**Data Product Reconciliation — Low‑Level Design**

1.Overview

This Low‑Level Design (LLD) document provides the blueprint for implementing **Data Product Reconciliation** on Neo, detailing how curated datasets (**Input‑A**) are validated against independently derived ingestion or source‑of‑truth measures (**Input‑B**) to ensure accuracy, trust, and auditability across environments.

**Detailed Scope:**

* Defines the **technical architecture**, **configuration**, and **operational procedures** for reconciliation.
* Details how **curated datasets (Input‑A)** from the product’s transformed layer are systematically compared against **independently derived ingestion or source‑of‑truth measures (Input‑B)**.
* Establishes methods to **identify and explain discrepancies** between Input‑A and Input‑B.
* Covers the **scope**, **data interfaces**, and **configuration models** used for reconciliation.
* Describes **pipeline workflows** and **Delta Lake schema definitions** that support the process.
* Defines **SLOs** and **tolerances** to measure reconciliation performance and accuracy.
* Outlines **monitoring**, **alerting mechanisms**, and **security/compliance controls**.
* Includes a **testing strategy** to validate functionality and reliability.
* Provides **runbooks** to ensure robust, scalable, and auditable operations across **development, test, and production** environments.

3. **Technical Specification – Input B Configuration & Metadata Standards**

**1. Overview**

This defines the technical structure, configuration patterns, and metadata design required for generating and storing **Input B** in the reconciliation process. Input B serves as the baseline dataset for validating FDP outputs against original source data, and is stored in **Azure Storage Table** in a normalized structure for downstream reconciliation.

This section defines the configuration and metadata framework that governs how **Input B** is generated, structured, and validated. Input B acts as the **reference dataset** for reconciliation, derived independently from ingestion or source-of-truth measures, and stored in a **normalized format** in Azure Storage Table (ReconInputB).

The design supports **multiple reconciliation scenarios**, ensuring flexibility across different Data Product Nodes (FDP, IDP, CDP):

* **Filtration** → selective subset creation
* **Joins** → intersection, anti-join, semi-join to compare across datasets
* **Union** → consolidated measures across multiple sources
* **Complex SQL transformations** → applied to views or curated subsets

The framework follows **data mesh principles**:

* **Config JSON** → Defines file/table-level logic, transformations, and scenario type.
* **Metadata JSON** → Defines column-level validation rules, measure definitions, and tolerances.

**4.2 Input B Storage Structure (Azure Storage Table)**

Each record in ReconInputB represents a **measure instance** tied to a **Data Product (DP)**, **Source Zone**, and **Table**, for a given business date and range.

| **Column Name** | **Description** |
| --- | --- |
| **EffectiveDate** | Business effective date for which the record applies |
| **StartRangeDateTime** | Lower bound of the date/time range covered by the measure |
| **EndRangeDateTime** | Upper bound of the date/time range covered by the measure |
| **ExtractDateTime** | Timestamp when the Input B record was generated |
| **DP\_NAME** | Data Product Name (e.g., FDP\_SALES, IDP\_CUSTOMER) |
| **DP\_SOURCE\_ZONE** | Data Product Source Zone (Hist, Harvested, Published) |
| **DP\_TABLE** | Logical/physical table name within the Data Product |
| **MeasureName** | Name of the measure (e.g., COUNT, SUM\_SALES\_AMOUNT, AVG\_BALANCE) |
| **MeasureValue** | Numeric value of the computed measure |

**4.3 Configuration Model (Config JSON)**

Defines the **data extraction and transformation logic** for Input B generation.  
Sample structure:

**{**

**"data\_product": "FDP\_SALES",**

**"source\_zone": "Harvested",**

**"table": "SALES\_TRANSACTIONS",**

**"scenario": "Filtration",**

**"filters": {**

**"column": "TRANSACTION\_DATE",**

**"condition": ">= '2025-01-01'"**

**},**

**"measures": [**

**{"name": "COUNT", "expression": "COUNT(\*)"},**

**{"name": "SUM\_SALES\_AMOUNT", "expression": "SUM(SALES\_AMOUNT)"}**

**]**

**}**

**4.4 Metadata Model (Metadata JSON)**

Defines **column-level validation rules**, tolerances, and measure configurations.  
Sample structure:

{

"table": "SALES\_TRANSACTIONS",

"validations": [

{

"column": "SALES\_AMOUNT",

"rule": "NOT\_NULL",

"tolerance": "0%"

},

{

"column": "CUSTOMER\_ID",

"rule": "UNIQUENESS",

"tolerance": "0%"

}

],

"measures": [

{

"measure\_name": "SUM\_SALES\_AMOUNT",

"expected\_precision": "2",

"aggregation": "SUM"

}

]

}

**4.5 Linkage with Pipelines**

* **Pipeline-1: Input B Summary Generator**
  + Uses Config JSON to extract, filter, and aggregate source data.
  + Produces normalized measures and stores them in ReconInputB.
* **Pipeline-2: Reconciliation Engine**
  + Reads Input A (FDP/IDP/CDP outputs) and Input B from ReconInputB.
  + Applies Metadata JSON rules for validation.
  + Produces reconciliation results (match/mismatch, variance) and publishes to Delta Lake (ReconResults).

flowchart TD

A[Source Systems (Raw Data)] --> B[Central Ingestion Pipeline - SADP]

B --> C[Generate Input B (Standard Snapshot)]

C --> D[Store Input B in ADLS / Delta Table]

B --> E[FDP Transformation Pipeline]

E --> F[Input A (FDP Curated Data)]

F --> G[Reconciliation Pipeline Trigger]

D --> G

G --> H[Read Config JSON for Data Product]

H --> I[Loop Through Target Tables]

I --> J{Target Table Logic}

J -->|Filter / Join / Union| K[Apply PySpark Transformations]

J -->|Pre-processing SQL| L[Create Temp View and Run SQL]

K --> M[Read Metadata JSON for Target Table]

L --> M

M --> N[Perform Validations (SUM, COUNT\_DISTINCT, EXPRESSION)]

N --> O[Write Results to Recon Output Delta]

O --> P[Alert via Logic App/Email on Failures]

O --> Q[Publish to Power BI Dashboard]

{

"SparkPoolName": "{{ default\_spark }}",

"MetaBasePath": "{{config\_adls\_url }}",

"DPType": "FDP"

"DPName": "GL",

"SrcTableZone": "Historical",

"TableSpecificProperties": [

{

"SrcTablesName": ["03\_ACCOUNT"]

"DPTable": "GL\_ACCOUNT\_SEGMENT",

"SourceTableFilePatternType": "parquet",

"MetadataFileName": "Recon\_Ingest\_Metadata\_SADP\_GL\_ACCOUNT\_SEGMENT.json"

},

{

"SrcTablesName": ["05\_ENTITY"]

"DPTable": "GL\_ENTITY\_SEGMENT",

"SourceTableFilePatternType": "parquet",

"MetadataFileName": "Recon\_Ingest\_Metadata\_SADP\_GL\_ENTITY\_SEGMENT.json"

}

]

}

{

"DPTable": "GL\_ACCOUNT\_SEGMENT",

"MeasureProperties": [

{

"MeasureType": "COUNT\_DISTINCT",

"MeasureName": "COUNT\_DISTINCT\_ACCOUNT\_ID",

"MeasureColumn": "account\_id"

}

]

}

Here is the updated Input B COnfig and for now we go with count and continue with the given metadata by you

{

"SparkPoolName": "{{ default\_spark }}",

"MetaBasePath": "{{config\_adls\_url }}",

"DPType": "FDP"

"DPName": "GL",

"TableSpecificProperties": [

{

"DPTable": "GL\_ACCOUNT\_SEGMENT",

"MetadataFileName": "Recon\_Metadata\_SADP\_GL\_ACCOUNT\_SEGMENT.json"

},

{

"SrcTablesName": ["05\_ENTITY"]

"DPTable": "GL\_ENTITY\_SEGMENT",

"SourceTableFilePatternType": "parquet",

"MetadataFileName": "Recon\_Metadata\_SADP\_GL\_ENTITY\_SEGMENT.json"

}

]

}

{

"DPTable": "GL\_ACCOUNT\_SEGMENT",

"MeasureProperties": [

{

"MeasureType": "COUNT\_DISTINCT",

"MeasureName": "COUNT\_DISTINCT\_ACCOUNT\_ID",

"MeasureColumn": "account\_id"

}

]

}