATH, WITHOUT\_ARRAY\_WRAPPER

);

EXEC dbo.sp\_Audit\_Write

@Entity='AppConfig', @EntityKey=@ConfigKey, @Action='Upsert',

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

END

**10. Data Quality & Validation**

* **ValueType** validation at SP edge (e.g., bool accepts true|false|1|0).
* **Scope** whitelist enforced (Alpha|Beta|RTM|Prod|Global).
* **Lookup domain/key** regex (simple LIKE constraints).
* **Audit** rows MUST exist for every mutation SP call (unit tests enforce).

**11. Operational Notes**

* **Backups:** full nightly, differential hourly, log every 15 min (see Runbooks).
* **Restore drills:** quarterly; ensure AppConfig/Flags/Lookups and Audit integrity.
* **Rollbacks:** config changes are safe to re-apply via sp\_Config\_SetValue (idempotency by key).
* **Parity:** RTM parity tool compares **intended Prod** JSON (config/expected/expected-prod.json) to **effective** RTM config via sp\_Config\_GetAll.

**12. Acceptance Criteria (DB Layer)**

1. Tables created as defined; **no secrets** stored; **add-only** rule in place.
2. Present SPs (GetAll/GetValue/IsEnabled/Lookup\_Get) return within SLOs; mutation SPs implemented with **AuditEvent** writes.
3. API user (app\_mcpx) has **EXECUTE-only** on whitelisted SPs; **no table DML**.
4. Query Store captures baselines; p95 targets achieved under CI perf smoke.
5. Parity tool can diff RTM effective config vs intended Prod with stable keys.

**13. Open Issues**

* Confirm whether **Job** persistence is required or transient (SSE-only).
* Decide if **Origins allow-list** lives in Lookup or a dedicated table.
* Finalize **ValueType** enumeration and strict validators.

**End of Document — TJ-MCPX-DOC-07 v1.1.0**