Fabric - naming conventions

SYSM-353 - Jira project doesn't exist or you don't have permission to view it.

- Naming Conventions for Tables, Columns and Data Structures
- Platform Context: Microsoft Fabric (Lakehouse / Warehouse)

Version	Date	Description	Contributor
V0.1	16 Mar 2026	Initial document	COLOMBANI Théo
V0.2	29 Apr 2026	Revised document: • added new suffixes for columns naming • added section 9 and 10	COLOMBANI Théo
V0.3	07 Apr 2026	Added to the wiki	COLOMBANI Théo
V0.4			

---

1. Purpose and Scope

This document defines the enterprise standards for naming tables, columns, schemas and data structures within the organization's Data Platform implemented on Microsoft Fabric.

The goal of this standard is to ensure:

- Consistent data structures across the platform
- Improved discoverability of datasets
- Simplified onboarding of data engineers and analysts
- Clear lineage between data layers
- Support for BI dimensional models and analytics workloads

This standard focuses specifically on naming conventions for:

- Tables
- Columns
- Column types
- Data layers
- Domain schemas
- Supporting technical tables

---

2. Core Platform Principles

2.1 Medallion Data Architecture

The data platform follows a layered Medallion architecture where data quality increases progressively across three layers:

application platform entreprise

- application contains raw ingested data.
- platform contains cleaned and standardized data.
- entreprise contains business-ready datasets optimized for analytics.

2.2 One Object = One Concept

Each table must represent one logical business concept or process.

Examples:

- customer
- sales_order
- invoice_line

Tables should never represent multiple unrelated concepts, whatever the layer.

2.3 Naming Style

All names must follow the same style:

- lowercase only
- snake_case format
- singular nouns
- explicit descriptive names
- no spaces
- no special characters

Example:

- sales_order
- customer_account
- inventory_movement

---

3. Platform Naming Structure

The general naming structure of the platform is: *<grouping>.<object>*

Examples:

- application : sap.vbak
- platform : sap.sales_order_header
- entreprise : sales.fact_sales_order

---

4. Table Naming Standards

4.1 Application - Tables

application tables store raw source data exactly as ingested.

Naming pattern: *<source_system>.<source_object>*

Examples:

- sap.vbak
- sap.vbap
- salesforce.account
- workday.worker

Rules:

- table names follow the source system object names
- minimal transformation

4.2 Platform - Tables

Platform tables contain cleaned and standardized datasets aligned to source systems.

Naming pattern: *<source_system>.<object_name>*

Examples:

- sap.sales_order_header
- sap.sales_order_item
- salesforce.account

Rules:

- normalized column naming
- deduplicated records
- standardized data types
- still aligned with source system entities

4.3 Entreprise - Tables

Entreprise tables contain enterprise analytics datasets organized by business domain.

Naming pattern: *<domain>.<object>*

Examples:

- sales.fact_sales_order
- sales.dim_customer
- finance.fact_invoice

---

5. Entreprise Table - Types

Dimension Tables

Pattern: *dim_<entity>*

Examples:

- dim_customer
- dim_product
- dim_employee
- dim_date

Fact Tables

Pattern: *fact_<xxxx>*

Examples:

- fact_sales_order
- fact_invoice
- fact_inventory_movement

Bridge Tables

Pattern: *brg_<entity>_<entity>*

Examples:

- brg_customer_segment
- brg_product_category

Reference Tables

Pattern: ref_<reference_object>

Examples:

- ref_currency
- ref_country
- ref_status

Aggregate Tables

Pattern: *agg_<metric>_<grain>*

Examples:

- agg_daily_sales
- agg_monthly_revenue

---

6. Column Naming Standards

Column names must be explicit, consistent, and semantically meaningful.
All columns must follow this structure:

<business_term>_<suffix>

Examples:

- customer_id
- order_amt
- payment_status_cd

6.1 Identity & Keys

Suffixes:

- **_id** Unique business identifier coming from a source system or representing a real-world entity. It is stable and meaningful outside the data platform.
  Example: *customer_id = "C12345"*
- **_key** Surrogate key generated within the data platform for internal joins and modeling purposes (typically in dimensional models).
  Example: *customer_key = 102938*
- **_ref** External reference identifier used to link to external systems, documents, or transactions.
  Example: *order_ref = "EXT-99821"*

Decision Rules: _id vs _key vs _cd

Use _id when:

- The identifier exists in the source system
- It represents a real business entity
- It can be understood by business users

Use _key when:

- The identifier is generated inside the data platform
- It is used for joins between tables
- It supports dimensional modeling (star schema)

Use _cd when:

- The value represents a classification or coded value
- It belongs to a controlled list or reference table

Summary:

- **_id** business identifier
- **_key** technical surrogate key
- **_cd** classification code

6.2 Codes & Classifications

Suffixes:

- **_cd** Short coded value representing a classification (often linked to a reference table).
  Example: *status_cd = "SHIPPED"*
- **_type** High-level classification describing the nature or category of an entity.
  Example: *customer_type = "B2B"*
- **_category** Business grouping used to organize entities into categories.
  Example: *product_category = "ELECTRONICS"*
- **_segment** Business segmentation (e.g., customer segmentation, marketing segmentation).
  Example: *customer_segment = "PREMIUM"*
- **_level_cd** Code representing a hierarchical level or classification tier.
  Example: *risk_level_cd = "HIGH"*

6.3 Names & Descriptions

Suffixes:

- **_name** Human-readable name of an entity or object.
  Example: *customer_nm = "John Smith"*
- **_desc** Detailed textual description providing more context or explanation.
  Example: *status_desc = "Order has been shipped"*
- **_label** Display-friendly label used for reporting or UI purposes.
  Example: *product_label = "Laptop - Premium Range"*

6.4 Dates & Time

Suffixes:

- **_dt** Business date without time component, used for events like orders or deliveries.
  Example: *order_dt = "2025-03-12"*
- **_ts** Timestamp including date and time, typically used for technical or event tracking.
  Example: *created_ts = "2025-03-12 14:25:32"*
- **_time** Time-only value (rare use cases such as time-of-day analysis).
  Example: *event_time = "14:25:32"*

6.5 Financial & Monetary

Suffixes:

- **_amt** Monetary amount representing a financial value.
  Example: *total_amt = 1250.75*
- **_cost_amt** Cost associated with a product, service, or operation.
  Example: *unit_cost_amt = 45.20*
- **_price_amt** Price of a product or service.
  Example: *unit_price_amt = 60.00*
- **_rev_amt** Revenue generated from transactions.
  Example: *net_rev_amt = 1200.00*
- **_tax_amt** Tax amount applied to a transaction.
  Example: *tax_amt = 250.75*
- **_disc_amt** Discount amount applied to a transaction.
  Example: *disc_amt = 50.00*

6.6 Quantities & Measures

Suffixes:

- **_qty** Measured quantity of items or units.
  Example: *order_qty = 3*
- **_cnt** Integer count of occurrences or records.
  Example: *item_cnt = 5*
- **_vol** Volume measurement (e.g., shipment or storage volume).
  Example: *shipment_vol = 1.5*
- **_wt** Weight measurement of an item or shipment.
  Example: *product_wt = 2.3*
- **_len** Length or size measurement.
  Example: *cable_len = 1.2*

6.7 Percentages & Ratios

Suffixes:

- **_pct** Percentage value expressed between 0 and 100.
  Example: *margin_pct = 35.5*
- **_rate** Ratio or rate, often expressed between 0 and 1 or as a fraction.
  Example: *conversion_rate = 0.12*
- **_ratio** Relationship between two values (e.g., debt ratio).
  Example: *debt_ratio = 0.45*
- **_idx** Indexed value used for benchmarking or comparison.
  Example: *price_idx = 105*

6.8 Boolean Fields

Suffixes:

- **is_<condition>** Boolean flag indicating whether a condition is true or false.
  Example: *is_active = true*
- **has_<condition>** Boolean flag indicating the presence of something (less preferred).
  Example: *has_discount = false*

Rule:

- Always prefer is_

6.9 Status & Lifecycle

Suffixes:

- **_status_cd** Code representing the current status of an entity or process.
  Example: *order_status_cd = "DELIVERED"*
- **_status_desc** Description of the status for readability and reporting.
  Example: *order_status_desc = "Order successfully delivered"*

6.10 Audit & Metadata

Columns:

- **created_ts** Timestamp when the record was first created in the system.
  Example: *created_ts = "2025-03-12 10:00:00"*
- **updated_ts** Timestamp of the most recent update applied to the record.
  Example: *updated_ts = "2025-03-12 12:30:00"*

- **load_ts** Timestamp when the data was loaded into the current layer.
  Example: *load_ts = "2025-03-12 13:00:00"*
- **pipeline_run_id** Identifier of the pipeline execution that produced the data.
  Example: *pipeline_run_id = "run_20250312_01"*
- **record_source** Source system or process that generated the record.
  Example: *record_source = "sap"*
- **row_hash** Hash value used to detect changes in the record.
  Example: *row_hash = "a94f8c1d2b"*

6.11 Technical Fields

Suffixes:

- **_hash** Hash value used for change detection or deduplication.
  Example: *row_hash = "a94f8c1d2b"*
- **_version** Version number of a model, record, or logic.
  Example: *model_version = "v1.2"*
- **_seq** Sequence number representing ordering or position.
  Example: *line_seq = 1*

6.12 Event / Log Data

Suffixes:

- **_event** Name or type of an event.
  Example: *event_type = "LOGIN"*
- **_ts** Timestamp when the event occurred.
  Example: *event_ts = "2025-03-12 09:15:00"*
- **_id** Unique identifier of the event.
  Example: *event_id = "EVT123456"*

6.13 Geography

Suffixes:

- **_country_cd** Country code based on a standard reference (e.g., ISO).
  Example: *country_cd = "FR"*
- **_region_cd** Region or state code.
  Example: *region_cd = "IDF"*
- **_city** City name.
  Example: *city = "Paris"*
- **_postal_cd** Postal or ZIP code.
  Example: *postal_cd = "75001"*

---

7. Column Type Standards

Suffix	Expected Type
_id	string or integer
_key	integer
_ref	string
_cd	string
_type	string
_category	string
_segment	string
_level_cd	string
_name	string
_desc	string
_label	string

_dt	date
_ts	timestamp
_time	time
_amt	decimal
_cost_amt	decimal
_price_amt	decimal
_rev_amt	decimal
_tax_amt	decimal
_disc_amt	decimal
_qty	decimal or integer
_cnt	integer
_vol	decimal
_wt	decimal
_len	decimal
_pct	decimal
_rate	decimal
_ratio	decimal
_idx	decimal or integer
is_	boolean
has_	boolean
_status_cd	string
_status_desc	string
_hash	string
_version	string
_seq	integer
_event	string
_country_cd	string
_region_cd	string
_city	string
_postal_cd	string
created_ts	timestamp
updated_ts	timestamp
load_ts	timestamp
pipeline_run_id	string
record_source	string
row_hash	string

---

8. Mandatory Metadata Columns

Application Layer

The application layer captures raw ingestion information to ensure traceability and replay capability.

Column Name	Type	Description	Example
_ingestion_ts	timestamp	Timestamp when the record was ingested into the platform. Used to track ingestion time and support replay or debugging of ingestion pipelines.	2025-03-12 14:25:32
_batch_id	string	Identifier of the ingestion batch or pipeline execution that loaded the record. Useful for pipeline monitoring and troubleshooting.	batch_20250312_01
_source_system	string	Name of the source system from which the record originated. Helps identify data lineage and upstream systems.	sap
_source_file (or source object)	string	Name or path of the file used to ingest the record (when file-based ingestion is used). Enables traceability back to the original data file.	Sap_sales_20250312.csv
_record_hash	string	Hash value representing the content of the record. Used to detect changes between loads or identify duplicates efficiently.	a9f4b12c89d2

Platform Layer

Column Name	Type	Description	Example
created_ts	timestamp	Timestamp when the record was first created in the platform layer. Helps identify when the data became available in the curated layer.	2025-03-12 14:25:32
updated_ts	timestamp	Timestamp of the latest update applied to the record in the platform layer. Useful for incremental processing and debugging transformations.	2025-03-12 16:02:01
row_hash	string	Hash calculated from the business columns of the row. Used to detect if the data has changed between loads.	8fa21a3b5d
record_source	string	Source system or pipeline that generated the record in the platform layer. Used for lineage and audit.	sap_sales_pipeline
is_deleted	boolean	Logical deletion flag used when the source record is removed or marked as inactive. Enables soft deletion instead of physical removal.	false

Entreprise Layer

Column Name	Type	Description	Example
created_ts	timestamp	Timestamp when the record was first created in the entreprise dataset. Useful for lineage and auditing.	2025-03-12 17:30:10
updated_ts	timestamp	Timestamp of the latest modification applied to the record. Helps identify refresh cycles of the dataset.	2025-03-13 02:00:00

Others Technical standards

Column Name	Type	Description	Example
load_ts	timestamp	Timestamp when the record was loaded into the table.	2025-03-13 02:00:00
pipeline_run_id	string	Identifier of the pipeline or job execution that produced the object.	pipeline_run_84721

---

9. Error and Data Quality Tables

Error and data quality tables are used to **manage invalid, incomplete, or inconsistent data** during ingestion and transformation processes.

They are critical to:

- ensure data quality without blocking pipelines
- isolate problematic records, enable debugging and monitoring
- support data governance and audit

Naming Patterns

- reject_<object>
- quarantine_<object>
- error_<object>

Examples:

- reject_sales_order
- quarantine_customer
- Error_invoice

Key Differences

- reject invalid data (cannot be used)
- quarantine questionable data (needs review)
- error technical processing issue

When to Use Each Type

reject_<object>

Use this table when:

- records fail **hard validation rules**
- data is clearly invalid and cannot be processed
- records must be excluded from downstream layers

Typical cases:

- missing mandatory fields (e.g., customer_id is null)
- invalid formats (e.g., wrong date format)
- broken schema

Example:

- reject_sales_order contains orders where order_id is missing

quarantine_<object>

Use this table when:

- data is **potentially usable but requires review**
- validation rules are not strict blockers
- business validation fails but data may still be recoverable

Typical cases:

- unknown reference values (e.g., unknown country_cd)
- inconsistent business logic
- partial data issues

Example:

- quarantine_customer contains customers with missing segmentation

error_<object>

Use this table when:

- errors occur during **processing or transformation logic**
- the issue is technical rather than data quality related

Typical cases:

- transformation failures
- join issues
- pipeline execution errors

Example:

- error_invoice contains records that failed during currency conversion

Columns

Typical structure:

- object_id Identifier of the record related to the error (usually a business identifier).
  Example: *object_id = "ORD12345"*

- source_object Name of the data platform object (table or dataset) where the error was detected. This refers to the layer, domain, and table within the platform.
  Example: *source_object = "platform.sap.sales_order_header"*
- column_name Name of the column that caused the error (when applicable). Helps pinpoint the issue precisely.
  Example: *column_name = "currency_cd"*
- invalid_value Value that caused the error. Useful for debugging and data quality analysis.
  Example: *invalid_value = "EUROPE"*
- error_code Standardized error code used to classify the type of issue. Enables monitoring and grouping of errors.
  Example: *error_code = "CURR_001"*
- error_message Human-readable description of the error explaining what went wrong.
  Example: *error_message = "Invalid currency code"*
- error_type Category of error indicating whether it is a data issue, validation issue, or technical issue.
  Example: *error_type = "DATA_VALIDATION"*
- pipeline_run_id Identifier of the pipeline execution that generated the error. Used for traceability and debugging.
  Example: *pipeline_run_id = "run_20250312_01"*
- load_ts Timestamp when the error record was written to the error table.
  Example: *load_ts = "2025-03-12 14:32:10"*

---

10. Mapping Tables

Mapping tables are used to **translate, standardize, or align data between source-oriented datasets and enterprise business models**.

Especially for an architectures where:

- lower layers are source-system oriented
- upper layers are business-domain oriented

Mapping tables are used to:

- align source system values with enterprise standards
- resolve inconsistencies between systems
- standardize business definitions
- enable cross-system integration

Naming Pattern

map_<source_object>_<business_object>

Example:

- map_sap_customer_enterprise_customer

When to Use Mapping Tables

Use mapping tables when:

- multiple systems use different identifiers or codes
- business definitions differ across systems
- values need to be harmonized for analytics

Common Use Cases

Code Standardization

Different systems use different formats for the same concept.

Example:

- source_value = "FR"
- target_value = "FRA"

Entity Alignment

Different systems represent the same entity differently.

Example:

- SAP customer_id differs from CRM customer_id

Category Harmonization

Source categories do not match enterprise categories.

Example:

- source_value = "ELEC"
- target_value = "ELECTRONICS"

Hierarchy Standardization

Different hierarchy structures across systems.

Example:

- local region vs global region

Typical Structure

Mapping tables usually contain:

- source_system Origin of the data
  - Example: *source_system = "sap"*
- source_value Value from the source system
  - Example: *source_value = "ELEC"*
- target_value Standardized enterprise value
  - Example: *target_value = "ELECTRONICS"*
- mapping_type Type of mapping applied
  - Example: *mapping_type = "CATEGORY_MAPPING"*
- valid_from_dt Start date of the mapping validity
  - Example: *valid_from_dt = "2024-01-01"*
- valid_to_dt End date of the mapping validity
  - Example: *valid_to_dt = "9999-12-31"*