Deloitte logo1

**ABC & Ingestion Framework**

**Design Document**

**Document Change History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date Created / Modified** | **Created / Modified By** | **Brief Description of Change** | **Doc version** |
| 05/06/2020 | Ritika Bhatia | Initial Version | 1.0 |
| 05/12/2020 | Kanishk Malik | Reviewed |  |
|  |  |  |  |

**Contents**

[1. Introduction 3](#_Toc43320867)

[2. Technology Stack 4](#_Toc43320868)

[3. Data Model 4](#_Toc43320869)

[4. Data Dictionary 4](#_Toc43320870)

[5. Framework Data Flow 4](#_Toc43320871)

[5.1 Master Ingestion pipeline: pl\_source\_inbound\_generic\_ingestor\_abc 5](#_Toc43320872)

[5.2 Ingestion Child Pipeline 1: pl\_source\_inbound\_generic\_ingestor\_child\_abc 7](#_Toc43320873)

[5.3 Ingestion Child Pipeline: pl\_source\_inbound\_generic\_copy\_abc 8](#_Toc43320876)

[5.4 Data Quality Master Pipeline: pl\_inbound\_staging\_generic\_dq\_abc 8](#_Toc43320877)

[5.5 Data Quality Child Pipeline 1: pl\_inbound\_staging\_generic\_dq\_child\_abc 10](#_Toc43320878)

[5.6 Data Quality Child Pipeline 2: pl\_inbound\_staging\_abc 12](#_Toc43320879)

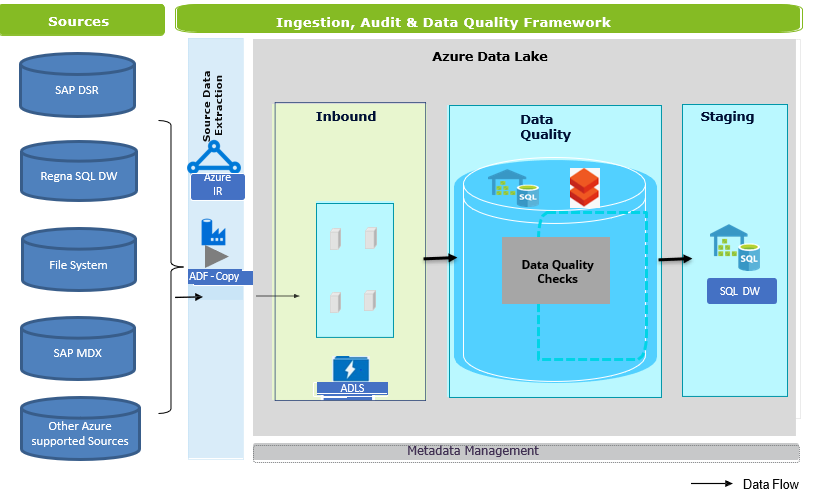
[6. Steps for Framework Integration with any pipeline 13](#_Toc43320880)

[7. Inventory Details 14](#_Toc43320881)

# Introduction

Data Lake forms an integrated environment where data from disparate systems is brought together and stored in a consistent manner. Providing new datasets at the earliest possible time; making sure the data is not stale and has good data quality are a few parameters that enables the business trust in the system. ABC framework as a whole can be categorized into 3 sub-frameworks:-

1. Ingestion Framework
2. Audit Control
3. Data Quality Framework



The above picture depicts the Architectural Diagram of ABC & Ingestion Framework. The data from multiple sources is ingested to Inbound Layer using Ingestion Framework. After the data is ingested to Inbound Layer, which is stored in Azure Data Lake Storage, Data Quality checks are performed on the inbound data and the data is loaded into Staging table using ABC-Data Quality Framework.

Here are some Features of this Ingestion framework:

* Automated Reusable Ingestion framework.
* Metadata driven ingestion to trigger the jobs based on predefined metadata configurations.
* Store and capture run stats for all Pipeline jobs, such as job start time and job end time. This information assist in Audits and debug errors.
* Configurable data quality checks.
* Data Quality checks can be done either through SQL Stored procedure or Databricks Notebook depending on the volume of source data.
* Provides a feature to check error threshold value. If the invalid records are more than error threshold, then the job is failed.
* Sends success and failure email notifications.
* It can support multiple azure data sources.
* Provides flexibility whether to reject invalid records or not.
* Dynamically creates staging tables, if it is not present.

# Technology Stack

To implement the framework the following technology stack has been used.

|  |  |
| --- | --- |
| **Ingestion and Orchestration** | Azure Data Factory V2 |
| **Storage** | ADLS Gen 2 |
| **Audit and Metadata Information** | Azure SQL DW tables and stored procedures |
| **Data Quality** | Databricks/ Stored procedures |

# Data Model

Data Model of the ABC and Ingestion Framework:



# Data Dictionary

All the tables along with their description is described in the below attached sheet:



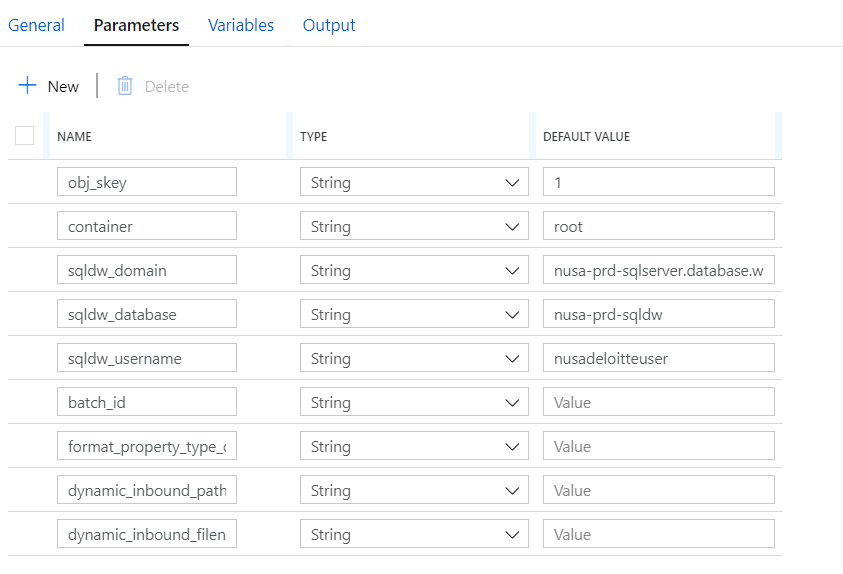
# Framework Data Flow

This Framework is a combination of Ingestion, ABC and Data Quality framework. It consists of the following pipelines:

|  |  |
| --- | --- |
| **Name** | **Description** |
| pl\_sqldw\_sqldw\_batch\_id\_abc | It is a reusable pipeline to capture logs and status of the pipelines |
| pl\_source\_inbound\_generic\_ingestor\_abc | Main Ingestion Pipeline. It sets all the variables required for ingestion. |
| pl\_source\_inbound\_generic\_ingestor\_child\_abc | Child Pipeline of the ingestion pipeline that sets all the format properties applicable. |
| pl\_source\_inbound\_generic\_ingestor\_child\_execute\_abc | Child Pipeline of the ingestion pipeline that sets all the format properties applicable. |
| pl\_source\_inbound\_generic\_copy\_with\_metadata\_abc | Child Pipeline of the ingestion pipeline that copies the datasets from source with additional metadata columns. |
| pl\_source\_inbound\_generic\_copy\_abc | Child Pipeline of the ingestion pipeline that copies the datasets from source. |
| pl\_inbound\_staging\_generic\_dq\_abc | Main Data Quality Pipeline. It sets all the variables required for DQ Checks |
| pl\_inbound\_staging\_generic\_dq\_child\_abc | Child Pipeline of Data Quality Pipeline where DQ checks are done. |
| pl\_inbound\_staging\_abc | Child Pipeline of Data Quality Pipeline where DQ checks are done. |
| pl\_sqldw\_sqldw\_update\_run\_flag\_abc | Reusable pipeline to update Run\_flag and next\_run\_date\_time in Audit.f\_pipeline\_config table |
| pl\_sqldw\_sqldw\_delete\_error\_tables\_abc | Reusable pipeline to delete error tables. |
| pl\_source\_inbound\_generic\_copy\_wildcard\_filename\_abc | Child Pipeline of the ingestion pipeline that copies the datasets from source with wildcard filenames |
| pl\_source\_inbound\_generic\_excel\_copy\_abc | Child Pipeline of the ingestion pipeline that copies the excel format datasets from source |
| pl\_fs\_sqldw\_pipeline\_template\_abc | End to End- Pipeline Template |

## Master Ingestion pipeline: pl\_source\_inbound\_generic\_ingestor\_abc

This pipeline ingests the object from source to Inbound Layer. Below are the input parameters required to execute this pipeline.



**Obj\_skey** : Primary key of audit.f\_ingest\_obj\_metadata table that specifies the object details to be processed.

**Container**: Storage Blob Container of Azure Data Lake Storage where the inbound file should be placed.

**Sqldw\_domain, sqldw\_database, sqldw\_username**: SQL Datawarehouse Connection details where the Metadata tables are stored.

**Batch\_id**: Batch number is passed as an input parameter.

**Format\_property\_type\_override**: It is an optional parameter. It is a string that contains values of the Format properties to be overridden. Sample value is shown below:

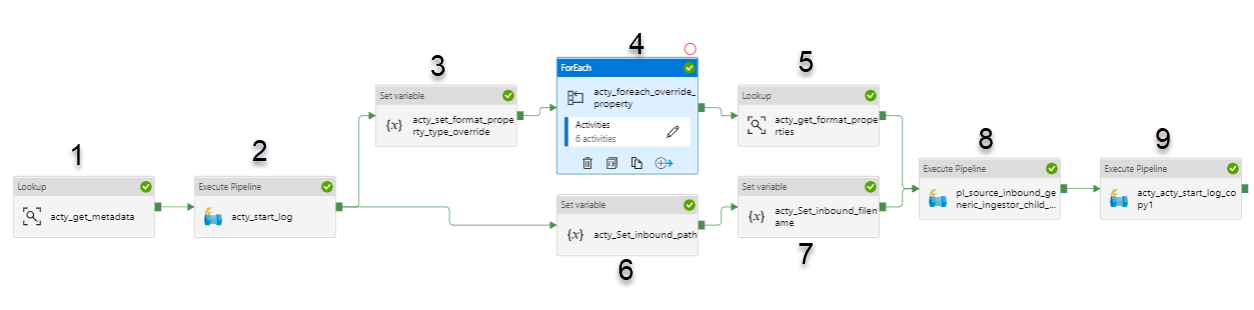
2==select \* from f\_nusa\_sre\_posdata^^5==;^^6==true

Here, 2,5,6 are the format\_property\_type\_id from audit.f\_ingest\_obj\_fmt\_properties table corresponding to below format\_property\_name. Below properties can be set using format\_property\_type\_override variable. Each format property is separated by ‘^^’ delimiter.

|  |  |
| --- | --- |
| **Format\_property\_type\_ID** | **Format\_property\_name** |
| 1 | expected\_file\_size |
| 2 | source\_query |
| 3 | source\_file\_name |
| 4 | source\_dataset\_path |
| 5 | obj\_delimiter |
| 6 | obj\_header |
| 7 | table\_skip\_rows |
| 8 | file\_skip\_rows |
| 9 | mdx\_source\_query |
| 10 | add\_metadata\_columns |
| 11 | saphana\_partition\_column |
| 12 | saphana\_partition\_type |
| 13 | wildcard\_file\_name |
| 14 | last\_modified\_date |
| 15 | excel\_format |
| 16 | excel\_range |
| 17 | excel\_sheet\_name |

**Dynamic\_inbound\_path, Dynamic\_inbound\_filename**: These are also optional parameters. If the inbound path and inbound file names are passed in these input parameters, then they override with values fetched from Metadata tables.

**Process Flow of Ingestion Pipeline:**



**Step 1:** Activity ‘acty\_get\_metadata’ fetches all the object related details from audit.f\_ingest\_obj\_metadata table.

**Step 2**: Activity ‘acty\_start\_log’ executes **pl\_sqldw\_sqldw\_batch\_id\_abc** pipeline which creates a log entry in audit.f\_pipeline\_run\_stats table specifying that the pipeline is in progress.

**Step 3**: Activity ‘acty\_set\_format\_property\_type\_override’ sets format\_property\_type\_override input parameter into a variable and converts it into an array incase there are multiple format\_property\_type\_override.

**Step 4**: Activity ‘acty\_foreach\_override\_property’ sets the dynamic format properties that are received as input parameters in their respective variables.

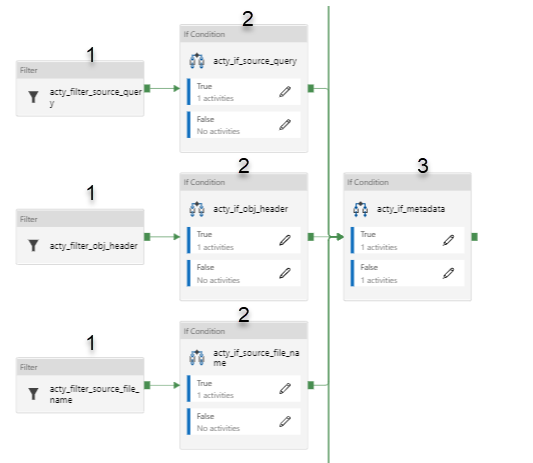
**Step 5**: Activity ‘acty\_get\_format\_properties’ fetches the format properties from [AUDIT].[F\_ObJ\_Fmt\_Properties] table that are applied to the object.

**Steps 6 &7:** Activities ‘acty\_Set\_inbound\_path’ and ‘acty\_Set\_inbound\_filename’ set inbound\_path variable and inbound\_filename variables. **Here, input parameters ‘dynamic\_inbound\_path’ and ‘dynamic\_inbound\_filename’ are given preference over the details fetched from audit.f\_ingest\_obj\_metadata table.**

**Step 8**: All the above parameters are passed to a child Pipeline ‘pl\_source\_inbound\_generic\_ingestor\_child\_abc’ and it is executed.

**Step 9:** After the completion of Step 8, Log entry in audit.f\_pipeline\_run\_stats is updated to’Success’ status. While updating the log entry to Success, ‘Pipeline\_out\_param’ is also updated with inbound\_path and inbound\_filename to be used in DQ pipelines.

## 5.2 Ingestion Child Pipeline 1: pl\_source\_inbound\_generic\_ingestor\_child\_abc



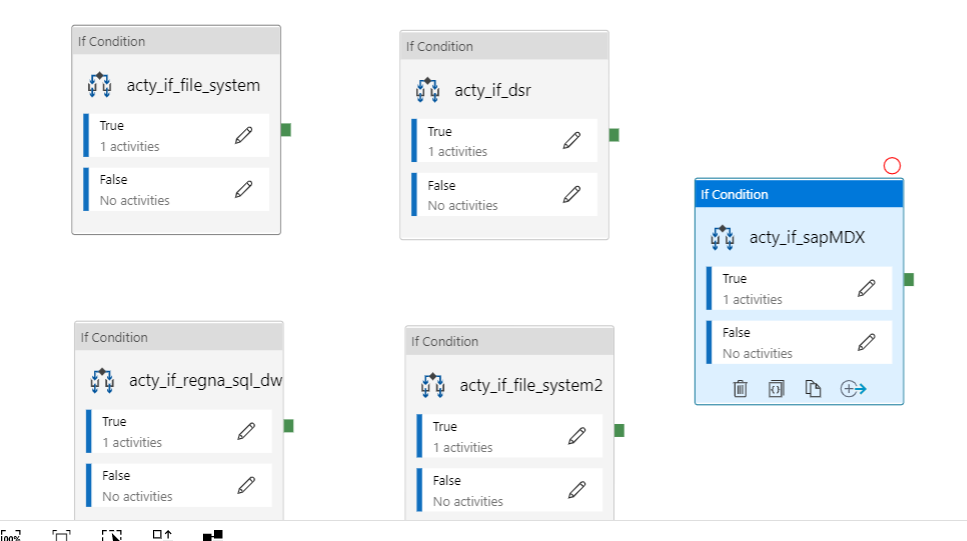
**Step 1:** Activities ‘acty\_filter\_source\_dataset\_path’, ‘acty\_filter\_obj\_delimiter’, ‘acty\_filter\_source\_query’, ‘acty\_filter\_obj\_header’, ‘acty\_filter\_source\_file\_name’ filters on the format properties fetched from [AUDIT].[F\_ObJ\_Fmt\_Properties] table.

**Step 2:** Incase the format property is filtered in step 1, then the dynamic variable set in step 4 are validated. If the dynamic variables are null, then the format properties are set in this step. In brief, **Input parameter ‘format\_property\_type\_override’ is given preference over format properties fetched from [AUDIT].[F\_ObJ\_Fmt\_Properties] table.**

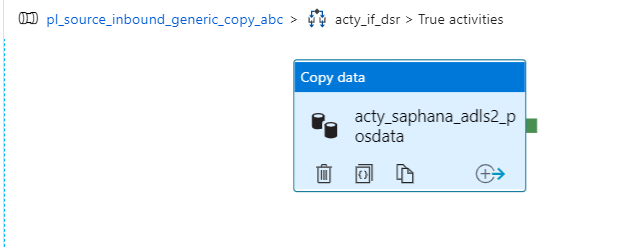
**Step 3:** Activity ‘acty\_if\_metadata’ checks whether the metadata columns to be added in inbound layer or not. Accordingly, it executes a child pipeline ‘**pl\_source\_inbound\_generic\_copy\_abc**’ or ‘**pl\_source\_inbound\_generic\_copy\_with\_metadata\_abc**’ where the object is copied from the source to inbound layer. Refer to next section for details.



## Ingestion Child Pipeline: pl\_source\_inbound\_generic\_copy\_abc



Based on the **source\_id** of the object, only one of these ‘if Conditions’ is executed. In all of these if conditions, Data copy activity is executed that copies the object from its respective source to the inbound layer.

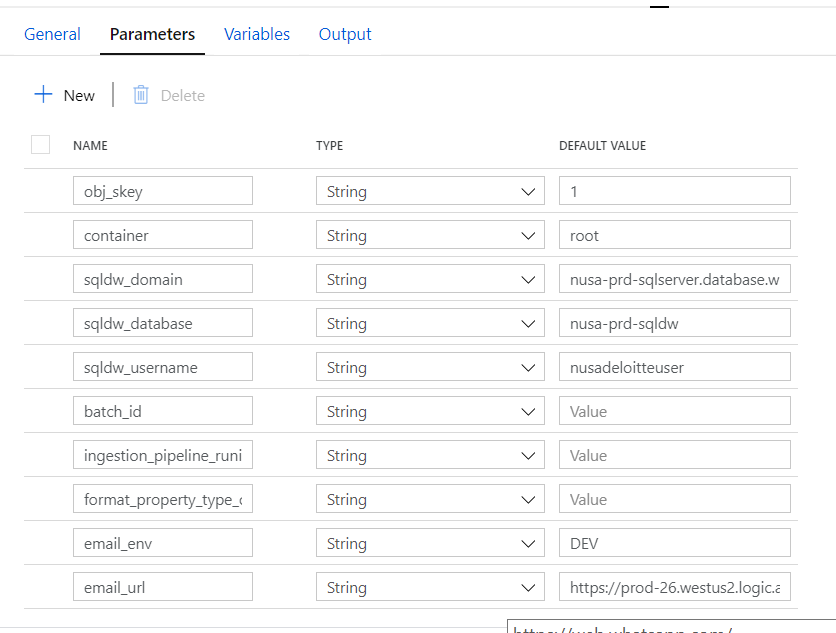


After the above two pipelines are executed successfully, the object is copied to Inbound Layer that means, **Ingestion is complete and Data quality pipelines can be executed now.**

**Pipeline ‘pl\_source\_inbound\_generic\_copy\_with\_metadata\_abc’ is similar to this pipeline. The only difference is it adds additional metadata columns along with the source columns in inbound layer.**

## Data Quality Master Pipeline: pl\_inbound\_staging\_generic\_dq\_abc

Data Quality pipeline requires below input parameters:

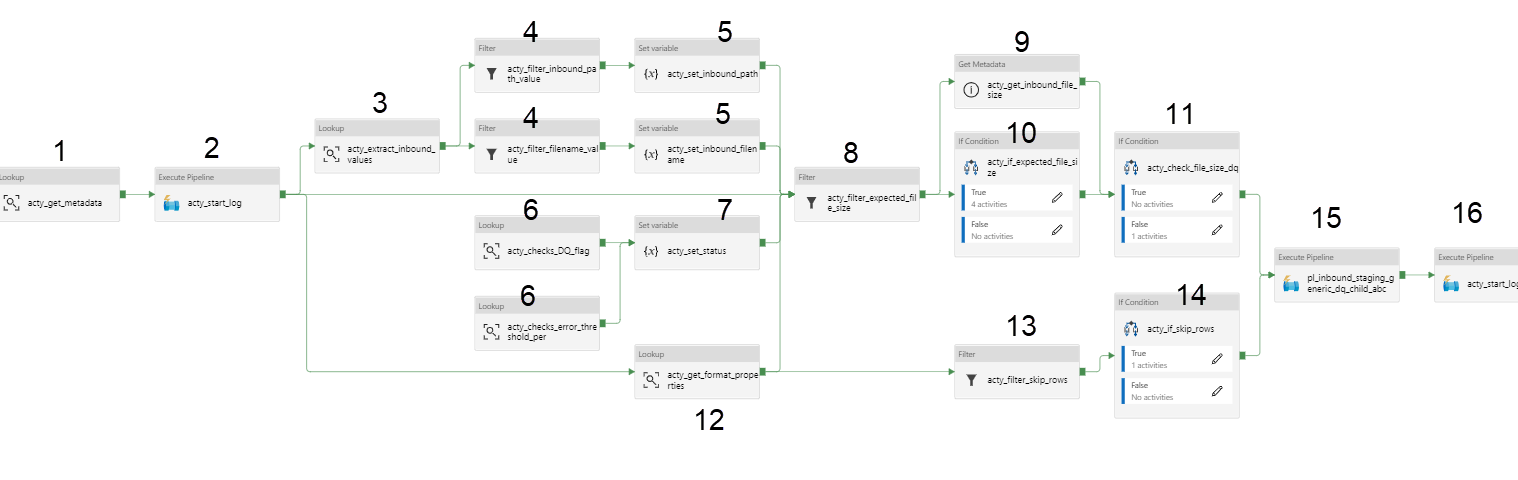


All the input parameters are same as that of Ingestion pipeline except:

**Ingestion\_pipeline\_runid**: Run Id of the ingestion pipeline.

**Email\_env, Email\_url**: These are logic app parameters for sending email notifications.

**Process Flow of Data Quality Pipeline:**



**Step 1:** Activity ‘acty\_get\_metadata’ fetches all the object related details from audit.f\_ingest\_obj\_metadata table.

**Step 2:** Activity ‘acty\_start\_log’ executes pl\_sqldw\_sqldw\_batch\_id\_abc pipeline which creates a log entry in audit.f\_pipeline\_run\_stats table specifying that the pipeline is in progress.

**Step 3:** Activity ‘acty\_extract\_inbound\_values’ retrives pipeline\_out\_param column from audit.f\_pipeline\_run\_stats table based on run\_id of the Ingestion pipeline.

**Step 4 & 5:** Activities ‘acty\_filter\_inbound\_path\_value’ and ‘acty\_filter\_filename\_value’ filters inbound\_path and inbound\_filename and stores in variables.

**Step 6:** Activity ‘acty\_checks\_DQ\_flag’ checks whether ‘File Size Check’ is enabled or not whereas Activity ‘acty\_checks\_error\_threshold\_per’ checks the error\_threshold\_per from [AUDIT].[F\_Ingest\_obj\_attr\_dq\_map] table corresponding to ‘File Size Check’.

**Step 7:** Activity ‘ acty\_set\_status’ sets the ‘Status’ variable based on Step 5.

|  |  |  |
| --- | --- | --- |
| **‘File check' DQ** | **error\_threshold\_per** | **Status** |
| Not active | Null | 0 |
| Active | Null | 1 |
| Active | Not Null | 2 |

**Step 8:** Activity ‘acty\_filter\_expected\_file\_size’ fetches the ‘expected\_file\_size’ from audit.f\_obj\_fmt\_properties table, if there is any.

**Step 9:** Activity ‘acty\_get\_inbound\_file\_size’ fetches the size of the inbound file irrespective of the Data Quality.

**Step 10:** Activity ‘acty\_if\_expected\_file\_size’ calculates the minimum expected file size and maximum expected file size incase expected\_file\_size is fetched at step 8.

**Step 11:** Activity ‘acty\_check\_file\_size\_dq’ validates File Size Data Quality based ‘Status’ variable.

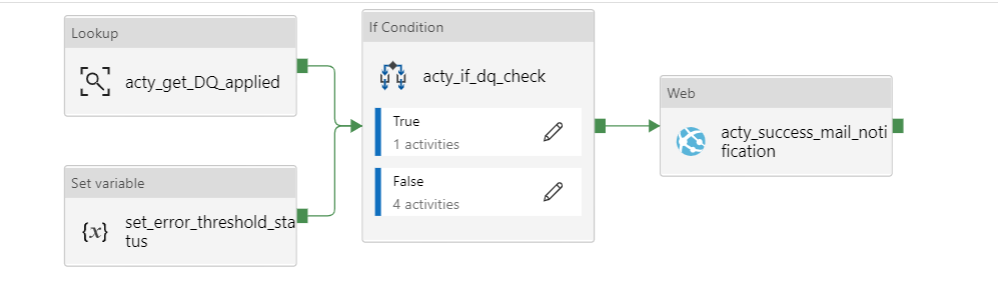
**Step 12:** Activity ‘acty\_get\_format\_properties’ fetches the format properties from [AUDIT].[F\_ObJ\_Fmt\_Properties] table that are applied to the object.

**Step 13 & 14:** Activity ‘acty\_filter\_skip\_rows’ filters on the format properties fetched from [AUDIT].[F\_ObJ\_Fmt\_Properties] table.

**Step 15:** Activity ‘pl\_inbound\_staging\_generic\_dq\_child\_abc’ executes the child pipeline ‘pl\_inbound\_staging\_generic\_dq\_child\_abc’ where the data quality is done based on these input parameters. All the parameters and variables set in this pipeline are passed to the child pipeline.

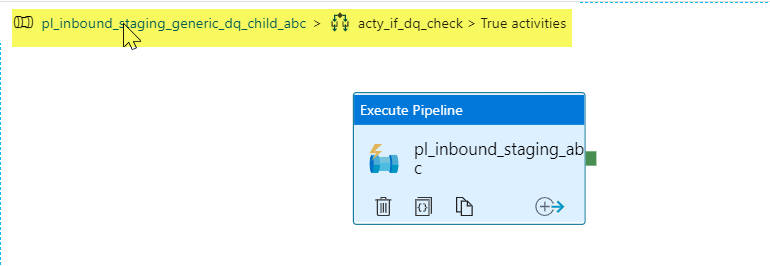
**Step 16:** After the completion of Step 17, Log entry in audit.f\_pipeline\_run\_stats is updated to ’Success’ status.

## Data Quality Child Pipeline 1: pl\_inbound\_staging\_generic\_dq\_child\_abc

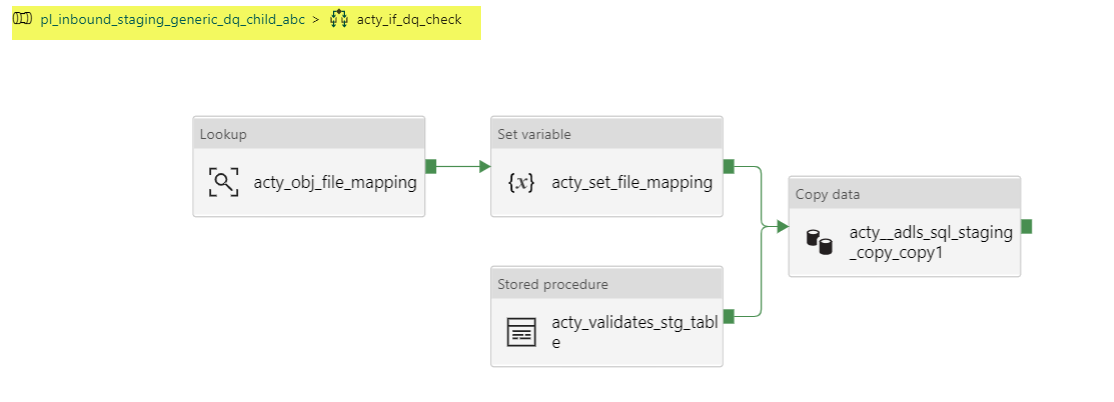


**Step 1:** Activity ‘acty\_get\_DQ\_applied’ checks if there are any active DQs from [AUDIT].[F\_Ingest\_obj\_attr\_dq\_map] for that object. Activity ‘set\_error\_threshold\_status’ checks if the overall error threshold is present or not in Audit.f\_ingest\_obj\_metadata table.

**Step 2:** Activity ‘acty\_if\_dq\_check’ takes the output from previous activity and checks if the value of ‘err\_threshold\_per\_overall’ parameter. If both the parameters are null, then ‘FALSE’ part is executed, otherwise ‘TRUE’ part is executed.



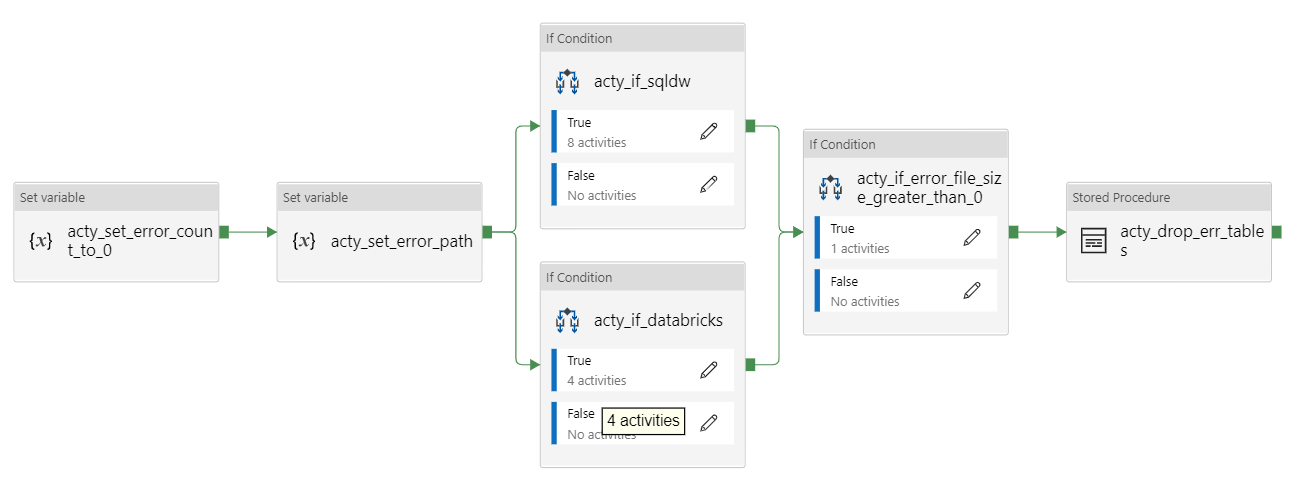
If either ‘err\_threshold\_per\_overall’ parameter is not null or if there are any active DQs for that object, then another child pipeline ‘pl\_inbound\_staging\_abc’ is executed. For more details on this child pipeline, refer to next section.



**Note : If ‘err\_threshold\_per\_overall’ parameter is null and there are no active DQs, then the above activities are executed. Activity ‘acty\_validates\_stg\_table’ creates the staging table if it is not present and Activity ‘acty\_adls\_sql\_staging\_copy\_copy1’ copies the inbound file to staging table.**

**Step 3:** Once the step 2 is completed, then Success Mail notification is sent to Team using the below Logic App ‘nusa-dhpr-dv-abc-mail-notification’.

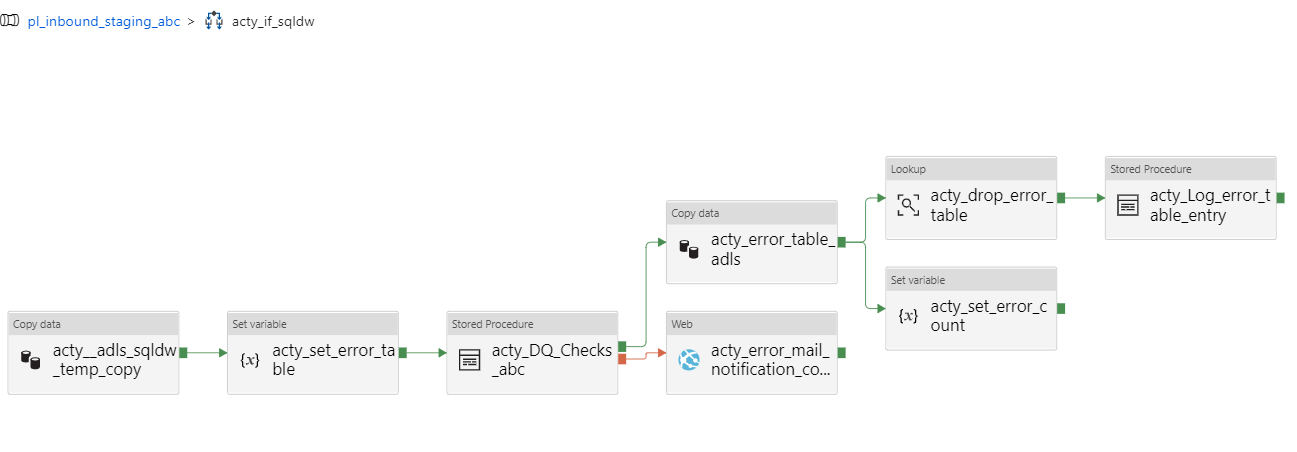
## Data Quality Child Pipeline 2: pl\_inbound\_staging\_abc



**Step 1:** Activities ‘acty\_set\_error\_count\_to\_0’ and ‘acty\_set\_error\_path’ sets ‘error\_count’ and ‘error\_path’ variables.

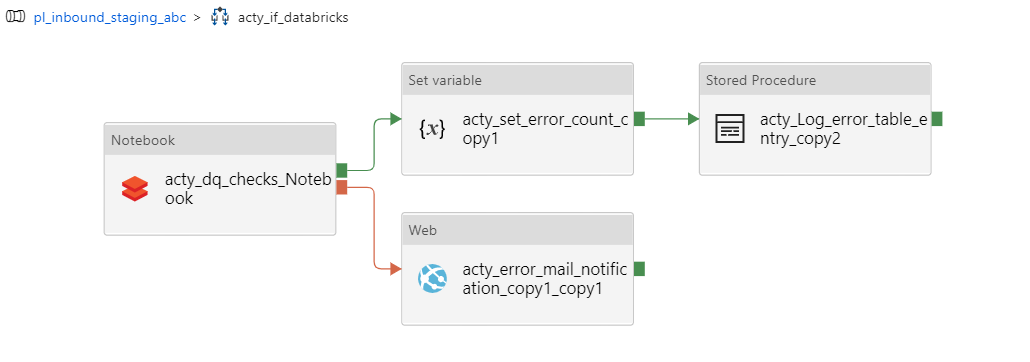
**Step 2:** Based on the value of ‘IncDataVol’ parameter, activities ‘acty\_if\_sqldw’ and ‘acty\_if\_databricks’ are executed. If IncDataVol=’SQL DW’, then ‘acty\_if\_sqldw’ is executed, otherwise ‘acty\_if\_databricks’ is executed.

**Step 2(a):** **If IncDataVol=’SQL DW’**, then the below activities are executed.



* Activity ‘acty\_adls\_sqldw\_temp\_copy’ copies inbound file to a temp table.
* Activity ‘acty\_set\_error\_table’ sets error\_table variable which is passed in subsequent activities.
* Activity ‘acty\_DQ\_Checks\_abc’ executes Data Quality stored procedure where the data quality checks are done and the data is loaded to Staging Layer.
* If there is any error while performing data quality checks, then Error notification is sent to Object owners, otherwise ‘acty\_error\_table\_adls’ activity copies error table to Azure Data Lake Storage.
* After the error records are copied to error file at Data Lake Storage, error table is dropped and the count of error records is stored in error\_count variable.
* Activity ‘acty\_Log\_error\_table\_entry’ inserts a log entry in audit.f\_ingest\_err\_records\_facts with error file details.

**Step 2(b):** **If IncDataVol=’databricks’**, then the below activities are executed.



* Activity ‘acty\_dq\_checks\_Notebook’ performs the Data Quality checks applicable for that object and loads the data into Staging Layer.
* If there is any error while performing data quality checks, then Error notification is sent to Object owners.
* Activity ‘acty\_set\_error\_count\_copy1’ captures the count of the invalid records based on data quality checks.
* Activity ‘acty\_Log\_error\_table\_entry\_copy2’ inserts a log entry in audit.f\_ingest\_err\_records\_facts with error file details.

Step 2 covers Data Quality checks and loads the inbound file to Staging Layer.

**Step 3:** Activity ‘acty\_if\_error\_file\_size\_greater\_than\_0’ checks the count of invalid records while doing the data quality checks. If the record count is greater than 0, then ‘Failure’ email notification is sent to Object owners.

This completes Data Quality pipeline execution.

**Pipeline ‘pl\_sqldw\_sqldw\_delete\_error\_tables\_abc’** executes ‘[AUDIT].[SP\_DROP\_ERROR\_TABLES]’ stored procedure which drops all error tables that were created before 24 hours.

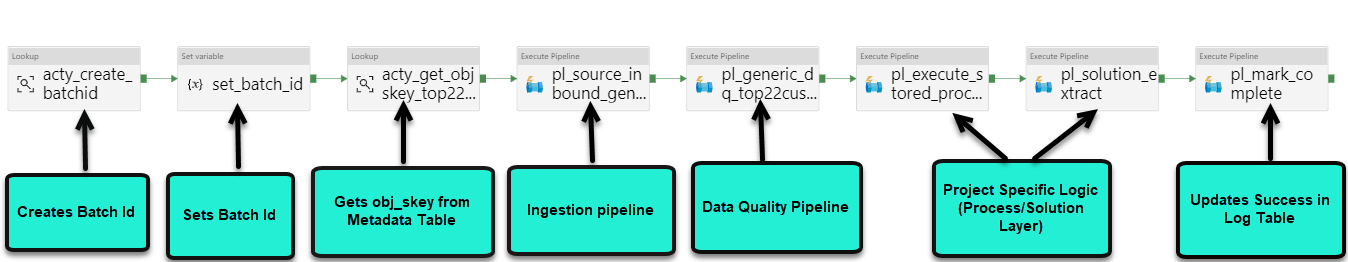
# Steps for Framework Integration with any pipeline

**Step 1**: Framework Integration requires few entries in Metadata tables. **Make sure all the metadata tables are loaded with proper entries.** Metadata entries can be made using the below sheet. Also, the stored procedure specified in the sheet should be executed in the order of their sequence.



**Step 2:** This Framework Integration requires integration of 2 pipelines: Ingestion Pipeline **(**pl\_source\_inbound\_generic\_ingestor\_abc) and Data Quality Pipeline (pl\_inbound\_staging\_generic\_dq\_abc).

Below picture shows a sample pipeline in which Ingestion and DQ pipelines are integrated.



* Batch\_Id can be created using reusable stored procedure ‘[AUDIT].[SP\_ISRT\_PIPELINE\_RUN\_STATISTICS\_ABC]’
* Obj\_skey can be fetched from the Audit.F\_ingest\_obj\_Metadata table which is passed in Ingestion and Data Quality pipelines using th below query:

SELECT obj\_skey

FROM [AUDIT].[F\_Ingest\_Obj\_Metadata]

WHERE obj\_name = <obj\_name>

* Ingestion and Data Quality pipelines are executed after the batch\_id and obj\_skey is fetched from the tables. Make sure the input parameters of these pipelines are properly configured, as explained in detail in Section5.1 and Section5.4.
* At the end, reusable pipeline ‘pl\_sqldw\_sqldw\_batch\_id\_abc’ is called which updates the log entry in Audit.F\_pipeline\_run\_stats table as ‘Completed’.

**Step 3**:

In case of new source addition, new source must be added in Pipelines ‘pl\_source\_inbound\_generic\_copy\_abc’ and pl\_source\_inbound\_generic\_copy\_with\_metadata\_abc’, explained in detail in section 5.3.

# Inventory Details

