Data & Analytics (DNA) Platform Design

Version: 1.0

# Summary

We need a framework to serve as a platform for all data ingestion activities at Advantage.

# Requirements

[DNA Platform - Ingestion Framework Requirements](https://advantagesolutionsnet.sharepoint.com/:w:/r/sites/DataLakePlatform/Shared%20Documents/Data%20Engineering/DNA%20Platform%20-%20Ingestion%20Framework%20Requirements.docx?d=wa66b87bde3d84acf81fcb476907a2000&csf=1&web=1&e=ZdjZ6Z)

# Scope & Context

## In-Scope/Current/Milestone 2

* Centralized Operational Configuration
* Centralized Project Level Configuration
* Operational Parameter in Workflows & Pipelines
* Intelligent (Auto) Bronze Level Ingestion
* CDC
  + Watermark
  + Primary key
  + Hash
* Project level code & config review process
* Operational level config governance
* Framework code as a versioned package or library
* Databricks Features:
  + DLT
  + Workflow
  + Asset Bundle
* Notebook Templates
* Alerting & Notifications
  + Failures
  + Latency (with a configurable threshold)
* System Health & Monitoring (at a basic level)
* As-is ingestion into Raw layer
* Bronze level transformations exist based on Milestone 1 project deliverables
  + Column rename
  + Static column expressions
  + CDC processing
  + Column cleanup
  + Deduplication on provided keys (may or may not be primary)
* Basic Data Quality Rules for Bronze & Silver

## Out of Scope/Future State

* Consolidated UI (beyond what Databricks provides) is not in scope, but the design should not preclude
* System Health & Monitoring Dashboarding/UI
  + UI/Dashboards. NOTE: The design shall not preclude this, it can however leave details to future designs
  + Full lineage views tying pipelines, status, config, etc... together in one place. NOTE: The design shall not preclude these, it can however leave details to future designs
* CDC:
  + Source log based
* Custom error handling & logging
* Databricks Features:
  + LakeFlow
* Direct integration with semantic layer or other reporting platform features
* Silver, Gold or other standardized transformations
* Advanced Data Quality Rules

# Key Decisions

* What format would be best for enterprise level controls?
* What is the configuration/metadata model standard?
  + For project level config, as defined in M3

For enterprise level, a new model will follow:

Each project will insert records into this model on deployment (if not exists) to ensure it can be controlled at the organizational level

Controls will be to:

* Enable/disable the entire project
* Enable/disable the workflows
* Enable/disable at a layer level (i.e., bronze, silver, gold, or others)

Framework Code Repository

Will be in the “” GitHub repository

GitHub Actions will run available unit tests

On successful tests, GitHub Actions will package the files into a versioned artifact

Must be accessible to current & future projects requiring data ingestion & processing

### Repository Considerations

Databricks Asset Bundles provide features to support a single repository & multiple deployment model. [See Asset Bundles #paths](https://docs.databricks.com/en/dev-tools/bundles/settings.html#paths)

We considered this approach but ultimately decided against it due to experiences in previous projects. Having multiple teams working in different parts of a single repository caused issues with branch merging, ownership concerns on various parts, mistakes can cascade across multiple teams & require a technical lead or architect resource for oversight.

# Operational Configuration – Enterprise

Controls Workflow, Pipeline or Job run states (enabled/disabled)

Controls Layer level run states (enabled/disabled)

# Project/Datasource Configuration

Each project will manage its configuration in a centralized store using a template notebook provided by the framework

This notebook will accept a list of JSON files & write to a unique project named folder in the central volume

The notebook will be declarative, so if a table’s JSON file is missing in source control, it will be removed from the central store. Changes will always be overwrites

TableSet Configuration File Features:

Can define 1 or many tables in the configuration file

Configuration Items:

* Layer transformations can be enabled or disabled
* Mappings for column renaming (to conform to standards and project needs)
* Optional Primary Key definitions
* [Raw-Bronze Only] Datasource PySpark configuration
* Data Quality Rules
* Source & Target DBSchema dictionaries when Unity Catalog is not available
* CDC Options for
  + Watermark column to be used
  + Primary key
  + Hash

## File vs Table Considerations

We

Workflow/Pipeline/Job Configuration

Utilizes Databricks standard YAML features. Managed by Asset Bundles including default parameters, environment overrides & more.

The config file would have a standard model, as defined by a Python Class (configuration.py)

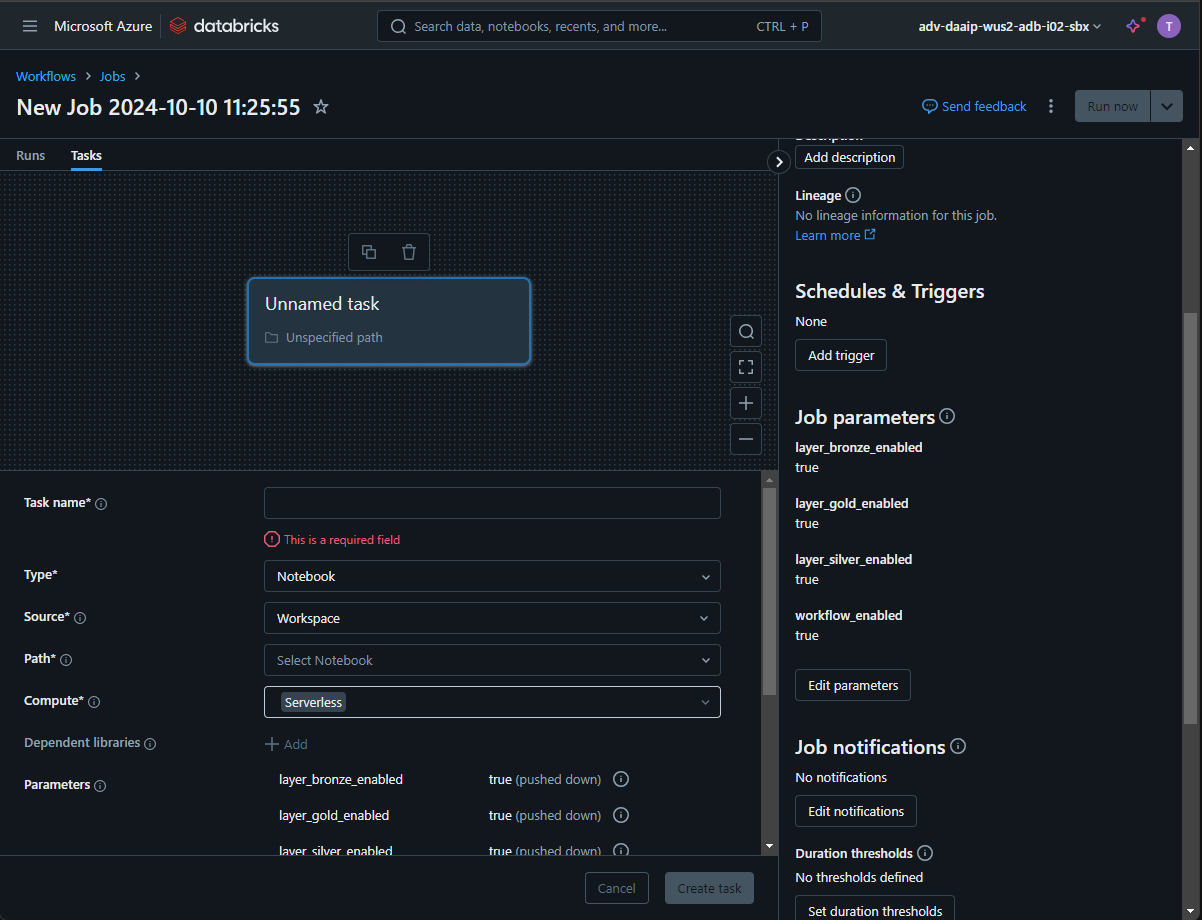
Configuration files validation step at design time. As a validation step before PR/merge/Deploy

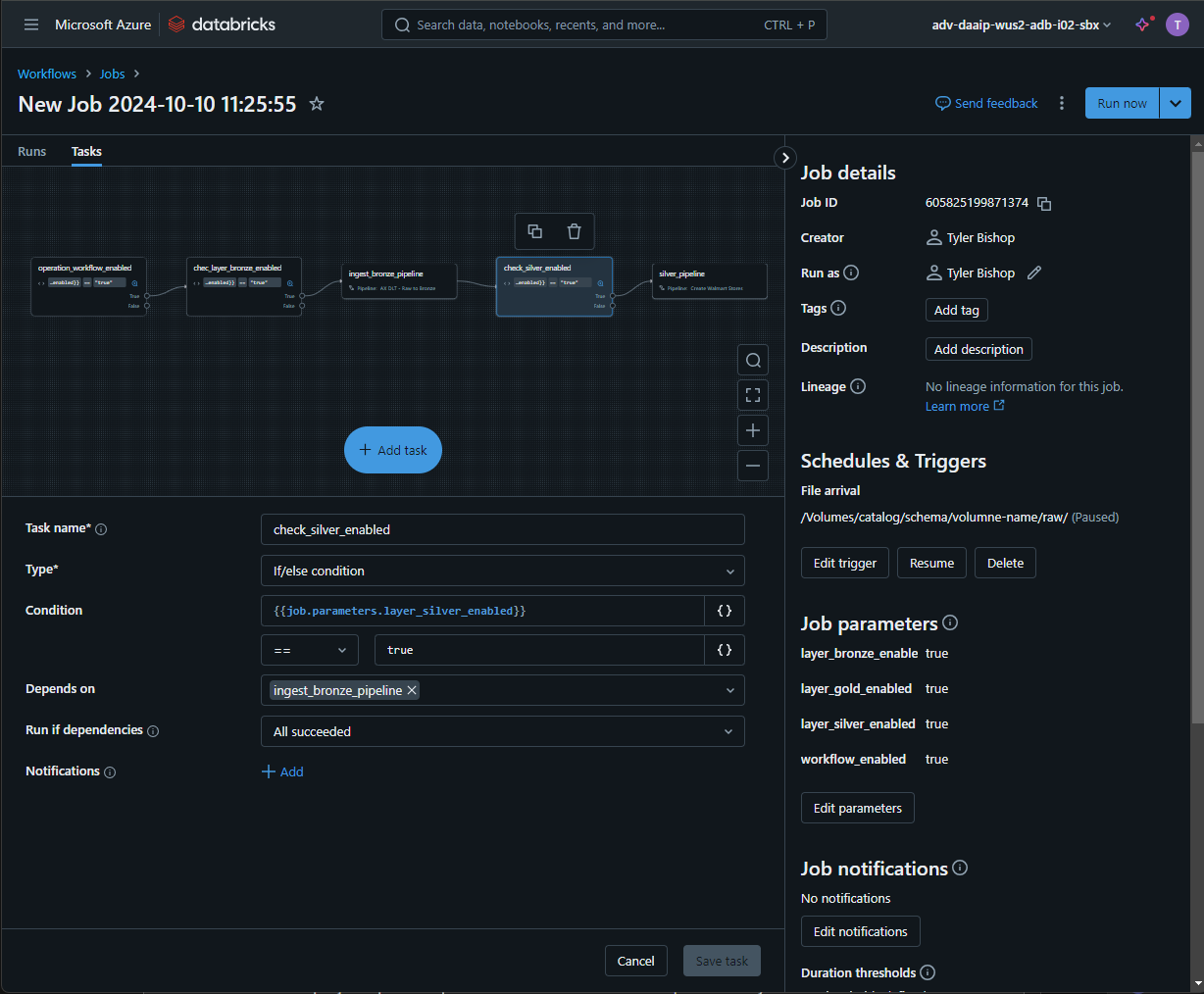
Configuration file generation utilities

# Workflow Controls

Each workflow will accept an enable/disabled parameter & need to account for that parameter at runtime

Each workflow or pipeline will accept a list of parameters for its “layers” (i.e., bronze, silver, gold, etc...) & account for those parameters at runtime



Example workflow with checks for operational parameters before proceeding to processing the project’s defined layers.

# Default Bronze Ingestion Process

Given a list of tables/configurations, reads the table source according to configurations defined above into the target

Features:

Reads from source as defined in the table configuration JSON (pyspark config, source type, source location, etc...)

Enforces source & target schemas, if provided

Performs column rename as defined in mapping dictionary, if provided

Enforce Data quality rules enforced, if provided. Must use Databricks’ validation so it is properly recorded in the Databricks system

Only supported when run within a DLT pipeline

# Custom Bronze Ingestion Process

If required for the project, a custom process can be defined via 2 methods

A clone of the default ingestion notebook can be made, with customizations applied in code. Assumes it still runs as a DLT, just with one-off code changes applied

A notebook is provided & mapped in the project’s workflow at the appropriate level. Can be run as a job, task, DLT pipeline or others as needed

# CDC Strategies

## Source Log Based

Out of scope for now

Is the preferred, if & when it’s available, as it utilizes database implementation for change detection

## Watermark Columns

Detects changes to records based on a provided set of columns

Utilizes an optionally provided query (with a ‘SELECT \*’ as a default) as well as a given timestamp for the watermark columns to pull the latest changes in

Rather than explicitly store these values in a table, the process will involve reading the target table for those columns to determine the values.

Optionally, it can take override values if a manual backfill process is desired

# Silver & Gold Processes

Largely out of scope, but custom notebooks of Python, SQL or hybrid are allowed & should be available to be ran in the project’s workflow

# Alerting & Notifications

Utilizes Databricks’ out of the box feature set: <https://docs.databricks.com/en/jobs/notifications.html>

Workflows shall have operational hooks, if available as well as project specific requirements (a project team’s shared email, Teams channels, etc...)

A default notebook shall be provided by the framework to facilitate status cards if Teams Channel integration is needed

# Governance

Workflow, pipelines and jobs shall have default permissioning to allow operations teams to manage, as well as including project specific users or groups.

Handled by the Asset Bundle definition with options to override in the Databricks environment

# System Health Monitoring

Shared Queries shall be provided that:

* Query the centralized table configurations on all standard model fields
* Query the Jobs table for current status, parameters, runtime & other fields as needed to determine health
* Query the jobs audit table to understand historical runs status
* Queries for current failures in the system

# Enterprise Orchestration

Utilizes Databricks out of the box integration & API to facilitate external orchestration

Databricks Workflows offer needed orchestration feature set

No additional work is in scope, other than mentioned control/operational parameters in the workflows, pipelines & jobs

# Secret Management

[Databricks Secret Scopes - Process and Naming Convention.docx](https://advantagesolutionsnet.sharepoint.com/:w:/r/sites/DataLakePlatform/Shared%20Documents/Data%20and%20Analytics%20Platform%20-%20Shared%20Documentation/Design%20Documents/General%20%26%20Non-Stage%20Specific/Security%20Artifacts/Databricks%20Secret%20Scopes%20-%20Process%20and%20Naming%20Convention.docx?d=w7f3e484d45154710a33da003e094c51b&csf=1&web=1&e=dIA7so)

# Scalability

Infinitely scalable by allowing datasource/use-case/project level cluster (dynamic or static) as well as leveraging Compute Policies, Pools & Shared Compute. This is a native Databricks feature via configuration & optionally handled in Asset Bundles

See cluster values at each level in the Asset Bundle documentation:  
<https://learn.microsoft.com/en-us/azure/databricks/dev-tools/bundles/settings>

<https://learn.microsoft.com/en-us/azure/databricks/compute/>

# Appendix

## Table Configuration Spec

