**Objective:**

The objective of this document is to define the implementation of auditing process within the Generic Workflow Framework. This audit process will capture and track job execution details at table level. The implementation ensures data integrity, transparency, and enables real-time monitoring and performance tracking across different workflow layers.

**Audit Table Design:**

The **Generic Workflow Framework** maintains an audit table/view to ensure comprehensive tracking and monitoring:

1. **PAC\_Audit\_View (Table-Level Audit Table/View)**

* Maintains detailed execution logs at the table level(landing to bronze), tracking individual task & child task runs within a workflow.
* Stores information on data ingestion, and processing status for each table.

**e.g.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Job\_id** | **Job\_Run\_id** | **Layer** | **Task\_Run\_id** | **Table\_Name** | **Start\_Date\_Time** | **End\_Date\_Time** | **Status** | **Error** |
| a | 1 | Raw | 1 | Geo\_Meta | 2024-12-20 | 2024-12-20 | Succeeded | null |
| a | 1 | Bronze | null | Geo\_Meta | 2024-12-20 | 2024-12-20 | Succeeded | null |

**Audit Implementation Approach 1:**

**A diagram of a process

AI-generated content may be incorrect.**

* Since the Raw layer’s table-level execution details can be retrieved using the Databricks SDK, these logs can be used to create the **Raw\_Audit\_View.** To ensure the view reflects the latest logs, it needs to be refreshed after every workflow run.
* The Bronze layer’s table-level execution details can be obtained from published event logs using inputs (DLT\_Pipeline\_id and DLT\_Run\_id) from the **Raw\_Audit\_View**.
* Finally, by combining the table-level execution logs from both layers, a consolidated view **(PAC\_Audit\_View)** will be created.
* To accomplish this, introduce a new notebook task in the existing workflow, placed after the "Bronze Notebook." This task will be responsible for creating and refreshing both the **Raw\_Audit\_View and PAC\_Audit\_View** with the latest run details.
* In the existing bronze trigger notebook, we need to add a notebook exit statement to pass:
  1. DLT\_Pipeline\_id (Required to fetch bronze audit logs in the final audit view)
  2. DLT\_ Run\_id (Required to fetch bronze audit logs in the final audit view)
  3. New DLT Pipeline Flag (only if a new DLT pipeline is created in the current run)
* We will use the Databricks SDK Workspace Client to capture the DLT Pipeline ID, DLT Pipeline Latest Run ID, and New DLT Pipeline Flag. These details will be logged in the **Raw\_Audit\_View** from "Audit\_view\_refresh" task.
* The **PAC\_Audit\_View** will be refreshed only if a new DLT pipeline is created in the current run. For an existing DLT pipeline, refreshing the **PAC\_Audit\_View** is not necessary, as the view always reflects the latest data.
* Example, if a single table (e.g., geo\_meta) is ingested from the landing zone to the bronze layer, the audit views will reflect:
  1. **Raw\_Audit\_View**: 1 record (for the raw layer ingestion).
  2. **PAC\_Audit\_View**: 2 records (1 for the raw layer and 1 for the bronze layer ingestion).

A screenshot of a computer

AI-generated content may be incorrect.

**Audit Implementation Approach 2:**

Generic Workflow:

**A diagram of a diagram

AI-generated content may be incorrect.**

* Since the Raw layer’s table-level execution details can be retrieved using the Databricks SDK, these logs can be used to create the **Raw\_Audit\_View.** To ensure the view reflects the latest logs, it needs to be refreshed after every workflow run.
* The Bronze layer’s table-level execution details can be obtained from published event logs using inputs (DLT\_Pipeline\_id and DLT\_Run\_id) from the current run.
* Based on the above generic workflow, the following explains the Bronze notebook task and its role in implementing the audit mechanism for this approach.

A diagram of a diagram

AI-generated content may be incorrect.

* Finally, by combining the table-level execution logs from both layers, a consolidated view **(PAC\_Audit\_View)** will be created.
* Example, if a single table (e.g., geo\_meta) is ingested from the landing zone to the bronze layer, the audit views will reflect:

1. **Raw\_Audit\_View**: 1 record (for the raw layer ingestion).
2. **Bronze\_Audit\_table**: 1 records (for the bronze layer ingestion).
3. **PAC\_Audit\_View**: 2 records (1 for the raw layer and 1 for the bronze layer ingestion).

A screenshot of a computer program

AI-generated content may be incorrect.

**Limitations of Approach 2:**

* The Bronze notebook task will remain in a running state until the associated DLT pipeline completes execution. This leads to additional costs as one cluster remains active for monitoring the DLT pipeline run, while another cluster runs the actual DLT pipeline execution.