Metadata-Driven Framework – Microsoft Fabric (Detailed, Simplified)

Inspired by Paul Andrew’s metadata-driven ETL framework (ADF) — implemented in Microsoft Fabric.  
  
This document consolidates the supplied ETL Framework reference and the Fabric implementation. It describes the metadata design, pipeline hierarchy (Grandparent → Parent → Child), load behavior (full & incremental), lakehouse ingestion folder structure, operational patterns, and governance.  
  
Note: This version explains the metadata tables conceptually without including DDL scripts.

# Revision History

# Table of Contents

* 1. Introduction and Objectives

# 2. High-Level Architecture

The high-level architecture of the metadata framework is designed to clearly separate configuration (metadata) from execution (pipelines). This separation ensures that orchestration is dynamic, reusable, and scalable across multiple data sources and applications.  
  
Key principles:  
- Separation of Concerns: Metadata tables define what should run, while pipelines define how execution happens.  
- Hierarchy of Control: The orchestration hierarchy is managed through grandparent, parent, and child pipelines, corresponding to application → stage → pipeline levels.  
- Late Binding: Pipelines only receive parameters and queries at runtime from metadata. This avoids hardcoding and allows flexibility for future changes without redeployment.  
- Lakehouse Integration: All ingestion and transformation pipelines land data into a Lakehouse with a standardized folder and file naming convention. This supports efficient partition pruning and downstream analytics.  
- Observability & Auditability: Execution tables provide full lineage and status tracking across all levels, from orchestration runs down to individual file loads.  
  
The architecture is modular, which means new pipelines or sources can be added simply by inserting metadata rows, minimizing development effort and ensuring long-term maintainability.

* 3. Metadata Repository Design (Conceptual)
* 4. Pipeline Hierarchy & Orchestration Flow
* 5. Full vs Incremental Load Patterns & Watermarking
* 6. Lakehouse Ingestion Folder Structure & File Naming
* 7. Execution Logging, Audit & Retention
* 8. Error Handling, Retries & Alerts
* 9. Operational Runbook & Day-to-Day Procedures
* 10. Security, Governance & Change Control
* 11. Appendix: Examples of Metadata Rows & Pseudocode

# 1. Introduction and Objectives

This framework provides an end-to-end metadata-driven orchestration model in Microsoft Fabric. It uses centralized metadata tables to dynamically drive pipeline execution, routing, parameterization, and logging. It is inspired by Paul Andrew’s ADF metadata framework but adapted to Fabric with Lakehouse storage.

* Standardize ETL/ELT processes across all sources and stages.
* Enable both full and incremental loads to the Lakehouse.
* Provide observability, auditing, and error handling across runs.
* Reduce technical debt by removing hardcoded logic and using metadata.

# 2. High-Level Architecture

The high-level architecture of the metadata framework is designed to clearly separate configuration (metadata) from execution (pipelines). This separation ensures that orchestration is dynamic, reusable, and scalable across multiple data sources and applications.  
  
Key principles:  
- Separation of Concerns: Metadata tables define what should run, while pipelines define how execution happens.  
- Hierarchy of Control: The orchestration hierarchy is managed through grandparent, parent, and child pipelines, corresponding to application → stage → pipeline levels.  
- Late Binding: Pipelines only receive parameters and queries at runtime from metadata. This avoids hardcoding and allows flexibility for future changes without redeployment.  
- Lakehouse Integration: All ingestion and transformation pipelines land data into a Lakehouse with a standardized folder and file naming convention. This supports efficient partition pruning and downstream analytics.  
- Observability & Auditability: Execution tables provide full lineage and status tracking across all levels, from orchestration runs down to individual file loads.  
  
The architecture is modular, which means new pipelines or sources can be added simply by inserting metadata rows, minimizing development effort and ensuring long-term maintainability.

* Metadata Repository: Applications, Sources, Stages, Pipelines, Parameters, Queries.
* Grandparent Pipeline: application-level orchestration.
* Parent Pipeline: stage-level orchestration.
* Child Pipeline: executes worker pipelines with runtime parameters.
* Worker Pipelines: ingestion/transform jobs writing to Lakehouse.
* Lakehouse: Parquet storage with deterministic folder and file patterns.
* Execution & Audit Tables: track pipeline runs, watermarks, statuses, and errors.

# 3. Metadata Repository Design (Conceptual)

* tbl\_applications: Top-level projects/applications grouping stages and pipelines.
* tbl\_sources: Defines data sources with numeric order used for folder prefixes.
* tbl\_stages: Logical processing stages: ingest, transform, publish.
* tbl\_sets: Groups of pipelines for logical or operational grouping.
* tbl\_pipeline: Registers pipelines, default load type (full/incremental), enabled flag.
* tbl\_parameters: Pipeline parameters such as table names, watermark columns, paths.
* tbl\_pipeline\_queries: SQL snippets or queries linked to pipelines (e.g., incremental).
* tbl\_execution\_grandparent: Logs each top-level orchestration run.
* tbl\_execution\_parent: Logs each stage run within a grandparent execution.
* tbl\_execution\_child: Logs each pipeline run with row counts and error details.
* tbl\_execution\_current\_etl: Tracks current pipeline state including watermarks.
* tbl\_execution\_history\_etl: Audit trail of past runs for pipelines.

# 4. Pipeline Hierarchy & Orchestration Flow

1. Grandparent Pipeline starts, logs entry in execution\_grandparent.
2. Grandparent retrieves active stages and calls Parent Pipeline for each.
3. Parent logs entry in execution\_parent, retrieves pipelines, and calls Child.
4. Child reads parameters, executes worker, writes Parquet files, logs execution\_child.
5. Child updates execution\_current\_etl and appends execution\_history\_etl.
6. Parent completes and updates execution\_parent.
7. Grandparent completes and updates execution\_grandparent.

# 5. Full vs Incremental Load Patterns & Watermarking

The framework supports both load types, controlled by metadata or parameters.

* Full Load: extracts entire source, writes complete snapshot.
* Incremental Load: uses watermark from execution\_current\_etl, extracts new/changed rows, updates watermark after success.

# 6. Lakehouse Ingestion Folder Structure & File Naming

File path pattern:

* xx\_sourcename/Year\_yyyy/Month\_mm/Day\_dd/Table\_name\_yyyy\_mm\_dd\_hh\_mm\_ss.parquet
* xx = source order (zero-padded), sourcename = source\_name.
* Year/Month/Day = partition folders.
* Table name + timestamp = unique file name.
* Example: 03\_Customers/Year\_2025/Month\_09/Day\_18/Customers\_2025\_09\_18\_12\_30\_00.parquet

# 7. Execution Logging, Audit & Retention

Execution tables log every run with start/end times, statuses, row counts, and errors. Current ETL holds last state; history provides a complete audit trail.

# 8. Error Handling, Retries & Alerts

Errors logged at child pipeline level, retries based on metadata. Alerts sent via Fabric connectors (email/Teams/ServiceNow).

# 9. Operational Runbook & Day-to-Day Procedures

* Onboard new pipeline: add metadata entries in applications, sources, stages, pipelines, parameters, queries.
* Deactivate pipeline: set enabled flag to 0.
* Switch load type: update metadata field.
* Investigate failures: review execution\_child error\_log and related records.
* Trace run: use run\_batch\_id across execution tables.

# 10. Security, Governance & Change Control

* Restrict metadata updates via RBAC.
* Use managed identities/service principals for access.
* Audit all metadata changes.
* Enforce least privilege on Lakehouse access.

# 11. Appendix: Examples of Metadata Rows & Pseudocode

Example tbl\_pipeline row: Ingest\_Customers | Load Type: Incremental | Enabled: Y

Example tbl\_parameters row: param\_name=watermark\_column | param\_value=modified\_date

File path pseudocode:

xx = source\_order.zfill(2)  
folder = f"{xx}\_{source\_name}/Year\_{yyyy}/Month\_{mm}/Day\_{dd}"  
filename = f"{table}\_{yyyy}\_{mm}\_{dd}\_{HH}\_{MM}\_{SS}.parquet"  
path = folder + '/' + filename

End of document.