

HyperSense User Guide

Version 2.5.13.3



3. **Share:** Allows you to share the selected table with the selected **Users** or **User Groups**. On click, a [Share](#) window is displayed.
4. **Settings:** Allows you to perform required settings for any table instance. On click, it will display the [Configure Grid](#) and [Color Rules](#) of respective table instance.
5. **Fx:** Allows you to fetch the data from the data tables using predefined functions. For more information, see [Fx](#) section.
6. **Basic:** Allows you to filter the records based on the filter condition configured. On clicking it, the **Choose Fields** pane is displayed. Search for the columns and configure the conditions associated with it. For more information, see [Basic Filter](#).
7. **Advanced:** Allows you to filter the records based on the expression defined. For more information on expression filter, see [Advanced Filter](#).
8. **Date Selector:** Allows you to fetch the records based on the values configured in the Date Selector. Date selector option will be displayed only for partitioned tables. For more information on date selector, see [Date Selector](#).

The table instance window has the following bottom toolbar options:

1. **Jump to Row number:** Allows you to jump from the required row by entering the row number in jump to row no field. For more information click [here](#).
2. **Jump:** Allows you to jump to the first page from current page in the table instance resultant window. For more information click [here](#).

Jump to row number

This feature allows you to jump to the selected record number. Enter the required row number and hit enter. After search, the searched record will be highlighted.

Pagination

At the bottom pane, it displays the total number of records available and the current page record number that you reside in.

Jump to top

This feature allows you to jump to the first page from current page in the table instance resultant window.

To Jump from any page to top:

- From the table instance resultant window, click **Jump to Top** icon at the bottom right corner of the page. You are landed on the first page.

Expandable

Last updated on November 13, 2024

Expandable feature allows you to fetch the required record details from the data table by accessing through a link configured. The link configuration is done from UI side.

Expandable has the following steps:

- **Configuring Link for Expandables**
- **Viewing Records using Expandables**

Configuring Link for Expandable

To create link for Expandable:

1. From the workspace of **Data Catalog**, go to the required table for which you want to create link and click **View** icon as highlighted.
2. The data table screen of selected table is displayed. Click the **Links** tab beside the **Summary** tab.
3. The **Links** tab screen is displayed.
4. Enter the required link name in search bar and hit enter. The required link is displayed on the screen.
5. Click **Create New Link** at the top right of the pane. The application display the options. **Table** and **Dashboard**.
6. Select **Table** for other link creation OR **Dashboard** for BIS link creation. For Dashboard link, refer to [Navigation to Dashboard](#).
7. The **Create New Link** window is displayed. Below are the fields and options available in the window.
New Link Name : Enter the name of the new link as required.
Destination Table: Select the destination table from the **Destination Table** drop-down.
Source Column: Select the source column in **Source Column** drop-down in the **Map Columns** Pane.
Destination Column: Select the destination column in **Destination Column** drop-down in the **Map Columns** pane. For Dashboard link configuration, refer to [Dashboard Link Configuration](#).
8. After selecting the options, click **Save to save the set up**. Once selected, destination table user will get expression filter panel, where you can add filter while creating links which is applied in destination table.
9. The created link is displayed on the list.
10. To edit the link, click the **Edit icon at right side of the pane**.
11. The **Edit Link** window is displayed. Select the required options from source and destination **Select** drop-downs. Click **Update** at the bottom of the pane. User can edit the filter expression.

12. To delete the link, click **Ellipsis and select Delete**. The link will be deleted.
 13. To view the link, click **Ellipsis icon and select View**. The link can be viewed.
-

Viewing Records using Expandables

To view the records using Expandable feature:

1. Go to **Data Catalog workspace** > Click **Search** icon of any table instance > you are navigated to the respective table instance in a new window. The selected table name will be the tab name.
 2. In the table instance, from the data catalog. Right click the required row and select the required entity from **Expandable** options.
 3. You can search the link name from the search bar.
 4. The selected details are displayed in a new window with applying expression filter of selected row data for expandable.
 5. You can access cell links through clicking on **Jump to Cell Link** icon from the cell and select the entity under **Expandables** option. You can also search link name from the search bar.
 6. The selected details are displayed in a new window with applied expression filter of selected cell data for expandable.
 7. To view or make changes to expression filter, click the **Expression Filter** and make changes to expression as required. Click **Apply**.
 8. The records are fetched as per the expression filter configured.
-

Advanced Query Manager

Last updated on June 13, 2025

Advanced Query Manager is a system that executes trino queries from data catalog to view or join compacted tables. You can export the data in different formats and the exported data can be used for your business purpose.

Run Query

To run trino queries in **Advanced Query Manager**:

1. From the Data Catalog workspace, click **Settings** icon and hover over **AQM** and select **Run Query**.
2. The **Advanced Query Manager** screen is displayed with a new tab.
3. Configure the following options:

- **Select Connection:** This dropdown option provides the list of connections available. You can click the dropdown and select the required connection name.
 - **Default Query:** This dropdown option provides the list of queries that are available by default. You can click the dropdown and select the required query from the list.
 - **Add:** Click **Add** to add your query to the query pane.
4. Configure the tool bar options on the new tab pane.
 - **New Tab:** Click **New Tab (Plus)** icon to add a new tab in order to add your new query in it. You can add upto 30 tabs here.
 - **Write your query here:** You can write your query in this pane.
 - **Validate:** This icon allows you to validate the query that you have entered.
 - **Run Query:** This icon allows you to run the query that you have entered.
 - **Clear:** This icon allows you to clear the query that you have entered.
 - **Save as Template:** This icon allows you to save your query as template.
 - **Export:** This icon allows you to export the data from query results in different formats such as PDF, CSV, RTF and TEXT.
 5. After selecting connection and query details, click **Add**. The query expression is fetched as per your selection.
 6. You can also enter your query manually in the pane available and validate the same. Click **Save as Template** to save this query as a template.
 7. **You can not execute table manipulations queries like Alter queries, Delete queries etc.**
 8. To validate the query, click validate icon. A snackbar message is displayed after validation.
 9. If validation is failed, it displays the same as snackbar message.
 10. After successful validation of query, click run icon. It executes the query and fetch the required data.
 11. **Click export icon and select the required format to export the data. The data is exported in the selected format.**

Example when Advanced Query Manager (AQM) is used

HyperSense is provided with Advanced Query Manager (AQM) to execute trino queries from data catalog to view or join the compacted tables easily.

Let us consider an example, using **Advanced Query Manager**, user has applied **JOIN** query on two compact tables **aqm_50mn_tbl1** and **aqm_50mn_tbl2**. After query validation, if you run the query, **query manager joins the two compact tables and displays the resultant records in Advanced Query Manager** screen as shown in the below image. This is how you can use **AQM** to join different trino compacted tables in HyperSense.

Schedule Query

This option allows you to create and schedule a query from **Advanced Query Manager**.

To create and schedule a query in **Advanced Query Manager**:

1. From the Data Catalog workspace, click **Settings** icon and hover over **AQM** and select **Schedule Query**.
2. The **Schedule Query** window is displayed.
3. Click **New Tab** icon to create a new Query. Upon clicking the **Create Query** icon, the **Create Query** window is displayed.
4. Under **Query Details** pane, configure the following fields:
 - a. **Query Name**: Enter the query name. **Example**: QM_001
 - b. **Maximum Rows**: Enter the number of rows that you want to display in the output. **Example**: If there are 10 rows as resultant data of query measure, and if we have set maximum rows 6. It will display only 6 rows in the output.
 - c. **Select Target Store**: It lists all the connected database and allows you to select the target database type. The output data will be stored in the selected database.
 - d. **Return Distinct Rown Only**: If this check box is selected, rows with similar data are not to be repeated.
 - e. **Notify via Email**: If this check box is selected, the configured user gets the email notifications.
 - f. **Notify via SMS**: If this check box is selected, the configured user gets the SMS notifications.
5. **Create Query** window has the following tabs:
Add Source, **Join**, **Select Columns**, **Filter**, **Preview**, **Reporting**, and **Schedule**. Configure each tab as explained below steps.
6. **Add Source**: Add the source tables, on which the query must be executed. The **Add Source** tab has the **Select Tables** and **Columns** panels:
 - a. **Select Tables**: This pane has the list of tables available in the Data Catalog. You can select the required tables, on which you want to execute the query measure. It has the features as below:
 - **Search / Filter Fields**: Allows you to search the tables.
 - **Show Selected toggle switch**: When enabled, it displays the selected tables.
 - To add the data tables for a query measure, go to **Select Tables** pane > Hover the mouse on the tables instances. The plus icon is viewed. Click **Plus** icon.

Note:

- If a table has **Plus** icon associated with it, it indicates the table is not selected for Advanced query measure.

- If a table has **Plus icon** associated with it, then it indicates the table is selected for Advanced query measure.

- **Columns:** This pane lists all the columns available in the selected table. Example: In the below mentioned example, we have added subscriber table and cdr table. Out of which, cdr is selected. The data associated with the cdr table are displayed in the Columns pane.
- **Hourly Timestamp Columns(s):** It allows to process the data as per the selected date and time stamp column. When Hourly Timestamp Column(s) is not selected, records are processed based on partition column.

7. **Joins:** You need add the join conditions in this tab. If more than one table is available as input to the Query Measure, you must create a join to link all the tables and extract the required data. To perform join on any data in query measure:

- Go to **Joins** tab. The added table instances are seen in the configuration pane.
- Connect the two table instances. The connected instances is viewed as displayed:
- Click the dark circle between the two instances. The Join Details window is displayed. Select the required join operation.

Following are the operations associated with join:

- **Inner Join:** A join of two tables that returns records for which there is a matching value in the field based on which the tables are joined.
 - **Right Join:** The right outer join returns all the records from the right table and those records from the left table that matches the condition.
 - **Left Join:** The left outer join returns all the records from the left table and those records from the right table that matches with the condition.
 - **Full Outer Join:** The full outer join retrieves all records from both the left and the right table.
- Click **Add New Row** option. A new row is added. Complete the following fields:
 - **Column1: Subscriber Column:** Displays the columns from the first table based on the data catalog imported. Select the required column, based on which you want to define the match condition.
 - **Column2:** Displays the columns from the second table based on the data catalog imported. Select the required column, based on which you want to match the variable selected in Column1.
 - **Numeric Operator:** Select the operator to perform action between Column1 and Column2.

- e. Similarly, you can add multiple join conditions. When you add multiple conditions, the column And/Or is enabled.
 - **And/Or:** If you want to consider all the configured conditions, select AND. If you want to consider either of the configured conditions, select OR.
- f. Click **Apply**.

Note:

- In AQM, when you try to join the two columns from the old tables with the data types varchar and text, there is a mismatch in the column data types that results in error while validation. In this scenario, the **text data type automatically converts into Varchar** automatically in the backend and proceed with table joining process in the backend without any error.

8. **Select Columns:** This tab allows you to select the required output columns from each of the input tables that are added. These columns are returned as part of the result when the query is run. For more information, see [Select Columns](#).
 - a. **Select Columns:** The Select Columns tab allows you to select the required output columns from each of the input tables that are added (in Select Tables tab). These columns are returned as part of the result when the query is run. Select Columns tab has the [Select Columns](#) and [Selected Columns](#) panels.
 - b. **Select Columns Pane:** The Select Columns pane contains all the columns of the table selected as a tree structure.
 - c. **Selected Columns Pane:** The Selected Columns pane contains all the columns which you select to be part of the result returned by the Query Measure. To select the columns:
 - Drag and drop the columns of any Table Instance from **Select Columns** pane to **Selected Columns** pane. You can also add with the double click of the respective column in a Table Instance.
 - After you drag and drop the columns, the column options displayed in Selected Columns pane. Those are.
 - **Aggregate:** Aggregate functions are built-in functions to perform calculations on data. Aggregate functions return a single value, calculated from multiple values in a table column. Select the required aggregate functions. It has the following functions:
 - **Sum:** Calculates the sum of all the values in the selected column.
 - **Count:** Calculates the total count of values in the selected column.
 - **Min:** Identifies the least value in the selected column.
 - **Max:** Identifies the largest value in the selected column.
 - **Statement:** You can select the statement functions to perform the action. For more information, refer to [Statement Functions](#).
 - **Column Name:** Displays the column name.
 - **Display Name:** Edit or change the display name of the column if required.

Note:

- When you select same column again, message is displayed "**Duplicate display name**".
- If you want to use same column, change the display name.

- **Type:** By default a type associated with selected column is displayed. You can change to different type if required.

Note:

In **Type**, you can select the **Date** data type from the dropdown list for required row to view the date column for particular data.

- **Partition Column:** It is used to get partitioned table output. It works only for Trino Table.
 - **PR Enable:** Toggle switch is available for timestamp column. Enable the switch for the selected column, it will act as non partition column. Toggle switch must be enabled to generate PR.
 - **XDR Enable:** Enable Toggle switch to select the XDR column. Atleast one XDR column is required to generate participating record (PR). XDR column hold the unique key to generate PR. Datatype should be bigint. Toggle switch must be enabled to generate PR.
 - **Sort By:** Select the sorting order for the output column after a query measure is run. Example: if we select asc, the output columns are listed in the ascending order.
 - **Delete:** At the right hand side corner of table header, you can see the delete button. Click to delete all the rows. On hover of each row in **Selected Columns**, you can see delete. Click to delete the corresponding row.
 - Click **Save and Continue**.
9. **Filter:** This tab allows you to add an expression or filter to select the criterion that is used to select rows in the Output Table. The entries in this panel are optional steps and can be bypassed, if required. For more information follow the below steps:
- Filters tab** has **Choose Condition** and **Choose Having Condition by Group by Filter** panes.
 - Choose Condition:** To configure the filter conditions follow below steps:
 - Click **Add New Row**. A row is added.
 - Configure the following:
 - **Clause:** Select the clause (where, and, or) required for your conditional expression.
 - **Expression(1):** The selected table's columns are available in the list. Select the first expression for the condition.
 - **Operator:** Select the operator (=, !=, like, not like, <=, >=, >, <) required to define your expression.

- **Like Operator:** It allows you to return the records that matches the specified name in the configuration.
- For example, below is the data table with multiple records.
- Configure the **like operator** expression in the advanced filter as **name Like Victoria** as shown below.
- The expression filter fetches the data with the name **Victoria**.
- **Not Like Operator:** It returns all the records except those that match the given string.
- **Expression(2):** The selected table's columns are available in the list. Select the second expression in the condition.

Note:

- You can use *Absolute / Relative Date* to filter out the record. To know more refer, *Use of Absolute / Relative Date in Expression (2)*.

- **Delete:** click delete icon, to delete the row if not required.
 - **Go back to the** *Choose Having Condition by Group by Filter*.
- c. **Choose Having Condition by Group by Filter:** To configure the filter conditions:
- Click **Add New Row**. A row is added.
 - **Configure the following:**
 - **Clause:** Select the clause based on the requirement (Having, and, or).
 - **Aggregate:** Select the aggregate function (sum, count, min) for an expression.
 - **Expression(1):** The selected table's columns are available in the list. Select the first expression for the condition.
 - **Operator:** Select the operator (=, !=, like, not like, <=, >=, >, <) required to define your expression.
 - **Aggregate:** Select the aggregate function (sum, count, min) for the second expression.
 - **Expression(2):** The selected table's columns are available in the list. Select the second expression for the condition.
- d. Click **Save and Continue**.

Configured Example: The below example has the following conditions configured: where Subscriber ID = CDR ID and Subscriber Phone Number = CDR Phone Number Having (min CDR value > min Subscriber value).

Tip:

The above example has the following conditions configured:

- where Subscriber ID = CDR ID and Subscriber Phone Number = CDR Phone Number
- Having (min CDR value > min Subscriber value);

10. **Preview:** Click to preview the configured query measure. Test the query. After the query is run, you can view them in the output.
11. The Preview tab has the following configuration expandable:
 - a. **Query:** Click this expandable menu. A query is displayed in the expanded view. Click **Test Query** to test the configured query. After the query has run successfully, you are notified with the successful message in the grid.
 - b. **Output Table:** Click the expandable to view the output data of executed query.
12. **Reporting:** This tab allows you to configure the table details to store the resultant data. Configure the following:
 - a. **Enable Report Table:** Click and enable the **Enable Report Table Toggle** switch.
 - b. **Reporting Table:** Enter the reporting table name, where the output data must be stored. If the name is not specified and left blank, then the system generates the table name automatically.
 - c. **Truncate before Load:** If selected, will not allow you to load duplicate data.
 - d. **Specify Tags:** It allows you to segregate the table in the Data Catalog. Click on the **Type to add new** and then manually create the required tag or select the available tag from the drop down list. For more, refer to the [Data Catalog](#).
 - e. Click **Next**.
13. **Schedule:** This tab allows you to schedule the configured query. Configure the following details:
 - a. **Frequency:** This tab allows you to configure the frequency for scheduling the query.
 - i. **Frequency:** This option allows you to set the frequency of scheduled query. Click the **Frequency** dropdown and select the required option. For example, if you select Daily from frequency dropdown, the schedule runs on daily basis as configured.
 - ii. **Repeat Every:** This option allows you to set the frequency period. Click the **Repeat Every** textbox and enter the number for the frequency entity or you can click the upward arrow which increments the default value by one. The default value is 1. For example, if you select frequency dropdown as daily, then configure repeat every as 1 days or 2 days and so on.
 - iii. After configuring **Frequency**, click **Next**.

- b. **Day Group:** This tab allows you to configure the Day Group for **Default Runtime** and **Manual Runtime**.
 - i. **Default Runtime:** This option allows you to select and create a day group, **Start Time** and **End Time**.
 - ii. **Manual Runtime:** This option allows you to select the Day Group and Run Time from respective **Day Group** and **Run Time** dropdowns.
 - iii. After configuring **Day Group**, click **Next**.
 - c. **Lookback:** This option allows you to configure the lookback period for a query to be scheduled.
 - i. **Absolute:** This option allows you to select the absolute time period with a configurable **Start Date** and **End Date**. Select start date and end date from **Start Date** and **End Date** calendar options.
 - ii. **Relative:** This option allows you to select a relative time period that includes From period and To period. Select the required period.
 - iii. After configuring **Lookback** period, click **Next**.
 - d. **Exceptions:** This option allows you to add exceptions for the scheduled query. If you configure a specific day group, then your query does not run
 - i. To add exceptions, click **Add Row** and configure the following:
 - ii. **Day Group:** You can select a day group for which the query should not run for that specific configured timelines.
 - iii. **Start Time:** Select the start time from the **Start Time** drop down.
 - iv. **End Time:** Select the end time from the **End Time** drop down.
 - v. **Rule:** Select the rule from the rule dropdown. This fixed and by default **Week Ends**.
 - vi. **Delete:** You can click delete icon to delete the configured exception.
 - vii. **Add Row:** You can click **Add Row** to add a new row to configure another row.
 - e. After configuring exceptions, click **Submit** button.
 14. You can select **Run Now** check box to run the query now. Click **Submit**.
 15. **Preview** : It allows you to view the parent table of a column. On the canvas, use single-click to select a **Query Measure operator**, and click **Results Preview** at the bottom of the canvas. Table records details will be displayed.
 - a. **To view the parent table:** Single-click on the required column, the row gets selected.
 - b. Now, right-click on the row to open menu and go to **Drilldown** and click on the displayed parent table name.
 - c. Parent Table is displayed.
 - d. To learn more about Result preview, refer to the [Result Preview](#).

2. *In Version History window, use latest version to create version*

To create new version from **Version History** window, Click Vertical Ellipsis of Latest Version > Click **Create Version**.

Let us consider below example:

1. In below Version History window, there are two versions: 1.0 and 2.0. Here, Version 1.0 is **lower version whereas Version 2.0 is the latest version**.
2. **If you try to create new version from older version (Version 1.0), snackbar message is displayed as "Create version is only allowed from the latest version of the pipeline".**
3. Use latest version to create version. Click Vertical Ellipsis of Latest Version(i.e., 2.0) > Click **Create Version**.
4. **New Version 3.0 is created. It is the latest version. Pipeline status for this version is Drafted and it is not marked as Active.**

Note:

In following cases new version is not created:

- **In Pipeline Repository window, if you edit pipeline whose status is either Drafted or Deployed.** In this case, pipeline active version is edited. No new version is created.
- **In Version History window, if you edit the latest version. In this case, latest version is edited.** No new version is created.

Version History Status Changes

Whenever you activate any version available in the Version History window, you can notice the changes in the pipeline status.

Let us consider below example:

1. In Pipeline Repository window, pipeline status is **"Running"**, Current version is 1.0, and Latest version is 3.0.
2. In Version History, status of version 1.0 is **"Running"** and is marked **Active**.
3. When you activate version 2.0, status of version 1.0 changes to **"Deployed"**. Version 2.0 is marked **"Active"** and status is **"Drafted"**.
4. In Pipeline Repository window, pipeline status changes to **"Drafted"**, Current version is 2.0, and Latest version is 3.0. Schedule and Run the pipeline, status will change to **"Running"**.
5. Once Pipeline status changes to **"Running"**, in Version History status of version 2.0 changes to **"Running"**.

Features

Pipeline

Data Management Studio handles file collection and parsing of data based on the pipeline created.

Pipeline depicts the complete flow of data processing. It has the following grids:

- Data Source
- Transformation Operator
- Data Sink

To create a pipeline:

1. Configure the Data Source, Transformation Operator, and Data Sink.
2. Connect the components.

Pipeline

How to Create a Pipeline?

[Last updated on May 19, 2025](#)

The pipeline organizes and automates the flow of data from extraction to transformation to loading. **This makes sure that the data integration process goes smoothly and quickly. It may include work-flows that make it possible to extract, change, and load data.**

ETL pipelines are crucial in data integration and data warehousing scenarios, where data needs to be consolidated from multiple sources, cleaned, transformed, and loaded into a unified and consistent format for reporting, analytics, or business intelligence purposes.

To create a new pipeline:

1. Click **Create Pipeline** available on upper right corner side.
2. **Create New Workflow** window is displayed.

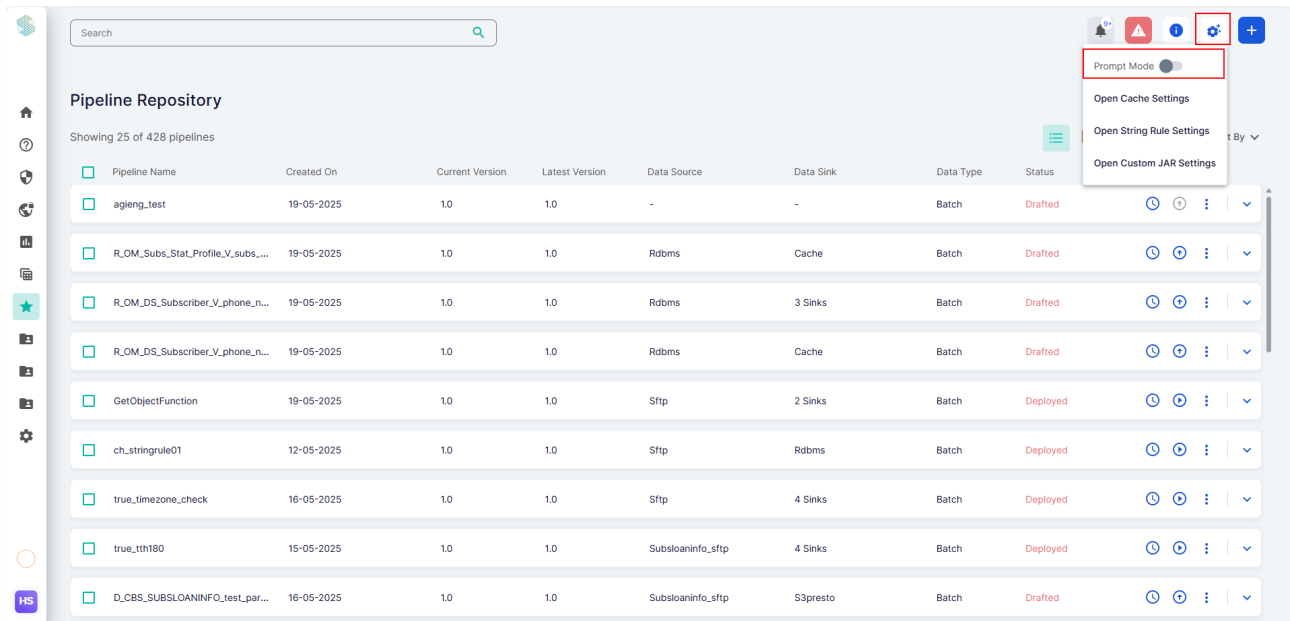
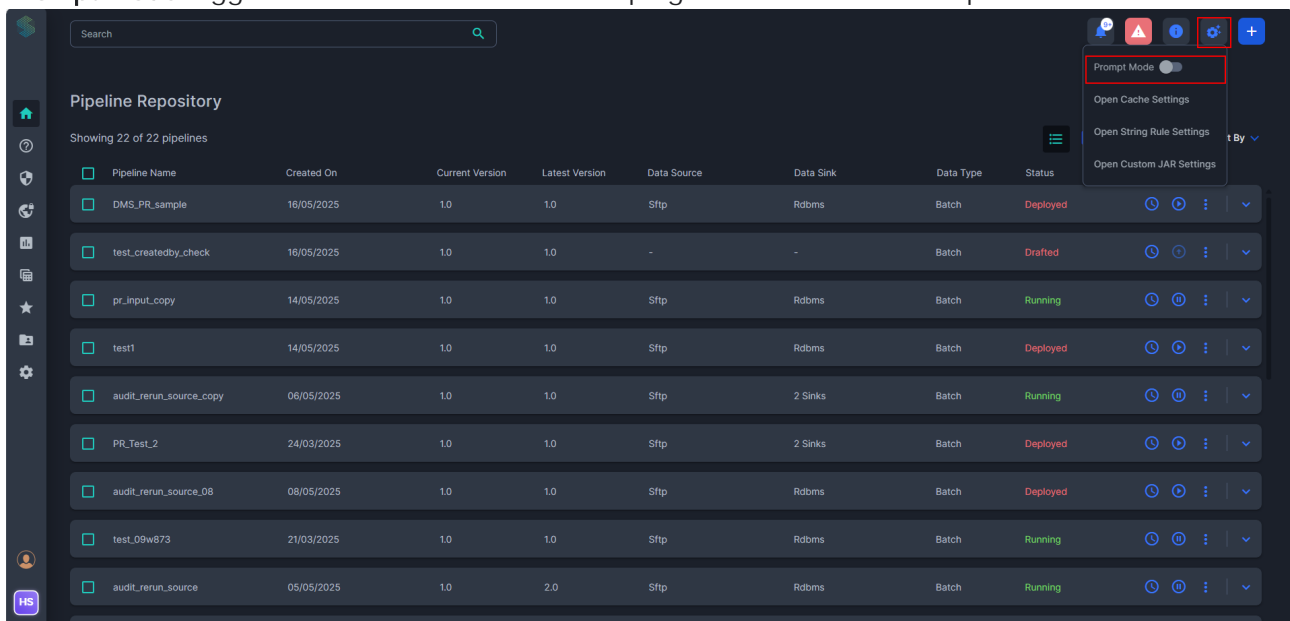
Complete the below details:

- **Pipeline Name:** Pipeline name is unique name which helps to identify and distinguish from other pipeline. Enter a unique pipeline name.
- **Add Tags:** Tags are words or phrases which describe your pipeline and help organise it better. It helps categorize and organize data within the pipeline, making it easier to track and manage. **Enter the desired tag name. The entered tag name is saved to be re-used by other users.**
- **Description:** A description is a written representation that provides details, characteristics, or explanations about a pipeline. **It is a way of conveying information or understanding of particular pipeline. Provide the description that explain about your pipeline.**

3. Click **Save**. A welcome prompt message is displayed. This prompt is displayed by default for all the newly created pipelines. You can disable it from the [Prompt Mode settings](#).
4. Click **Got it** to finish Pipeline creation step.
5. You are redirected to the Pipeline configuration page.
6. Go back to the [Pipeline Configuration Page](#) to know how to add and configure Data Source to the pipeline.

Prompt Mode

After you either create new pipeline or edit the existing pipeline, an information wizard appears. If you don't want it to be displayed every time you open the canvas page, then you can disable it by **Prompt Mode** toggle switch button located the top right corner of the workspace.



Edit Pipeline

Last Updated on December 9, 2024

Created Pipelines are displayed in the Pipeline Repository screen. You can edit the pipeline name and its component.

Note:

- If you edit pipeline name, it is applied across all the versions.
- You cannot edit the pipeline in case the latest version exists and a lower version is active.
- To edit the pipeline make sure latest version is active. To activate latest version, refer to [Version History Common Actions](#).
- If you edit a pipeline which is in Running State, new pipeline version is created. For more information, refer to Pipeline in *Running State*.

To edit the pipeline follow below steps:

1. For the particular pipeline, Click Vertical Ellipsis > **Edit**.
2. Pipeline opens in edit mode. Now you can edit the pipeline name, pipeline components such as **Data Source**, **Transformation Operator**, and **Data Sink**.
3. **Edit Basic Configuration:** To edit pipeline basic configuration such as name, tag and description follow below steps:
 - a. Click edit icon beside Pipeline Name
 - b. **Edit Workflow** window is displayed.
 - c. Update the *fields*.
 - d. Click **Create New**.
 - e. Basic configuration details are updated.
4. **Edit Pipeline Component:** To edit pipeline components such as *Data Source*, *Transformation Operator*, and *Data Sink* follow below steps:
 - a. For particular pipeline component, Click Vertical Ellipsis > **Edit**.
 - b. Configuration page is displayed.
 - c. Update the fields.
 - d. Click **Save**.
5. Click **Save and View** button available on the top right corner of the screen.

Pipeline Actions

Run Pipeline

Last updated on May 19, 2025

After any pipeline is created, you must run the pipeline to generate the results of any configurations performed. To run the pipeline:

- **Configure the pipeline and click run .**
- **If the pipeline has run successfully, you are notified with the successful message else you are notified with the error message.**

Pipeline Canvas

Canvas is the page where you can configure the pipeline using Date Source, Transformation Operator, and Data Sink.

Below are the list of action icons available on the canvas:

1. **Open Operator:** It allows you to display the Operator Grid, if closed. By default, Operator grid is present on the canvas.
2. **Zoom-in:** It allows you to Zoom In the pipeline.
3. **Zoom-out:** It allows you to Zoom Out the pipeline.
4. **Auto Adjust:** It allows you to auto adjust the pipeline with single click.
5. **Setting:** It allows you to do the setting for [Cache](#), [String Rule](#), and [Custom JAR](#).
6. **Cancel:** It allows you to cancel the changes in pipeline.
7. **Save and View:** It allows you to save the changes in pipeline and view it.
8. **Run Results:** It allows you to upload a sample file and run parsing. When you execute, Run Results will get data in UI according to your pipeline flow configurations. For more information, refer to [Run Results](#).

Connect Two Nodes

To connect two nodes in a pipeline:

1. Click anywhere on the node1 (Example: Data Source - FTP) and drag till the node2 (Example: Transformation Operator - Parser), which you want to connect to.
2. You will see the green highlighted border around the node2, which means the node is connected.
3. The connected node is shown in blue line.

Edit/ Delete a Component in Pipeline

1. Click on the vertical ellipses associated with the node.
2. Click **Edit** to edit the node. The respective component's edit window is displayed.
3. Click **Delete** to delete the node.

Note:

To edit any node, first ensure to create connection between the nodes and then edit the required node. If not, edit operation is not possible.

Rename Component

Below are the steps to rename the component:

1. Using mouse, double click over the component name.
2. Component name is highlighted.
3. Use backspace or delete to remove the existing name.
4. Enter the required name of the component.

Note:

You can use special characters to rename the component.

Auto Adjust Node(s)

Select the node, and drag and drop into the canvas. Click on **Auto Adjust** icon. Nodes will get aligned automatically.

Note:

The node is said to be selected when it is highlighted with the green outline.

Run Results

Last updated on May 12, 2023

Run Results is available at the bottom toolbar in pipeline canvas. It allows you to upload a sample file and run parsing. When you execute, Run Results will get data in UI according to your pipeline flow configurations.

To run the pipeline:

1. Upload a sample file to validate the parsing of data on the GUI.
2. Click Run.
3. The validation process starts in background and a prompt is displayed at the bottom of the window, displaying "**You will be notified once pipeline run using sample data is completed for (pipeline name)**". After the validation is complete, a confirmation message will be displayed.
4. Go back to the [Pipeline Configuration Page](#) to know how about the visualization.

Pipeline Scheduler

Last updated on May 12, 2023

Pipeline Scheduler is available at the workspace of the pipeline repository. This option helps you to schedule the time interval of file collection for a pipeline.

Note:

Scheduler configuration is applied across all the pipeline versions.

1. To configure the frequency of the file collection, Click **Pipeline Scheduler** on any Pipeline on the **Pipeline Repository** list.
2. Pipeline Scheduler window is displayed.
3. To trigger the pipeline and pull the respective data sets at a specific interval, configure the following details:
 - a. **Scheduler Type**: Select the type from the drop-down:
 - b. **CRON Driven**: It allows you to configure the schedule in cron expression. Configure the following details:
 - i. **Minutes**: Allows you to enter the file collection time interval in minutes and seconds. For example, if you want system to collect file at every 5 minutes and 30 seconds, then configure as: Every 5 minute(s) on 30 SS.
 - ii. **Hourly**: Allows you to enter the file collection time interval in Hours, Minutes, and Seconds. For example, if you want system to collect file at every 2 hours, then configure as: Every 2 Hour(s) on 0 MM on 0 SS.
 - iii. **Daily**: Allows you to enter the file collection time interval in a day.
 - **Every**: Every nth day, where n can be up to 5. Example: if you have selected **Every** as 2, then on every alternate day, file collection will be triggered.
 - **Every Working day**: By default, Monday to Friday is considered as Working days in the system.
 - iv. **Weekly**: Allows you to enter the file collection time interval and define the trigger day(s) of the week. For example, if you want system to collect file on every Monday at every 11 pm, then configure as: Select **Monday** checkbox, at 23 HH on 0 MM on 0 SS.
 - v. **Monthly**: Allows you to enter the file collection time interval in week or day of a particular month. For example, If you want to set the collection time as 3rd March at 22:00:00 or on the first working day of March, then configure as following:
 - On the 3rd day of every 3 Month at 22 HH 00 MM 00 SS
 - On the First Monday of 3 Month at 22 HH 00 MM 00 SS
 - vi. **Yearly**: Allows you to enter the file collection time interval in a year on a particular month. For example, If you want to set the collection time as 27th February at 23:59:59 or on the 3rd Sunday of February of every year then configure as following.

- Every **February** on the **27th day** at **23 HH 59 MM 59 SS**
 - On the **Third Sunday** of **February** at **23 HH 59 MM 59 SS**
- vii. **Scheduler expression:** Displays the time value in expression according to the configured settings.

Interface Priority

Interface priority helps you to prioritize and run the required pipeline faster by configuring **Priority**, **Back Pressure** and **Queue Memory** values in scheduler.

To run any pipeline, **Priority**, **Back pressure** and **Queue Memory** is required. To configure **Interface Priority** in Pipeline Scheduler:

1. Click **Pipeline Scheduler** on any **Pipeline** on the **Pipeline Repository** list.
2. **Pipeline Scheduler** window is displayed. By default **Priority** is set to 1, **Back Pressure** is set to 100 and **Queue Memory** is set to 1 GB.
3. From the **Scheduler Type** drop-down, select **Timer Driven**. The corresponding window is displayed.
4. Configure the following fields in pipeline scheduler window:
 - a. **Timer driven(Scheduler Type):** It allows you to define the schedule for a specific-time. Configure the following details:
 - b. **Priority:** The number of threads that should be concurrently scheduled for each processors. By default, **Priority** is set to 1. If you give priority value as one, one number of threads are allocated to the pipeline. The more number of threads you allocate to a pipeline, the more it perform the tasks faster. When you prioritize a pipeline with the priority value as 10, then maximum number of threads are allocated to run the current pipeline and comparatively the other pipelines work slower.
 - c. **Back Pressure:** The maximum number of objects that can be queued before back pressure is applied. By default, **Back Pressure** is set to 100.
 - d. **Queue Memory:** The maximum data size (in GB) of objects that can be queued before back pressure is applied. By default, **Queue Memory** is set to 1 GB.
 - e. **Back Pressure and Queue Memory** will cause the processor to stop being schedule to run until the queue clears out. This allows the system to avoid being over run with data and memory.
 - f. **Duration:** Enter the time for running the scheduler. For example, if the time is set as 3 sec, then the pipeline will run at every 3 sec.
 - g. **Unit(s):** Select the time unit from dropdown list.
 - h. **Scheduler expression:** Displays the time value in expression according to the configured settings.
5. After providing priority, Back Pressure and Queue Memory, if we click on done button in scheduler screen, it will update the given values in system postgres database in a pipeline table. Once

we deploy and run the pipeline, it will update the provided priority. Back pressure and Queue Memory to the required pipeline.

6. Based on the configured values, the prioritized pipelines run faster as per requirement.
7. Go back to the [Pipeline Configuration Page](#) to know how to deploy and run the pipeline.

Data Source

Data Source Overview

Last updated on December 16, 2024

After creating a Pipeline, the next step in the pipeline creation of Data Management Studio is to have **Data Source**. It allows you to poll and collect the files from the different data sources. A data source can take various forms depending on the specific ETL pipeline implementation. Below types are the supported Data source:

- **Relational databases** : Data can be extracted from database systems such as Oracle or Postgres. The ETL process connects to the database, executes queries, and retrieves the required data.
- **File System** : Data can be sourced from files in formats like CSV (comma-separated values), Excel spreadsheets, JSON (JavaScript Object Notation), XML (eXtensible Markup Language), or log files. The ETL pipeline reads the files and extracts the relevant information.
- **Cloud storage** : Data can be stored in cloud-based storage systems like [Amazon S3](#), [Google Cloud Storage](#), or [Azure Blob Storage](#) or [Hive](#). The ETL pipeline connects to these storage systems, accesses the data files, and performs the extraction.
- **Message queues**: Some ETL pipelines may source data from message queues such as Apache Kafka. The pipeline consumes messages from the queue and processes the data they contain.
- **Streaming platforms**: In real-time or near-real-time ETL scenarios, data sources can be streaming platforms like Apache Kafka. The pipeline continuously processes data as it arrives.
- **Remote File Storage**: [FTP](#) and [SFTP](#)
- **Cloud Storage**: [Hive](#), [S3](#), [GCS](#) and [Azure Blob Storage](#)
- **Message Brokers**: [Kafka](#)
- **SQL Database**: [RDBMS](#), MySQL, MSSQL, Oracle, Postgres
- **Email Source**: [Email](#)
- **File Sorting**: [File Sorting](#)
- **Local File Location**: [Local File](#)

Data Source Configuration

To configure Data Source:


1. In Canvas, expand the **Data Source**, drag and drop the selected Data Source.
2. Connect the Data Source and the Transformation operator.

3. Click vertical ellipsis > Edit.
4. Configuration window is displayed.
5. Configure the Data Source.
6. Go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to the pipeline.

ABS

Last updated on January 19, 2024

Azure Blob Storage offers a scalable and cost-effective solution for storing and managing unstructured data, such as images, videos, documents, backups, and logs. Azure Blob Storage Source is used to fetch the data from the Azure server.

 This page provides step-wise configuration about the Data Source. After you configure the Data Source, go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to your pipeline(s).

To configure Azure Blob Storage as Data Source, you can [Create New Connection](#) or use [Existing Connections](#) from the list.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Source**. Drag and drop **Azure Blob Storage** source in canvas.
2. Provide the output connection to **Transformation Operator**.
3. Click on **Vertical Ellipsis** > Edit.
4. **Configure: Azure Blob Storage** window is displayed. Click on **Create New Connection**.
5. **Connection Details** tab is displayed. You should enter the authentication details to establish connection.
6. Configure the **Connection Details** Tab:
 - a. In **General Details**, **Connection Name**: Enter the connection name for data source Azure.
 - b. In **Azure Connection Details**: Enter the **Account Name** where the Azure is available.
 - c. In **Account Key**: Enter the account key for azure connection.
 - d. **Test Connection**: Allows you to test the connection setup.
7. Click **Save Connection**. **Collection Details** tab is enabled.
8. Configure the following in the **Collection Details** tab:

- a. **Container Name:** Enter the Container Name.
- b. **Prefix:** Enter the prefix. This represents folder under container.
- c. **What should the system do with the duplicate files?** This feature helps you to check if there are any duplicate files exist in your input files. You can detect and process the duplicate files. It will consider the incoming file as duplicate file, only when the pipeline has two or more files with same file name and the same content. Otherwise, it will not mark the file as duplicate even though the pipeline has two files with the same file names.
 - i. **Detect:** Select **Detect** checkbox. This allows you to detect the duplicate files and store them in database. It does not allow the duplicate files to enter the parser.Alert is generated when duplicate file is detected. For more refer to [Alert Support](#).
 - ii. **Process:** Select **Process** checkbox. This allows you to process the duplicated files to data sink.

Note:

- **Process** check box is enabled only after selecting the **Detect** checkbox.
- **You can not perform Process** operation alone.

- d. **Cache Age for File Duplicate Check(Seconds):** Enter the value in seconds. The file duplicate check functionality is active only for the configured time(seconds) line that you provided in **Cache Age for File Duplicate Check** text box. Click **Next**. **File Integrity** tab is enabled. It is used to check the data integrity and data quality. For more information, refer to [File Integrity](#).
9. Click **Save** to keep the connection configuration.

File Integrity

File Integrity tab is used to check the data integrity and data quality.

In this tab, **File Sequence Check** is configured. **File Sequence Check** is the process of checking whether the files arrive in the configured time and also to check if all the files are available from a source. It detects the missing files based on the sequence number present in the file name. If a file is missing, an alert is generated for the missing sequence number. To know more about support, refer to [Alert Support](#).

Steps to configure the File Sequence Check as follows:

1. Click on **File Integrity** Tab.
2. Under File Sequence tab, Click on **Add File Sequence Check**.
3. **Enable File Sequence Check** is added.
4. Click on expand icon to complete details.
5. Complete the below details:

- a. **Enable File Sequence Check:** Select the checkbox to enable the file sequence check.
- b. **Integrity Name:** It is unique name to identify the file sequence check. Enter the Integrity Name. **For example: CDR_check, outgoing_roaming etc.**
- c. **Group Information:** In this section, complete the property details related to file name in the expression field. In expression field, first value denotes the start position and second value denotes the property length from the start position of individual property. Complete the below details:
 - i. **Group Name:** It is name of the file. It is mandatory field. Enter the start position, length from the start position of the Group Name.
 - ii. **Sequence:** It represents the sequence of the file. It can be decimal, octal, or hexadecimal value. It is mandatory field. Enter the start position, length from the start position of the Sequence. In case of variable sequence length, mention the highest sequence length. **For example: Sequence can start with 2 digit such as 01 and end with 4 such as 9999. Sequence length is 4.**
 - iii. **Date:** It represents on which date specific file was generated. It can be in any suitable combination of day, month and year. **For example: DDMMYYYY,YYYYMMDD,MMDDYYYY etc. Enter the start position, length from the start position of the Date.**
 - iv. **Time:** It represents timestamp when specific file was generated. It can be in any suitable combination of hour, minute, second. **For example: HHmmSS, mmHHSS etc. Enter the start position, length from the start position of the Time.**
 - v. **Format:** It represents the combination in which date and time are mentioned in the file-name. Enter the date and time format. **For example: DDMMYYYYHHmmSS etc.**
 - vi. **For example: Consider XYZ company CDR file for region 401, which is generated every hour. Company uses the standard file name as "XYZCDR401" as group name. Total there are 24 files at file collection location. Lets decode the Group Information of XYZC-DR401_001_25092023070000 with help of illustration:**

| Property | Expression |
|------------|----------------|
| Group Name | 1,9 |
| Sequence | 11,3 |
| Date | 15,8 |
| Time | 23,6 |
| Format | DDMMYYYYHHmmSS |

- d. **Sequence Information:** In this section, complete the property details related to the sequence number, and update the value. Complete the below details:
 - i. **Number Base:** Select the file sequence type. It can be decimal, octal, or hexadecimal.
 - ii. **Start Number:** Indicates the start number of sequence. Enter the value.

- iii. **End Number:** Indicates the end number of sequence. Enter the value.
- iv. **Reset Number:** Indicates at what point the reset should occur once the file sequence is completed. It should either be less than start number or more than end number.
- v. **Increment:** Indicates the files sequence increment size. **For example: Suppose first file sequence is 001, second file sequence is 005, third file sequence is 009. Then increment value will be 4, as file sequence increases by 4.**
- vi. **For example: Consider XYZ company CDR file for region 401, which is generated every hour. Company uses the standard file name as "XYZCDR401" as group name. Total there are 24 files at file collection location. Suppose first record is XYZCDR401_001_25092023000000, last record is XYZCDR401_024_25092023230000, and file sequence is increase by value 1. Then Sequence Information as follows:**

| Property | Value |
|--------------|-------------|
| Number Base | Decimal(10) |
| Start Number | 1 |
| End Number | 24 |
| Reset Number | 0 |
| Increment | 1 |

- e. **Sample File Preview:** In this section, you can enter the sample file name and preview the group information property. Complete the below details:
 - i. **Sample File:** Enter the sample file name to preview the group information. It is a mandatory field. Sample File must contain maximum sequence length.
 - ii. **Check Preview:** Click to view the breakdown of group information property. You must first complete the group information details to preview the breakdown.
- f. **Variable Length Sequence:** If checked, you can use variable length sequence. Sequence can start and end with any digits. **For example:** Sequence can start with 2 digit such as 01 and end with 4 such as 9999.
- g. **Fill in the look back information:** In this section, you have to provide the look back days to perform file sequence check. Complete the below details:
 - i. **Initial Run Look back(days):** Enter the look back days for the file sequence check.
 - ii. **Sequence Completion Time(min):** Enter the time in minutes to complete the file sequence check.
- h. **Grace Period to start File Integrity Check:** In this section, you have to provide the period before system generates file sequence missing alert. Period should always be after file sequence completion time. Complete the below details:
 - i. **Period Value:** It is the period after the file sequence check is completed and system waits before generating alert. Enter the Value. **For example:** File Sequence check completes at 10 am. Grace Period is 30 minutes. System will generate file sequence missing alert after

10.30 am. In case, missing file is placed in file collection location before 10.30 am, alert will not generate.

- ii. **Period:** Select the period from the dropdown list. It can be Days, Minutes, and Hours.

6. Click on **Save** to keep the configuration.

7. Similarly you can add / edit / delete multiple File Sequence Check. Once you save the configuration, run the pipeline. An Alert is auto triggered as per the configured Alert Template. This is explained in below section.

Alert Support

Alert support is provided to find out the duplicate files or the missing file sequences from the large collection of files stored at file collection location. You have to create an alert configuration which will generate alert notification for duplicate and missing files.

Example: Consider the XYZ company CDR file for region 401, which is generated every hour. The company uses the standard file name "XYZCDR401" as its group name. In total, there are 24 files at the file collection location. The first record is XYZCDR401_001_25092023000000, the last record is XYZCDR401_024_25092023230000, and the file sequence is increased by value 1. Also Now consider the second record, XYZCDR401_002_25092023010000, is file sequence 2 that is missing from the file collection location and there is a duplicate copy of the first record.

Steps to receive Alert Notification:

Below are the steps to receive the alert notification for duplicate files and missing file sequences

1. When you create new data source connection or use existing connection, make sure below check box is selected:
Detect Checkbox under **File Collection/Collection Details** Tab.
Enable File Sequence Check under **File Integrity > File Sequence**.
2. From common left panel, navigate through **Admin > Common Settings > Alerts Configuration** to create new alert for Duplicate Check and Missing File Sequence.
3. Alerts for "**Duplicate Check**" and "**Missing File Sequence Check**" are displayed under **Alerts Configuration**.
4. Click on bell icon to view the generated alerts.
5. Duplicate Check and Missing File Sequence alert notification is displayed. Unread messages has blue dot.
6. Click on individual alert notification to view the message. Alert message displays: **Studio Name** in which alert has occurred, **Alert Level** that indicates the severity of alert, **Component** that indicates source of alert where it has occurred, **Description** that contains the detailed alert message.
7. Duplicate Check Alert Message is displayed as below: Missing File Sequence Alert Message is displayed as below:

Existing Connections Re-usability

Steps to use existing connections as follows:

1. To use the existing connection, select any of them from **Recently Used** or **All Connection list**. Click **Next**.
2. Collection Details tab is displayed. Complete the details in tab and Click **Next**. **File Integrity** tab is displayed. Complete the details.
3. Click **Save** to save the connection.
4. Go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to the pipeline.

Email

Last updated on January 19, 2024

Email as data source refers to the collection of email having attachment serving as data source. Information available in the attachment is parsed with help of appropriate parser, and transformed data is stored in form of tables. This data source read the email from folder, select the subject from which the attachment to be read. It also read compressed files from email. This data source do not extract any information present in the email body, image, or video files. Multiple file collection and parallel email processing is possible.



This page provides step-wise configuration about the Data Source. After you configure the Data Source, go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to your pipeline(s).

To configure Email as Data Source, you can [Create New Connection](#) or use the [Existing Connection](#). Also, you can [Edit](#) the existing connection.

Create New Connection

Steps to create new connections as follows:

1. In Canvas, expand **Data Source**. Drag and drop **Email source** in canvas.
2. Provide the output connection to **Transformation Operator**.
3. Click on **Vertical Ellipsis > Edit**.
4. **Configure: Email** window is displayed. Click on **Create New Connection**.
5. **Connection Details Tab** is displayed. This tab has details regarding host, authentication details such as server, port, user name, password to fetch emails.

Complete the below details:

- a. **Connection Name:** Connection name is unique name which helps to identify and distinguish from other connection. Enter the connection name for Email data source. Example: gmail_pop

- b. **Tags:** Tags are words or phrases which describes your email data source. **Enter the desired tag name.** The entered tag name is saved to be re-used by other users.
- c. **Account Type:** Account type refers to type of protocol that allow you to download messages. Available options are: POP3, IMAP. You should enable the "POP3" and "IMAP" services of your email account.
- d. **Host Name:** It is unique address assigned to each server connected to computer network that can receive or send data. **Enter the IP or host details of the machine where database resides. For example: Account type is POP3 and want to access email from Gmail account,** then host name will be pop.gmail.com. Similarly for Account type IMAP, host name will be imap.gmail.com. For outlook account, host name will be outlook.office365.com.

Note:

- AD integration is not supported.
- While processing emails, source will fetch all the mails from the specified email folder.
- One pipeline to be configured for one email account.

- e. **Port:** It is logical construct that allows multiple services to use same network interface on device. **Enter the host server port number.** The port number should be configured appropriately based on the type of account and the selected protocol (IMAP or POP).
 - f. **User Name:** It is unique name used to distinguish an individual user from others. Enter the Email ID to fetch the attachment from email source. **For example:** abcx@gmail.com.
 - g. **Password:** It is secret combination of characters that is used to authenticate and verify the identity of user. **Enter the password of your email account. For Gmail accounts, an App Password must be generated and used instead of the regular password. For more information on how to create App password, refer to [Sign in with app passwords](#).**
 - h. **Connection Timeout(Seconds):** It is the time (in seconds) required to establish the connection between your system and email server. Enter the Connection Timeout. **For example: Suppose Connection Timeout is 30 seconds. System will try for 30 seconds to establish connection with your email account. In case connection do not establish in 30 seconds, try again.**
 - i. **Test Connection:** Click to validate whether connection is successful. If connection is unsuccessful, check connection details.
 - j. Click **Save Connection** to keep Connection Details configuration.
6. Next Click on **Email Collection** Tab to configure. In this tab you have to provide email connection and file collection details to fetch the attachment from email.

Complete the below details:

- a. **Email to be Fetched From:** Enter the specific folder in the email account from where emails should be fetched. Currently "**Inbox**" Folder is supported. **For example:** You want emails to be pulled from Inbox folder, mention as Inbox.

- b. **New File Collection:** Click to Add File Collection. Multiple email file collection can be added for different subjects.

Complete the below details:

- i. **Email Collection Name:** Enter the email collection name.
 - ii. **Mail Subject Contains:** Emails are fetched on the basis of subject heading. Enter the Subject heading. This field is case sensitive and works regex match. Regex Match means system will look for specific mail subject, if subject is found email will be processed else email is not processed.
 - iii. **File Extension:** Enter the file extension to search the email attachment. For example: csv, txt, xls etc.
 - iv. **Compression Type:** This field is selected when the email has compressed file as an attachment. When compression type is selected, file extension field gets disabled. Select the appropriate compression type from the dropdown list. Available options are: None, GZIP, SNAPPY, TAR, ZIP.
 - v. **Save:** Click on Save icon to keep the file collection.
- c. You can add multiple file collection. Also you can edit and delete the existing file collection.
 - **Edit:** Click on **Edit** icon > Update fields > Click **Save** to modify the file collection.
 - **Delete:** Click on **Delete** icon > Click **Yes** on confirmation prompt to delete the file collection.
 - d. Click on **Save** to keep the Email Collection.

Existing Connections Re-usability

Steps to use existing connections as follows:

1. In **Connection Details Tab**, Select **Connection** either from **Recently Created** or **All Connections**.
2. Click **Next**.
3. **Email Collection** Tab is displayed. Complete the email collection details.
4. Click **Save** to use the existing configuration.

Edit Existing Connection

Steps to edit existing connection as follows:

1. To edit any connection's detail, Click on **Edit** icon.
2. Connection Details tab is displayed. Modify the required details and Test Connection.
3. Click **Save Connection** to keep the changes.

4. Go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to the pipeline.

File Sorting

Last updated on September 6, 2024

File Sorting is Data Source used to sort the incoming CDR based on the Timestamp to handle out of sequence records. It uses two pipelines. First Pipeline is used to fetch the incoming CDR and store the data in Local File Sink. Second Pipeline is used to fetch the records stored in Local File sink, apply sorting to out of sequence records and use ARC to calculate the benefit. Calculated result is stored in sink. Data table result can be viewed in the Data Catalog.

File Sorting data source is configured for pipelines that has sink as Local File Sink. Record is fetched from local directory, sorting is applied for out of sequence records and rate benefit is calculated using ARC operator.

File Sorting Pre-requisite

Below are the Pre-requisites for both the pipelines:

1. **First Pipeline: It is created with help of Data Source, Transformation Operator (Parser / Mapping), and Local File Sink. Pre-requisites as follows:**
 - a. While configuring **Parser**, Datetime Column on which record is sorted should be in date time format. Rest all other datetime / timestamp column should be converted to String Datatype. For more information, see [Record Structure Definition](#).
 - b. For **ASCII Parser**, Mapping should be added. While configuring Mapping, uncheck Record Address and Record Length. To configure Mapping, see [Mapping](#). For more information on ASCII Parser, see [ASCII Parser](#).
 - c. For **CSV Parser**, Mapping is optional. For more information on CSV Parser, see [CSV Parser](#).
2. **Second Pipeline: It is created with help of File Sorting, Transformation Operator (Parser/ Mapping), ARC and Sink. Pre-requisites as follows:**
 - a. While configuring **Parser**, set datetime format of sorting column as "yyyy-MM-dd HH:mm" format. Rest all other datetime / timestamp column should be converted to String Datatype. For more information, see [Record Structure Definition](#).
 - b. **Scheduler supports CRON driven. Minimum CRON time should be equal or greater than 1 minutes. Timer Driven Schedule is not supported. For more information, see [Pipeline Scheduler](#).**



This page provides step-wise configuration about the Data Source. After you configure the Data Source, go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to your pipeline(s).

Steps to configure File Sorting is listed as follows:


1. In Canvas, expand **Data Source**. Drag and drop **File Sorting** source in canvas.
2. Provide the output connection to **Transformation Operator**.

3. Click on **Vertical Ellipsis** > **Edit**.
4. **Configure: File Sorting** window is displayed.
5. Complete the below details:
 - a. **File Path:** Enter the Local File Sink path of the first pipeline to fetch the file. Example: /datas-tore/demo/CDR.
 - b. **Select Pipeline Name:** It displays the pipeline name which has sink as Local File Sink. Select pipeline from the dropdown list.
 - c. **Sink Name:** It displays the Local File Sink name available in the selected pipeline. Select the sink from the dropdown list.
 - d. **Choose Sorting Column:** It displays the column name on which sorting to be done. Select the sorting column from the dropdown list.
 - e. **Subscriber Grouping:** It displays the column names on which records to be grouped. Select the subscriber column from the dropdown list.
6. Click **Submit** to save the configuration.

FTP

Last updated on May 13, 2025

File Transfer Protocol (FTP) is a communication protocol providing for the transfer of files between remote devices and a server across a local (LAN) or wide-area network (WAN) like the internet. FTP **allows user to connect to an FTP server**, browse directories, upload files from their local system to the server, and download files from the server to their local system.

 This page provides step-wise configuration about the Data Source. After you configure the Data Source, go back to the *Pipeline Configuration Page* to know how to add and configure **Transformation Operator** to your pipeline(s).

To configure FTP as Data Source, you can [Create New Connection](#) or use [Existing Connections](#) from the list. If required, you can [Edit](#) the existing connection.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Source**. Drag and drop **FTP source** in canvas.
2. Provide the output connection to **Transformation Operator**.
3. Click on **Vertical Ellipsis** > **Edit**.
4. **Configure: FTP** window is displayed. Click on **Create New Connection**.
5. **Connection Details** tab is displayed.

6. In **Connection Details**, you should enter the authentication details to establish connection. For more information, refer to [Connection Details](#).
7. Next is to configure **File Collection**. This tab allows you to configure from which path the files need to be collected and you can choose the type of files you need to collect in order to perform a required action.
8. In File Collection complete the below details:
 - a. **Polling Details:** Allows you to configure the polling configurations. For more information, refer to [Polling Details](#).
 - b. **Collection Details:** Allows you to specify the details on file collection. For more information, refer to [Collection Details](#).
 - c. **Post Collection Details:** Allows you to configure action to perform post collection. For more information, refer to [Post Collection Details](#).
9. Click **Save** to keep the connection configuration. Click **Next** to configure **File Integrity**. This tab is used to check the data integrity and data quality. For more information, refer to [File Integrity](#)

Connection Details

To configure connection details, complete the below details:

1. General Details

- a. **Connection Name:** Connection name is unique name which helps to identify and distinguish from other connection. Enter the FTP connection name. Example: FTP_Conn1.
- b. **Tags:** Tags are words or phrases which describe your FTP Data Source. Enter the desired tag name. The entered tag name is saved to be re-used by other users.

2. Connection Details

- a. **Host Name/IP Address:** It is unique address assigned to each device connected to computer network that can receive or send data. Enter the host name where the data has to be loaded.
- b. **User Name:** It is unique name used to distinguish an individual user from others. Enter the user name to connect to host machine.
- c. **Port:** It is logical construct that allows multiple services to use same network interface on device. Enter the port number where the host is running.
- d. **Password:** It is a secret combination of characters that is used to authenticate and verify the identity of user. Enter the password for the connection.
- e. **Server Timezone:** It refers to time zone setting of server. It represent specific time zone that server's clock based on. It is essential for accurate recording and managing timestamp, scheduling task, and ensuring consistent time-based operation across different system. Select the timezone from dropdown list.
- f. Click **Test Connection**. If the connection is successful, you are notified and can proceed.
- g. Click **Save Connection** to save the connection.

3. Next is to configure *File Collection*.

Polling Details

To configure polling details flow below steps:

1. Click Polling Details to expand the menu.
2. Complete the below details:
 - a. **How many files can be polled in parallel?:** Allows you to configure the number of tasks that should be concurrently scheduled for this connection.
 - b. **Select stability seconds:** The minimum age that a file must be considered to be pulled. Any file less than this interval of time (last modification) will be ignored.
Example: Stability Seconds is set as 30 seconds, then the file's last modification time should be at 30 seconds older than current time. Therefore, it means that the file is fully written and can be considered for Polling.
 - c. **What batch size do you want to collect?:** Indicates the batch size for the file collection. Example: If the batch size is 100, 100 files are collected in one batch.
 - d. **Select local directory:** Select the path in the local machine where the files must be collected.
 - e. **If not collected how many retries are allowed?:** In few instances, files may not be collected. This parameter allows you to configure the number of attempts to be made to collect the files. Example: If the configured value is 5 and if any file is not collected on an attempt, the system tries for 10 times.
 - f. **How often do you want the system to retry?:** If the files are not collected, this parameter allows you to configure the time in seconds, after which the system tries to collect the files. Example: 5. If the value is 5 then system retries after every 5 seconds.
9. **What should the system do with the duplicate file?:** This feature helps you to check if there are any duplicate files exist in your source directory. You can detect and process the duplicate files. It will consider the incoming file as duplicate file, only when the pipeline has two or more files with same file name and the same content. Otherwise, it will not mark the file as duplicate even though the pipeline has two files with the same file names.
 - i. **Detect:** Select **Detect** checkbox. This allows you to detect the duplicate files and stores in system Postgres database. Once, a duplicate file is detected, a new table (duplicate_file) is created in the system Postgres database and the detected files will be stored in the duplicate_file table. It also adds duplicate files in local server directory. It does not allow the data to enter parser. Alert is generated when duplicate file is detected. For more refer to [Alert Support](#).
 - ii. **Process:** Select **Process** checkbox. This allows you to process the duplicated files to data sink and also stores in system Postgres database.

Note:

- **Process** check box is enabled only after selecting the **Detect** checkbox.
- You can perform **Detect** operation alone.

- h. **Cache Age for File Duplicate Check(Seconds):** Enter the value in seconds. The file duplicate check functionality is active only for the configured time(seconds) line that you provided in **Cache Age for File Duplicate Check** text box.

3. Click **Next** to complete Polling Details.

4. Next is to configure *Collection Details*.

Collection Details

To configure collection details flow below steps:

1. Click Collection Details to expand the menu.
2. Complete the below details:
 - a. **File List Mask:** Indicates the type of the file to be collected from any system. By default, it supports **.*** and it indicates all the file types will be collected. Example; **.txt** indicates only txt files will be processed.
 - b. **Collection Order:** Indicates the order in which the files must be collected. Example: If the selected order is **Oldest File First** then the oldest file will be selected to collect.
 - c. **Compression Type:** Select the compression type that the files are sent by the customers.

Note:

HyperSense supports the following compression types: GZip (.gz), Snappy, Tar (.tar), Zip (.zip), and Z (.z).

- d. **File Integrity Check:** This feature allows you to configure the minimum and maximum size of the file that is allowed to be processed by application. Example: If the minimum file size is 10 kb and maximum file is 20 kb, then application considers to process the files that fall between 10 - 20 kb only. There are two modes of check to perform:
 - i. **Manual:** If selected, the application considers minimum and maximum file size while processing. You should define minimum and maximum file size range before processing the file.
 - ii. **None:** If selected, the application doesn't consider the file size while processing.
- e. Configure the following for Manual File Integrity Check:
 - i. **Minimum File Size:** Enter the minimum file size in numbers and select the **Minimum File Size Unit** from dropdown list.
 - ii. **Maximum File Size:** Enter the maximum file size in numbers and select the **Maximum File Size Unit** from dropdown list.

Note:

- Select minimum and maximum file size unit from dropdown list. Available options are: Byte(B), Kilo Byte(KB), Mega Byte(MB), Giga Byte(GB).

3. Click **Next** to complete Collection Details.
4. Next is to configure [Post Collection Details](#).

Post Collection Details

To configure post collection details follow below steps:

1. Click Post Collection Details to expand the menu.
2. Complete below details:
 - a. **Post Collection Actions:** After the files are collected, you can either delete those files or move to the specified directory.
 - b. Click on **Next** to complete Post Collection Details.
3. Click **Save to keep the configuration changes**. **Next** to configure [File Integrity](#).

File Integrity

File Integrity tab is used to check the data integrity and data quality.

In this tab, File Sequence Check is configured. File Sequence Check is the process of checking whether the files arrive in the configured time and also to check if all the files are available from a source. **It detects the missing files based on the sequence number present in the file name. If a file is missing, an alert is generated for the missing sequence number.** To know more about support, refer to [Alert Support](#).

Steps to configure the File Sequence Check as follows:

1. Click on **File Integrity** Tab.
2. Under File Sequence tab, Click on **Add File Sequence Check**.
3. **Enable File Sequence Check** is added.
4. Click on expand icon to complete details.
5. Complete the below details:
 - a. **Enable File Sequence Check:** Select the checkbox to enable the file sequence check.
 - b. **Integrity Name:** It is unique name to identify the file sequence check. Enter the Integrity Name. **For example: CDR_check, outgoing_roaming etc.**

- c. **Group Information:** In this section, complete the property details related to file name in the expression field. In expression field, first value denotes the start position and second value denotes the property length from the start position of individual property. Complete the below details:
- Group Name:** It is name of the file. It is mandatory field. Enter the start position, length from the start position of the Group Name.
 - Sequence:** It represents the sequence of the file. It can be decimal, octal, or hexadecimal value. It is mandatory field. Enter the start position, length from the start position of the Sequence. In case of variable sequence length, mention the highest sequence length. For example: Sequence can start with 2 digit such as 01 and end with 4 such as 9999. Sequence length is 4.
 - Date:** It represents on which date specific file was generated. It can be in any suitable combination of day, month and year. **For example:** DDMMYYYY,YYYYMMDD,MMDDYYYY etc. Enter the start position, length from the start position of the Date.
 - Time:** It represents timestamp when specific file was generated. It can be in any suitable combination of hour, minute, second. **For example:** HHmmSS, mmHHSS etc. Enter the start position, length from the start position of the Time.
 - Format:** It represents the combination in which date and time are mentioned in the file-name. Enter the date and time format. **For example:** DDMMYYYYHHmmSS etc.
 - For example:** Consider XYZ company CDR file for region 401, which is generated every hour. Company uses the standard file name as "XYZCDR401" as group name. Total there are 24 files at file collection location. Lets decode the Group Information of **XYZC-DR401_001_25092023070000** with help of illustration:

| Property | Expression |
|------------|----------------|
| Group Name | 1,9 |
| Sequence | 11,3 |
| Date | 15,8 |
| Time | 23,6 |
| Format | DDMMYYYYHHmmSS |

- d. **Sequence Information:** In this section, complete the property details related to the sequence number, and update the value. Complete the below details:
- Number Base:** Select the file sequence type. It can be decimal, octal, or hexadecimal.
 - Start Number:** Indicates the start number of sequence. Enter the value.
 - End Number:** Indicates the end number of sequence. Enter the value.
 - Reset Number:** Indicates at what point the reset should occur once the file sequence is completed. It should either be less than start number or more than end number.

- v. **Increment:** Indicates the files sequence increment size. **For example:** Suppose first file sequence is 001, second file sequence is 005, third file sequence is 009. Then increment value will be 4, as file sequence increases by 4.
- vi. **For example:** Consider XYZ company CDR file for region 401, which is generated every hour. Company uses the standard file name as "XYZCDR401" as group name. Total there are 24 files at file collection location. Suppose first record is **XYZC-DR401_001_25092023000000**, last record is **XYZCDR401_024_25092023230000**, and file sequence is increase by value 1. Then Sequence Information as follows:

| Property | Value |
|--------------|-------------|
| Number Base | Decimal(10) |
| Start Number | 1 |
| End Number | 24 |
| Reset Number | 0 |
| Increment | 1 |

- e. **Sample File Preview:** In this section, you can enter the sample file name and preview the group information property. Complete the below details:
 - i. **Sample File:** Enter the sample file name to preview the group information. It is a mandatory field. Sample File must contain maximum sequence length.
 - ii. **Check Preview:** Click to view the breakdown of group information property. You must first complete the group information details to preview the breakdown.
- f. **Variable Length Sequence:** If checked, you can use variable length sequence. Sequence can start and end with any digits. **For example:** Sequence can start with 2 digit such as 01 and end with 4 such as 9999.
- g. **Fill in the look back information:** In this section, you have to provide the look back days to perform file sequence check. Complete the below details:
 - i. **Initial Run Look back(days):** Enter the look back days for the file sequence check.
 - ii. **Sequence Completion Time(min):** Enter the time in minutes to complete the file sequence check.
- h. **Grace Period to start File Integrity Check:** In this section, you have to provide the period before system generates file sequence missing alert. Period should always be after file sequence completion time. Complete the below details:
 - i. **Period Value:** It is the period after the file sequence check is completed and system waits before generating alert. Enter the Value. **For example:** File Sequence check completes at 10 am. Grace Period is 30 minutes. System will generate file sequence missing alert after 10.30 am. In case, missing file is placed in file collection location before 10.30 am, alert will not generate.

- ii. **Period:** Select the period from the dropdown list. It can be Days, Minutes, and Hours.

6. Click on **Save** to keep the configuration.

7. Similarly you can add / edit / delete multiple File Sequence Check. Once you save the configuration, run the pipeline. An Alert is auto triggered as per the configured Alert Template. This is explained in below section.

Alert Support

Alert support is provided to find out the duplicate files or the missing file sequences from the large collection of files stored at file collection location. You have to create an alert configuration which will generate alert notification for duplicate and missing files.

Example: Consider the XYZ company CDR file for region 401, which is generated every hour. The company uses the standard file name "XYZCDR401" as its group name. In total, there are 24 files at the file collection location. The first record is XYZCDR401_001_25092023000000, the last record is XYZCDR401_024_250920232300000, and the file sequence is increased by value 1. Also Now consider the second record, XYZCDR401_002_25092023010000, is file sequence 2 that is missing from the file collection location and there is a duplicate copy of the first record.

Steps to receive Alert Notification:

Below are the steps to receive the alert notification for duplicate files and missing file sequences

1. When you create new data source connection or use existing connection, make sure below check box is selected:
Detect Checkbox under **File Collection/Collection Details** Tab.
Enable File Sequence Check under **File Integrity > File Sequence**.
2. From common left panel, navigate through **Admin > Common Settings > Alerts Configuration** to create new alert for Duplicate Check and Missing File Sequence.
3. Alerts for "**Duplicate Check**" and "**Missing File Sequence Check**" are displayed under **Alerts Configuration**.
4. Click on bell icon to view the generated alerts.
5. Duplicate Check and Missing File Sequence alert notification is displayed. Unread messages has blue dot.
6. Click on individual alert notification to view the message. Alert message displays: **Studio Name** in which alert has occurred, **Alert Level** that indicates the severity of alert, **Component** that indicates source of alert where it has occurred, **Description** that contains the detailed alert message.
7. Duplicate Check Alert Message is displayed as below: Missing File Sequence Alert Message is displayed as below:

Existing Connections Re-usability

Steps to use existing connection as follows:

1. To use the existing connection, select any of the them from **Recently Used** or **All Connections** and click **Next**.
2. File collection tab is displayed. Complete the details in each tab. For more information refer to [File Collection](#).
3. Click **Save** to save the connection. Complete the File Integrity tab to find the missing file sequence.

Edit Connection

Steps to edit existing connection as follows:

1. To edit a connection's detail, click the **Edit** action against the required connection.
2. **Connection Details** tab is displayed. Modify the connection.
3. After modifying the details, Click **Save Connection** to save.

Go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to the pipeline.

GCS

Last updated on January 19, 2024

GCS stands for Google Cloud Storage. It is a scalable, durable, and highly available object storage service that offers high performance and reliability. It allows you to store and retrieve any amount of data in a secure, scalable, and highly available manner. GCS Source is used to fetch the data from the GCS server.



This page provides step-wise configuration about the Data Source. After you configure the Data Source, go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to your pipeline(s).

To configure GCS as Data Source, you can [Create New Connection](#) or use [Existing Connections](#) from the list.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Source**. Drag and drop GCS source in canvas.
2. Provide the output connection to **Transformation Operator**.

3. Click on **Vertical Ellipsis** > **Edit**.
4. **Configure: GCS** window is displayed. Click on **Create New Connection**.
5. **Connection Details** tab is displayed. You should enter the authentication details to establish connection.
6. Configure the **Connection Details** Tab:
 - a. **General Details**
 - i. **Connection Name:** Connection name is unique name which helps to identify and distinguish from other connection. Enter the connection name for data source GCS.
 - b. **GCS Connection Details**
 - i. **Project ID:** Enter the Project ID where the GCS is available.
 - ii. **Auth JSON file:** Displays the preview of the JSON file.
 - iii. **Upload:** Allows you to upload a json file to authenticate the GCS connection.
 - c. **Test Connection:** Allows you to test the connection setup.
7. Click **Save Connection**. **Collection Details** tab is enabled.
8. Configure the following in the **Collection Details** tab:
 - a. **Bucket Name:** Enter the bucket name to connect to the GCS.
 - b. **Prefix Name:** Enter the prefix for the connection.
 - c. **What should the system do with the duplicate files?** This feature helps you to check if there are any duplicate files exist in your source directory. You can detect and process the duplicate files. It will consider the incoming file as duplicate file, only when the pipeline has two or more files with same file name and the same content. Otherwise, it will not mark the file as duplicate even though the pipeline has two files with the same file names.
 - i. **Detect:** Select **Detect** checkbox. This allows you to detect the duplicate files and stores in system Postgres database. Once, a duplicate file is detected, a new table (duplicate_file) is created in the system Postgres database and the detected files will be stored in the duplicate_file table. It does not allow the data to enter parser. Alert is generated when duplicate file is detected. For more refer to [Alert Support](#).
 - ii. **Process:** Select **Process** checkbox. This allows you to process the duplicated files to data sink and also stores in system Postgres database.
 - d. **Cache Age for File Duplicate Check(Seconds):** Enter the value in seconds. The file duplicate check functionality is active only for the configured time(seconds) line that you provided in **Cache Age for File Duplicate Check** text box. Click **Next**. **File Integrity** tab is enabled. It is used to check the data integrity and data quality. For more information, refer to [File Integrity](#)
9. Click **Save** to keep the connection configuration.

File Integrity

File Integrity tab is used to check the data integrity and data quality.

In this tab, **File Sequence Check** is configured. **File Sequence Check** is the process of checking whether the files arrive in the configured time and also to check if all the files are available from a source. It detects the missing files based on the sequence number present in the file name. If a file is missing, an alert is generated for the missing sequence number. To know more about support, refer to [Alert Support](#).

Steps to configure the File Sequence Check as follows:

1. Click on **File Integrity** Tab.
2. Under File Sequence tab, Click on **Add File Sequence Check**.
3. **Enable File Sequence Check** is added.
4. Click on expand icon to complete details.
5. Complete the below details:
 - a. **Enable File Sequence Check:** Select the checkbox to enable the file sequence check.
 - b. **Integrity Name:** It is unique name to identify the file sequence check. Enter the Integrity Name. For example: CDR_check, outgoing_roaming etc.
 - c. **Group Information:** In this section, complete the property details related to file name in the expression field. In expression field, first value denotes the start position and second value denotes the property length from the start position of individual property. Complete the below details:
 - i. **Group Name:** It is name of the file. It is mandatory field. Enter the start position, length from the start position of the Group Name.
 - ii. **Sequence:** It represents the sequence of the file. It can be decimal, octal, or hexadecimal value. It is mandatory field. Enter the start position, length from the start position of the Sequence. In case of variable sequence length, mention the highest sequence length. For example: Sequence can start with 2 digit such as 01 and end with 4 such as 9999. Sequence length is 4.
 - iii. **Date:** It represents on which date specific file was generated. It can be in any suitable combination of day, month and year. For example: DDMMYYYY,YYYYMMDD,MMDDYYYY etc. Enter the start position, length from the start position of the Date.
 - iv. **Time:** It represents timestamp when specific file was generated. It can be in any suitable combination of hour, minute, second. For example: HHmmSS, mmHHSS etc. Enter the start position, length from the start position of the Time.
 - v. **Format:** It represents the combination in which date and time are mentioned in the file-name. Enter the date and time format. For example: DDMMYYYYHHmmSS etc.
 - vi. For example: Consider XYZ company CDR file for region 401, which is generated every hour. Company uses the standard file name as "XYZCDR401" as group name. Total there are 24 files at file collection location. Lets decode the Group Information of **XYZC-**

DR401_001_25092023070000 with help of illustration:

| Property | Expression |
|------------|----------------|
| Group Name | 1,9 |
| Sequence | 11,3 |
| Date | 15,8 |
| Time | 23,6 |
| Format | DDMMYYYYHHmmSS |

- d. **Sequence Information:** In this section, complete the property details related to the sequence number, and update the value. Complete the below details:
- Number Base:** Select the file sequence type. It can be decimal, octal, or hexadecimal.
 - Start Number:** Indicates the start number of sequence. Enter the value.
 - End Number:** Indicates the end number of sequence. Enter the value.
 - Reset Number:** Indicates at what point the reset should occur once the file sequence is completed. It should either be less than start number or more than end number.
 - Increment:** Indicates the files sequence increment size. **For example: Suppose first file sequence is 001, second file sequence is 005, third file sequence is 009. Then increment value will be 4, as file sequence increases by 4.**
 - For example: Consider XYZ company CDR file for region 401, which is generated every hour. Company uses the standard file name as "XYZCDR401" as group name. Total there are 24 files at file collection location. Suppose first record is XYZC-DR401_001_25092023000000, last record is XYZCDR401_024_25092023230000, and file sequence is increase by value 1. Then Sequence Information as follows:**

| Property | Value |
|--------------|-------------|
| Number Base | Decimal(10) |
| Start Number | 1 |
| End Number | 24 |
| Reset Number | 0 |
| Increment | 1 |

- e. **Sample File Preview:** In this section, you can enter the sample file name and preview the group information property. Complete the below details:

- i. **Sample File:** Enter the sample file name to preview the group information. It is a mandatory field. **Sample File** must contain maximum sequence length.
 - ii. **Check Preview:** Click to view the breakdown of group information property. You must first complete the group information details to preview the breakdown.
 - f. **Variable Length Sequence:** If checked, you can use variable length sequence. Sequence can start and end with any digits. **For example:** Sequence can start with 2 digit such as 01 and end with 4 such as 9999.
 - g. **Fill in the look back information:** In this section, you have to provide the look back days to perform file sequence check. Complete the below details:
 - i. **Initial Run Look back(days):** Enter the look back days for the file sequence check.
 - ii. **Sequence Completion Time(min):** Enter the time in minutes to complete the file sequence check.
 - h. **Grace Period to start File Integrity Check:** In this section, you have to provide the period before system generates file sequence missing alert. Period should always be after file sequence completion time. Complete the below details:
 - i. **Period Value:** It is the period after the file sequence check is completed and system waits before generating alert. Enter the Value. **For example:** File Sequence check completes at 10 am. Grace Period is 30 minutes. System will generate file sequence missing alert after 10.30 am. In case, missing file is placed in file collection location before 10.30 am, alert will not generate.
 - ii. **Period:** Select the period from the dropdown list. It can be Days, Minutes, and Hours.
6. Click on **Save** to keep the configuration.
7. Similarly you can add / edit / delete multiple File Sequence Check. Once you save the configuration, run the pipeline. An Alert is auto triggered as per the configured Alert Template. This is explained in below section.

Alert Support

Alert support is provided to find out the duplicate files or the missing file sequences from the large collection of files stored at file collection location. You have to create an alert configuration which will generate alert notification for duplicate and missing files.

Example: Consider the XYZ company CDR file for region 401, which is generated every hour. The company uses the standard file name "XYZCDR401" as its group name. In total, there are 24 files at the file collection location. The first record is XYZCDR401_001_25092023000000, the last record is XYZCDR401_024_25092023230000, and the file sequence is increased by value 1. Also Now consider the second record, XYZCDR401_002_25092023010000, is file sequence 2 that is missing from the file collection location and there is a duplicate copy of the first record.

Steps to receive Alert Notification:

Below are the steps to receive the alert notification for duplicate files and missing file sequences

1. When you create new data source connection or use existing connection, make sure below check box is selected:
Detect Checkbox under **File Collection/Collection Details** Tab.
Enable File Sequence Check under **File Integrity > File Sequence**.
2. From common left panel, navigate through **Admin > Common Settings > Alerts Configuration** to create new alert for Duplicate Check and Missing File Sequence.
3. Alerts for "**Duplicate Check**" and "**Missing File Sequence Check**" are displayed under **Alerts Configuration**.
4. Click on bell icon to view the generated alerts.
5. Duplicate Check and Missing File Sequence alert notification is displayed. Unread messages has blue dot.
6. Click on individual alert notification to view the message. Alert message displays: **Studio Name** in which alert has occurred, **Alert Level** that indicates the severity of alert, **Component** that indicates source of alert where it has occurred, **Description** that contains the detailed alert message.
7. Duplicate Check Alert Message is displayed as below: Missing File Sequence Alert Message is displayed as below:

Existing Connections Re-usability


Steps to use existing connections as follows:

1. To use the existing connection, select any of them from **Recently Used** or **All Connection list**. Click **Next**.
2. Collection Details tab is displayed. Complete the details in tab and Click **Next**. **File Integrity** tab is displayed. Complete the details.
3. Click **Save** to save the connection.
4. Go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to the pipeline.

Hive

Last updated on January 19, 2024

Hive Reader can read the data from the hive external tables created on top different object stores like AWS S3, Minio, GCS and Microsoft Azure. Hive reader connects to hive metastore and query all the databases and respective tables. Once user selects a table Hive readers look for the metadata of the table and reads the data into a data frame from respective object store location.

 This page provides step-wise configuration about the Data Source. After you configure the Data Source, go back to the *Pipeline Configuration Page* to know how to add and configure Transformation Operator to your pipeline(s).

To configure Hive as Data Source, you can [Create New Connection](#) or use [Existing Connections](#) from the list.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Source**. Drag and drop **Hive source** in canvas.
2. Provide the output connection to **Transformation Operator**.
3. Click on **Vertical Ellipsis > Edit**.
4. **Configure: Hive** window is displayed. Click on **Create New Connection**.
5. **Connection Details** Tab is displayed.
6. Complete the following details:
 - a. **General Details:**
 - i. **Connection Name:** Enter the connection name for data source hive. Example: Hive_Conn1
 - ii. **Tags:** Allows you to add a label to this connection for the purpose of identification or to give other information description.
 - iii. Click **Save**. Complete **Connection Details**.
 - b. **Connection Details:**
 - i. **Connection Details:** Enter the user name and password to connect to the host machine.
 - ii. **IP/Host:** Enter the IP or host details of the machine where hive resides.
 - iii. **Port:** Enter the host server port number.
 - iv. **Database Name:** Enter the database name of the Hive.
 - v. **Test Connection:** Click **Test Connection**. If the connection is successful, you are notified and can proceed.
 - vi. Click **Save**.
 - vii. Click **Done**. **Collection Details** tab is enabled.
7. In **Collection Details** Tab complete the following details:
 - a. **Thrift URL:** Enter the Thrift URL.
 - b. **HiveQL Query:** Enter the HiveQL Query to fetch data.
8. Click **Save** to save the connection.

9. Configured Example:

Existing Connections Re-usability

Steps to use existing connection as follows:

1. To use the existing connection, select any of the them from **Recently Used** or **All Connections** and Click **Next**.
2. Complete the **Collection Details** tab:
 - a. **Thrift URL**: Enter the Thrift URL.
 - b. **HiveQL Query**: Enter the HiveQL Query to fetch data.
3. Click **Save** to save the connection.
4. Go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to the pipeline.

Kafka

Last updated on January 19, 2024

The data available in table format in Kafka can be parsed. Before you parse, you must connect to the respective database, fetch the required data tables.

Example: If you have 10 columns in distributed event streaming platform kafka, which needs to be parsed, you can configure a data source to fetch those details and pass it to the parser. To configure Kafka as data source:



This page provides step-wise configuration about the Data Source. After you configure the Data Source, go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to your pipeline(s).

To configure Kafka as Data Source, you can [Create New Connection](#) or use [Existing Connections](#) from the list.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Source**. Drag and drop **Kafka** source in canvas.
2. Provide the output connection to **Transformation Operator**.
3. Click on **Vertical Ellipsis > Edit**.
4. **Configure: Kafka** window is displayed. Click on **Create New Connection**.
5. **Connection Details** tab is displayed. You should enter the authentication details to establish connection.

6. Configure the following details:

a. **General Details:**

- i. **Connection Name:** Enter the connection name for Kafka data source. Example: kafka_Conn1
- ii. **Tags:** Enter the tags that helps in identifying the purpose of the data source in a pipeline repository.
- iii. Click **Next**.

b. **Connection Details:**

- i. **Broker Urls:** Enter the url of the broker and press Enter key. You can add any number of broker URLs. Broker Urls should be in proper format, example: kafka:6080.
- ii. Click **Test Connection**. If the connection is successful, you are notified and can proceed.
- iii. Click **Save**.

7. Click **Done**. **Collection Details** tab is enabled.

8. In **Connection Details** complete the following:

- a. **Auto Offset Reset:** Select the offset reset policy. You can choose either to reset the position to the **Earliest** offset or the **Latest** offset (the default).
- b. **Maximum Poll Records:** Allows you to set the maximum number of records (by default 10000) returned in a single call to poll.
- c. **Number of threads:** Allows you to set the number of threads to execute stream processing.
- d. **Topics:** Allows you to enter virtual groups of one or many partitions across Kafka brokers. Input the Topic and press **Enter** key to add.

9. Click **Save** to keep the connection.

10. **Configured Example:**

Existing Connections Re-usability

Steps to use existing connections as follows:


1. To use the existing connection, select any of the them from **Recently Used** or **All Connections** and Click **Next**.
2. Complete the **Collection Details** tab:
 - a. **Auto Offset Reset:** Select the offset reset policy. You can choose either to reset the position to the **Earliest** offset or the **Latest** offset (the default).
 - b. **Maximum Poll Records:** Allows you to set the maximum number of records (by default 10000) returned in a single call to poll.
 - c. **Number of threads:** Allows you to set the number of threads to execute stream processing.
 - d. **Topics:** Allows you to enter virtual groups of one or many partitions across Kafka brokers. Input the Topic and press **Enter** key to add.

3. Click **Save** to save the connection.
4. Go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to the pipeline.

Local File

Last updated on December 16, 2024

Local File Source is a object provider service that offers picking large files from local file system i.e from the PVC (persistent volume claim) mounts and process it.

 This page provides step-wise configuration about the Data Source. After you configure the Data Source, go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to your pipeline(s).

Steps to configure the Local File Data Source:

1. In Canvas, expand **Data Source**. Drag and drop **Local File data source** in canvas.
2. Provide the input link to **Transformation Operator**. It can be connected to Parser or Flattener.
3. Click on **Vertical Ellipsis > Edit**.
4. **Configure: Local File** window is displayed.
5. Complete the below fields:
 - a. **Directory:** Enter the source directory or path to poll the source data file.
 - b. **File Format:** It defines the file format supported by the data source. Select the file format from the dropdown list. It supports CSV format.
 - c. **Post Processing Action:** It defines the action to be performed once the source file is polled. Select the action from the dropdown list. Available options are **None**, **Delete File**.
 - d. **Configured Example:**
6. Click **Save** to save the configuration.

RDBMS

Last updated on September 23, 2024

The data available in table format in RDBMS can be parsed. Before you parse, you must connect to the respective database, fetch the required data tables. RDBMS data source will help you in achieving this. All the connection configurations are based on the selected database type. **Example:** If you have selected MySQL, then you must configure the connections for MySQL database, where it resides.

i This page provides step-wise configuration about the Data Source. After you configure the Data Source, go back to the *Pipeline Configuration Page* to know how to add and configure Transformation Operator to your pipeline(s).

To configure RDBMS as data source, you can *Create New Connection* or use *Existing Connections* from the list.

Note:

- You can connect multiple RDBMS source nodes to a single parser at a time.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Source**. Drag and drop **RDBMS** source in canvas.
2. Provide the output connection to **Transformation Operator**.
3. Click on **Vertical Ellipsis > Edit**.
4. **Configure: RDBMS** window is displayed. Click on **Create New Connection**.
5. **Connection Details** tab is displayed. Complete the below details:
 - a. **General Details:**
 - i. **Connection Name:** Connection name is unique name which helps to identify and distinguish from other connection. Enter the connection name for RDBMS data source. Example: rdbms_Conn1
 - ii. **Tags:** Tags are words or phrases which describe your RDBMS Data Source. Enter the desired tag name. The entered tag name is saved to be re-used by other users.
 - b. **Connection Details:**
 - i. **User Name:** It is unique name used to distinguish an individual user from others. Enter the user name to connect to the host machine where the database resides.
 - ii. **Password:** It is a secret combination of characters that is used to authenticate and verify the identity of user. Enter the password to connect to the host machine where the database resides.
 - iii. **Database Type:** It refers to database category used to organize, store and manage data within DBMS. Select the database type from which the data is to be fetched.

Note:

- Hypersense supports an existing database: Postgres.
- Hypersense also supports new database: Oracle, MySQL, and MSSQL.
- Ensure you have installed the database to extract the data.

- iv. **IP/Host:** It is unique address assigned to each device connected to computer network that can receive or send data. Enter the IP or host details of the machine where database resides.
 - v. **Port:** It is logical construct that allows multiple services to use same network interface on device. Enter the host server port number.
 - vi. **Database Name:** It refers to name assigned to specific database used for managing and storing data. Enter the database name for the selected database type.
- c. Click **Test Connection**. If the connection is successful, you are notified and can proceed.
 - d. Click **Done**. **Collection Details** tab is enabled.
6. **Collection Details** tab is displayed. Complete the below details:
- a. Enter the SQL query to fetch the data from the table.
7. Click **Save**. Connection is created.

Existing Connections Re-usability

Steps to use existing connection as follows:

1. To use the existing connection, select any of the them from **Recently Used** or **All Connections** and Click **Next**.
2. In **Collection Details** tab, Complete the below details:
 - a. Enter the SQL query to fetch the data from the table.
3. Click **Save** to save the connection.
4. Go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to the pipeline.

S3

Last updated on January 19, 2024

S3 stands for Amazon Simple Storage Service, which is an object storage service that offers industry-leading scalability, data availability, security, and performance. Amazon S3 provides management features so that user can optimize, organize, and configure data to meet specific business, organizational, and compliance requirements.



This page provides step-wise configuration about the Data Source. After you configure the Data Source, go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to your pipeline(s).

To configure S3 as Data Source, you can [Create New Connection](#) or use [Existing Connections](#) from the list.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Source**. Drag and drop **S3 source** in canvas.
2. Provide the output connection to **Transformation Operator**.
3. Click on **Vertical Ellipsis > Edit**.
4. **Configure: S3** window is displayed. Click on **Create New Connection**.
5. **Connection Details** tab is displayed. You should enter the authentication details to establish connection.
6. In **Connection Details**, you should enter the authentication details to establish connection. For more information, refer to [Connection Details](#).
7. Next is to configure **Collection Details**. This tab allows you to configure from which path the files need to be collected and you can choose the type of files you need to collect in order to perform a required action. For more information, refer to [Collection Details](#).
8. Click **Save** to keep the connection configuration. Next is to configure **File Integrity**. This tab is used to check the data integrity and data quality. For more information, refer to [File Integrity](#)

Connection Details

To configure connection details, complete the below details:

1. **General Details**
 - a. **Connection Name:** Connection name is unique name which helps to identify and distinguish from other connection. Enter the connection name for S3 Data Source.
 - b. **Tags:** Tags are words or phrases which describe your S3 Data Source. Enter the desired tag name. The entered tag name is saved to be re-used by other users.
2. **Connection Details**
 - a. Enter the **Access Key**, **Secret Key**, and **End Point Url**.
 - b. **Select the region where you want to push data.**
 - c. Click **Test Connection** to verify the details.
3. Click Done. **Collection Details** tab is enabled.

Here is a configure Example:

4. Next is to configure [Collection Details](#).

Collection Details

1. Configure the following in the **Collection Details** tab:
 - a. **Bucket Name:** Enter the bucket name to connect to the S3.
 - b. **Prefix:** Enter the prefix for the connection.
2. **What should the system do with the duplicate file?** This feature helps you to check if there are any duplicate files exist in your source directory. You can detect and process the duplicate files. It will consider the incoming file as duplicate file, only when the pipeline has two or more files with same file name and the same content. Otherwise, it will not mark the file as duplicate even though the pipeline has two files with the same file names.
 - a. **Detect:** Select **Detect** checkbox. This allows you to detect the duplicate files and stores in system Postgres database. Once, a duplicate file is detected, a new table (duplicate_file) is created in the system Postgres database and the detected files will be stored in the duplicate_file table. It does not allow the data to enter parser. Alert is generated when duplicate file is detected. For more refer to [Alert Support](#).
 - b. **Process:** Select **Process** checkbox. This allows you to process the duplicated files to data sink and also stores in system Postgres database.

Note:

 - **Process** check box is enabled only after selecting the **Detect** checkbox.
 - You can perform **Detect** operation alone.
 - c. **Cache Age for File Duplicate Check(Seconds):** Enter the value in seconds. The file duplicate check functionality is active only for the configured time(seconds) line that you provided in **Cache Age for File Duplicate Check** textbox.
3. Click **Save** to keep the configuration. **File Integrity** tab is enabled. Click [File Integrity](#) to configure the same.

File Integrity

File Integrity tab is used to check the data integrity and data quality.

In this tab, **File Sequence Check** is configured. **File Sequence Check** is the process of checking whether the files arrive in the configured time and also to check if all the files are available from a source. It detects the missing files based on the sequence number present in the file name. If a file is missing, an alert is generated for the missing sequence number. To know more about support, refer to [Alert Support](#).

Steps to configure the File Sequence Check as follows:

1. Click on **File Integrity** Tab.
2. Under File Sequence tab, Click on **Add File Sequence Check**.
3. **Enable File Sequence Check** is added.
4. Click on expand icon to complete details.

5. Complete the below details:

- a. **Enable File Sequence Check:** Select the checkbox to enable the file sequence check.
- b. **Integrity Name:** It is unique name to identify the file sequence check. Enter the Integrity Name. **For example: CDR_check, outgoing_roaming etc.**
- c. **Group Information:** In this section, complete the property details related to file name in the expression field. In expression field, first value denotes the start position and second value denotes the property length from the start position of individual property. Complete the below details:
 - i. **Group Name:** It is name of the file. It is mandatory field. Enter the start position, length from the start position of the Group Name.
 - ii. **Sequence:** It represents the sequence of the file. It can be decimal, octal, or hexadecimal value. It is mandatory field. Enter the start position, length from the start position of the Sequence. In case of variable sequence length, mention the highest sequence length. **For example: Sequence can start with 2 digit such as 01 and end with 4 such as 9999. Sequence length is 4.**
 - iii. **Date:** It represents on which date specific file was generated. It can be in any suitable combination of day, month and year. **For example: DDMMYYYY,YYYYMMDD,MMDDYYYY etc. Enter the start position, length from the start position of the Date.**
 - iv. **Time:** It represents timestamp when specific file was generated. It can be in any suitable combination of hour, minute, second. **For example: HHmmSS, mmHHSS etc. Enter the start position, length from the start position of the Time.**
 - v. **Format:** It represents the combination in which date and time are mentioned in the file-name. Enter the date and time format. **For example: DDMMYYYYHHmmSS etc.**
 - vi. **For example: Consider XYZ company CDR file for region 401, which is generated every hour. Company uses the standard file name as "XYZCDR401" as group name. Total there are 24 files at file collection location. Lets decode the Group Information of **XYZC-DR401_001_25092023070000** with help of illustration:**

| Property | Expression |
|------------|----------------|
| Group Name | 1,9 |
| Sequence | 11,3 |
| Date | 15,8 |
| Time | 23,6 |
| Format | DDMMYYYYHHmmSS |

- d. **Sequence Information:** In this section, complete the property details related to the sequence number, and update the value. Complete the below details:

- i. **Number Base:** Select the file sequence type. It can be decimal, octal, or hexadecimal.
- ii. **Start Number:** Indicates the start number of sequence. Enter the value.
- iii. **End Number:** Indicates the end number of sequence. Enter the value.
- iv. **Reset Number:** Indicates at what point the reset should occur once the file sequence is completed. It should either be less than start number or more than end number.
- v. **Increment:** Indicates the files sequence increment size. **For example:** Suppose first file sequence is 001, second file sequence is 005, third file sequence is 009. Then increment value will be 4, as file sequence increases by 4.
- vi. **For example:** Consider XYZ company CDR file for region 401, which is generated every hour. Company uses the standard file name as "XYZCDR401" as group name. Total there are 24 files at file collection location. Suppose first record is **XYZC-DR401_001_25092023000000**, last record is **XYZCDR401_024_25092023230000**, and file sequence is increase by value 1. Then Sequence Information as follows:

| Property | Value |
|--------------|-------------|
| Number Base | Decimal(10) |
| Start Number | 1 |
| End Number | 24 |
| Reset Number | 0 |
| Increment | 1 |

- e. **Sample File Preview:** In this section, you can enter the sample file name and preview the group information property. Complete the below details:
 - i. **Sample File:** Enter the sample file name to preview the group information. It is a mandatory field. Sample File must contain maximum sequence length.
 - ii. **Check Preview:** Click to view the breakdown of group information property. You must first complete the group information details to preview the breakdown.
- f. **Variable Length Sequence:** If checked, you can use variable length sequence. Sequence can start and end with any digits. **For example:** Sequence can start with 2 digit such as 01 and end with 4 such as 9999.
- g. **Fill in the look back information:** In this section, you have to provide the look back days to perform file sequence check. Complete the below details:
 - i. **Initial Run Look back(days):** Enter the look back days for the file sequence check.
 - ii. **Sequence Completion Time(min):** Enter the time in minutes to complete the file sequence check.
- h. **Grace Period to start File Integrity Check:** In this section, you have to provide the period before system generates file sequence missing alert. Period should always be after file sequence completion time. Complete the below details:

- i. **Period Value:** It is the period after the file sequence check is completed and system waits before generating alert. Enter the Value. **For example:** File Sequence check completes at 10 am. Grace Period is 30 minutes. System will generate file sequence missing alert after 10.30 am. In case, missing file is placed in file collection location before 10.30 am, alert will not generate.
 - ii. **Period:** Select the period from the dropdown list. It can be Days, Minutes, and Hours.
6. Click on **Save** to keep the configuration.
 7. Similarly you can add / edit / delete multiple File Sequence Check. Once you save the configuration, run the pipeline. An Alert is auto triggered as per the configured Alert Template. This is explained in below section.

Alert Support

Alert support is provided to find out the duplicate files or the missing file sequences from the large collection of files stored at file collection location. You have to create an alert configuration which will generate alert notification for duplicate and missing files.

Example: Consider the XYZ company CDR file for region 401, which is generated every hour. The company uses the standard file name "XYZCDR401" as its group name. In total, there are 24 files at the file collection location. The first record is XYZCDR401_001_25092023000000, the last record is XYZCDR401_024_25092023230000, and the file sequence is increased by value 1. Also Now consider the second record, XYZCDR401_002_25092023010000, is file sequence 2 that is missing from the file collection location and there is a duplicate copy of the first record.

Steps to receive Alert Notification:

Below are the steps to receive the alert notification for duplicate files and missing file sequences

1. When you create new data source connection or use existing connection, make sure below check box is selected:
Detect Checkbox under **File Collection/Collection Details** Tab.
Enable File Sequence Check under **File Integrity > File Sequence**.
2. From common left panel, navigate through **Admin > Common Settings > Alerts Configuration** to create new alert for Duplicate Check and Missing File Sequence.
3. Alerts for "**Duplicate Check**" and "**Missing File Sequence Check**" are displayed under Alerts Configuration.
4. Click on bell icon to view the generated alerts.
5. Duplicate Check and Missing File Sequence alert notification is displayed. Unread messages has blue dot.
6. Click on individual alert notification to view the message. Alert message displays: **Studio Name** in which alert has occurred, **Alert Level** that indicates the severity of alert, **Component** that indicates source of alert where it has occurred, **Description** that contains the detailed alert message.

7. Duplicate Check Alert Message is displayed as below: Missing File Sequence Alert Message is displayed as below:

Existing Connections Re-usability

Steps to use existing connections as follows:

1. To use the existing connection, select any of the them from **Recently Used** or **All Connection list** and Click **Next**.
2. Collection Details tab is displayed. Complete the details in tab and Click **Next**.
3. Click **Save** to save the connection. File Integrity tab is displayed. Complete the details.
4. Go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to the pipeline.

SFTP

Last updated on January 19, 2024

The configuration of SFTP is similar to FTP, except the connection details. When you configure the connections for SFTP, you must provide the IP and port values associated with SFTP.

Note:

- You can connect multiple SFTP source nodes to a single parser at a time.

For more information on configurations, see [FTP Configurations](#).

Go back to the [Pipeline Configuration Page](#) to know how to add and configure Transformation Operator to the pipeline.

Transformation Operator

Transformation Operator Overview

Last updated on December 16, 2023

The next step involved in Data Management is data transformation/parsing. To transform any data to required format, you must configure the operators and provide the data source as an input to them. The output from Transformation Operators can be fed to the required module to use it or [store it in the database for further usage](#).

The operators are configured in Transformation Operator grid. You must select the required operator from the list and define the logic.

Following are the types of operators supported:

- [ARC](#)
- [Filter](#)
- [Flattener](#)
- [JSON Mapper](#)

- [LookUp](#)
- [Mapping](#)
- [Parser](#)
- [Preparser](#)
- [Record Processor](#)
- [Sampling Operator](#)
- [Subscriber Mapper](#)
- [LDC](#)

Transformation Operator Configuration

To configure the Transformation Operator:

1. Expand Transformation Operators. List of Operators are displayed.
2. Drag and drop the operator into the configuration grid.
3. Provide the Data Source input to the selected operator. Configure the operator and save the changes.
4. The output can further can be connected to Data Sink, to store it.

ARC

Last updated on [February 26, 2024](#)

ARC stands for **Advance Rating and Charging**. ARC processor processes the incoming CDR records using the Rating Processor and Charging Processor to calculate and charge the subscriber. Rating Processor will look for the subscriber's rate plan and tariff plan to calculate the initial rate. Rate is **charged when specific destination and time is mapped to specific tariff rate band**. Charging Processor will look for benefits associated with the subscriber's account. The final rate will be charged to the subscriber after reducing the subscriber's benefit. In case, subscriber does not have any benefit, then the initial calculated rate will be the final rate.

Example: Suppose there is an incoming CDR of 2 minutes and the subscriber has a monetary benefit of \$0.5. If the calculated rate is \$3, then the final rate charged to the subscriber after reducing the subscriber's benefit will be \$2.5.



This page provides step-wise configuration about the Transformation Operator. After you configure the Transformation Operator, go back to the [Pipeline Configuration Page](#) to know how to add and configure **Data Sink** to your pipeline.

Features Supported by ARC

Below are the list of features supported by ARC:

1. **Rating:** It is configured in `rate_type` column of `tariff_definition` table. ARC supports 4 types of rating explained as below:

a. *Flat Rating*

1. Rates are applied for a particular time band and configured as Rate/Pulse. For Example: 2Rs/10sec.
2. Rates are applied for a particular pulse size.
3. In the given example, pulse size is 10 for voice calls, and rate is Rs.2, which means for every 10 seconds, Rs.2 will be charged. Rates are configured in tariff_rate table where pulse_count is 1. Tables should be configured as follows:

a. tariff_definition table:

| id | name | description | unit | pulse_size | pulse_rounding | rate_type | rate_rounding | precision | set-up_fee | profile_id |
|----|------|-------------|----------|------------|----------------|-----------|---------------|-----------|------------|------------|
| 1 | td1 | rating | duration | 10 | UP | Flat | UP | 2 | 0 | 1 |

b. tariff_rate table:

| id | tariff_dfn_id | rate | pulse_count | profile_id |
|----|---------------|------|-------------|------------|
| 1 | 1 | 2 | 1 | 1 |

4. SMS and Data CDRs are always configured as Flat rating Type.

Note:

- Pulse Size for Voice, SMS and Data is configurable as per user requirement.
- For SMS scenario where usage is character length, pulse size will be 140 (standard character size of SMS).
- For SMS scenario where usage is SMS count, pulse size will be 1.
- For Data scenario, pulse size will be in bytes.

b. *Stepped Rating*

1. Rates are configured for multiple steps, that is for each consumed steps (duration) **seperate rates are applied. For example: 0-30 sec : 1Rs/10sec, 30-60 sec : 0.75Rs/10sec, 60-600 sec : 0.5Rs/10sec**
2. In rate_type column of tariff_definition table, it should be Stepped. In tariff_rate table, 3 configuration should be made for 3 different steps with different pulse count but same

tariff_definition id, since all the rates belongs to same tariff definition. Configuration as follows:

- a. for step 0-30sec : as pulse size is 10, pulse_count should be 3 since $3 \times 10 = 30$
- b. for step 30-60sec : as pulse size is 10, pulse_count should be 6 since $6 \times 10 = 60$
- c. for step 60-600sec : as pulse size is 10, pulse_count should be 60 since $60 \times 10 = 600$

3. Tables should be configured as follows:

a. **tariff_definition** table:

| id | name | description | unit | pulse_size | pulse_rounding | rate_type | rate_rounding | precision | set-up_fee | profile_id |
|----|------|-------------|----------|------------|----------------|-----------|---------------|-----------|------------|------------|
| 1 | td1 | rating | duration | 10 | UP | Stepped | UP | 2 | 0 | 1 |

b. **tariff_rate** table:

| id | tariff_dfn_id | rate | pulse_count | profile_id |
|----|---------------|------|-------------|------------|
| 1 | 1 | 1 | 3 | 1 |
| 2 | 1 | 0.75 | 6 | 1 |
| 3 | 1 | 0.5 | 60 | 1 |

c. *Tiered Rating*

1. Rates are configured for multiple tiers, that is rates are applied for a specific tier where the duration is falling. For example: 0-30 sec : 1Rs/10sec, 0-60 sec : 0.75Rs/10sec, 0-600 sec : 0.5Rs/10sec
2. In rate_type column of tariff_definition table, it should be Tiered. In tariff_rate table, 3 configuration should be made for 3 different tiers with different pulse count but same tariff_definition id, since all the rates belongs to same tariff definition. Configuration as follows:
 - a. for step 0-30sec : as pulse size is 10, pulse_count should be 3 since $3 \times 10 = 30$
 - b. for step 0-60sec : as pulse size is 10, pulse_count should be 6 since $6 \times 10 = 60$
 - c. for step 0-600sec : as pulse size is 10, pulse_count should be 60 since $60 \times 10 = 600$
 - d. Example: If chargeable call duration coming from CDR is 240sec, that means it falls under 3rd tier 0-600 which will be rated as Rs0.5/10sec.

3. Tables should be configured as follows:

a. tariff_definition table:

| id | name | description | unit | pulse_size | pulse_rounding | rate_type | rate_rounding | precision | set-up_fee | profile_id |
|----|------|-------------|----------|------------|----------------|-----------|---------------|-----------|------------|------------|
| 1 | td1 | rating | duration | 10 | UP | Tiered | UP | 2 | 0 | 1 |

b. tariff_rate table:

| id | tariff_dfn_id | rate | pulse_count | profile_id |
|----|---------------|------|-------------|------------|
| 1 | 1 | 1 | 3 | 1 |
| 2 | 1 | 0.75 | 6 | 1 |
| 3 | 1 | 0.5 | 60 | 1 |

d. *Telescopic Rating*

1. Rates are configured for multiple slabs, that is rates are applied for a specific slabs where the duration is falling. Rating is similar to Tiered rating type. For example: 0-30 sec : 1Rs/10sec, 30-60 sec : 0.75Rs/10sec, 60-600 sec : 0.5Rs/10sec
2. In rate_type column of tariff_definition table, it should be Telescopic. In tariff_rate table, 3 configuration should be made for 3 different slabs with different pulse count but same tariff_definition id, since all the rates belongs to same tariff definition.
3. If chargeable call duration coming from CDR is 240sec, that means it falls under 3rd tier 0-600 which will be rated as Rs0.5/10sec.
4. Tables should be configured as follows:

a. tariff_definition table:

| id | name | description | unit | pulse_size | pulse_rounding | rate_type | rate_rounding | precision | set-up_fee | profile_id |
|----|------|-------------|----------|------------|----------------|------------|---------------|-----------|------------|------------|
| 1 | td1 | rating | duration | 10 | UP | Telescopic | UP | 2 | 0 | 1 |

b. tariff_rate table:

| id | tariff_dfn_id | rate | pulse_count | profile_id |
|----|---------------|------|-------------|------------|
| 1 | 1 | 1 | 3 | 1 |
| 2 | 1 | 0.75 | 6 | 1 |
| 3 | 1 | 0.5 | 60 | 1 |

2. **Calls:** ARC supports 4 types of call explained as below:

- CUG:** It refers to Closed User Group calls. To mark call as CUG, isCUG flag should be "Y" in incoming CDR
- FNF:** It refers to Friends and Family calls. To mark call as FNF, isFNF flag should be "Y" in incoming CDR
- SPE:** It refers to Special calls. To mark call as SPE, isSPE flag should be "Y" in incoming CDR
- Normal Call:** It refers to ordinary CDR for Voice, SMS and Data. To mark call as normal or ordinary call, isCUG, isFNF and isSPE should be "N" in incoming CDR.

3. **Service:**ARC supports below list of services:

- Voice: Usage is marked as duration in seconds.
- SMS: Usage is marked as count and character length.
- Data:** Usage is marked as volume in bytes.

4. **Benefits:** Benefit could be monetary, non-monetary or discounts on final charge.

Note:

- Benefits should be first defined in benefit_definition table and then its id should be attached in dest_time table configuration.
- Benefits should be also configured in account_benefit table for the particular subscriber along with benefit_definition id and initial value of the benefit.
- If multiple benefits are applied, priority of applying benefit is decided by the arc_config_table , BENEFIT_ORDER column.
- If configured as EXPIRY_DATE, benefits with early expiry will be applied first.
- If configured as PRIORITY, benefits will be applied based on priority rank.

ARC supports below list of benefits:

- Monetary Benefits:** Its a charge benefit which is applied on the final calculated rate at the end of rating and charging. Benefit value is deducted from the final calculated charge. Monetary benefits can be applied to any type of CDR - Voice/SMS/Data.
 - Points to be considered during configuration*
 - Benefits will be valid as per the validity configured in benefit_definition table.

2. Type should be **CHARGE**.
3. Any currency unit can be given.
4. Table is configured as follows:

a. **benefit_definition** table

| id | type | priority | value | unit | valid_from | valid_to | profile_id | benefit_status |
|----|--------|----------|-------|------|-------------------------|-------------------------|------------|----------------|
| 1 | CHARGE | 1 | 2 | Rs | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 | 1 | Standard |

b. **account_benefit** table

| id | phone_number | benefit_definition_id | value | profile_id | valid_from | valid_to |
|----|---------------|-----------------------|-------|------------|-------------------------|-------------------------|
| 1 | +919433436992 | 1 | 2 | 1 | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 |

c. **dest_time** table

| id | name | destination_prefix | destination_code | time_band_id | tariff_dfn_id | benefit_dfn_id | profile_id | discount_ids |
|----|------|--------------------|------------------|--------------|---------------|----------------|------------|--------------|
| 1 | SMS | +9184939234 | 1 | 1 | 1 | 1 | 1 | 1 |

- b. **Usage Benefits:** It is non monetary benefit, applied on the total usage of the CDR. It is first deducted from the total usage, and then CDR is rated based on the chargeable usage. In benefit_definition table, type column should be usage.

i. *Table Configuration*

1. **benefit_definition** table

| id | type | priority | value | unit | valid_from | valid_to | profile_id | benefit_status |
|----|--------|----------|-------|---------|-------------------------|-------------------------|------------|----------------|
| 1 | US-AGE | 1 | 2 | minutes | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 | 1 | Standard |

2. **account_benefit** table

| id | phone_number | benefit_definition_id | value | profile_id | valid_from | valid_to |
|----|---------------|-----------------------|-------|------------|----------------------------|----------------------------|
| 1 | +919433436992 | 1 | 2 | 1 | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 |

3. dest_time table

| id | name | destination_prefix | destination_code | time_band_id | tariff_dfn_id | benefit_dfn_id | profile_id | discount_ids |
|----|------|--------------------|------------------|--------------|---------------|----------------|------------|--------------|
| 1 | SMS | +9184939234 | 1 | 1 | 1 | 1 | 1 | 1 |

ii. Below are the type of Usage benefit provided in ARC:

- Voice**

For Voice CDR, Usage benefits supports units as seconds, minutes, hours, days.

benefit_definition table configuration as follows:

| id | type | priority | value | unit | valid_from | valid_to | profile_id |
|----|-------|----------|-------|---------|----------------------------|----------------------------|------------|
| 1 | Usage | 1 | 2 | minutes | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 | 1 |

- SMS**

For SMS CDR, Usage benefits supports units as count.

benefit_definition table configuration as follows:

| id | type | priority | value | unit | valid_from | valid_to | profile_id |
|----|-------|----------|-------|-------|----------------------------|----------------------------|------------|
| 1 | Usage | 1 | 2 | count | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 | 1 |

- Data**

For Data CDR, Usage benefits supports units as bytes, kiloBytes, megaBytes, gigaBytes.

benefit_definition table configuration as follows:

| id | type | priority | value | unit | valid_from | valid_to | profile_id |
|----|-------|----------|-------|-------|----------------------------|----------------------------|------------|
| 1 | Usage | 1 | 2 | bytes | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 | 1 |

c. **Discount:** Its a type of benefit which is applied at the end of the rating and charging process on the final calculated charge. Its unit is always percentage. It is applied on all type of calls

(Voice/SMS/Data) on final charge. Validity of the discount depends on the validity dates configured in discount table.

i. *Points to be considered during configuration*

- Discount definition should be first configured in discount table. Then the discount id should be attached in dest_time table configuration. discount id should also be mapped in account_discount table with subscriber phone number.
- If 5 value is given in discount definition, that means 5% of discount should be applied on final calculated charge for that particular subscriber.
- Tables should be configured as follows:

a. **discount table:**

| id | priority | value | unit | valid_from | valid_to | profile_id |
|----|----------|-------|-------------|-------------------------|-------------------------|------------|
| 1 | 1 | 5 | percent-age | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 | 1 |

b. **account_discount**

| id | phone_number | discount_id | profile_id | valid_from | valid_to |
|----|---------------|-------------|------------|-------------------------|-------------------------|
| 1 | +919433436992 | 1 | 1 | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 |

c. **dest_time**

| id | name | destination_prefix | destination_code | time_band_id | tariff_dfn_id | benefit_dfn_id | profile_id | discount_id |
|----|------|--------------------|------------------|--------------|---------------|----------------|------------|-------------|
| 1 | SMS | +9184939234 | 1 | 1 | 1 | 1 | 1 | 1 |

Steps to deploy ARC

Following steps should be performed to configure and deploy ARC:

- Place all the data in backend postgres tables in dmw database. The following tables are available without any data.

- profile_tbl*

This table consists of profile configurations.

Below is the table details:

| Column | Description |
|----------------------|--|
| id(int) | It is a primary key of profile table. |
| profile_name(string) | It denotes name of the profile under which a set of configurations can be made. |
| version(string) | It denotes the version of the current profile. Version gets incremented by 1 on importing configuration sheet from UI. |
| ref_code(string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC. |
| description(string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC. |

Example:

| id | profile_name | version | ref_code | description |
|----|--------------|---------|-------------|-------------|
| 1 | Profile1 | 1 | sample_code | sample |

- *arc_config_tbl*

This table is ARC reference table.

Below is the table details:

| Column | Description |
|----------------|--|
| name(string) | <p>Consists of 3 reference configurations:</p> <ul style="list-style-type: none"> • BENEFIT_ORDER: Order in which benefits will be applied in a rate plan • ARC_LOGS_ENABLED: It denotes whether the ARC trace logs will be enabled or not. • TIMEZONE: It denotes the timezone where ARC configurations are made. Start Time/End Time configured in time_band table will be based on this mentioned timezone. |
| values(string) | <p>Consists of 3 reference configurations:</p> <ul style="list-style-type: none"> • BENEFIT ORDER: It consist of below values: <ul style="list-style-type: none"> ◦ EXPIRY_DATE: Benefits are applied based on the expiry date where benefits with early expiry will be applied first. |

| | |
|---------------------|--|
| | <ul style="list-style-type: none"> ◦ PRIORITY: Benefits are applied based on the priority order mentioned in benefit_dfn table. • ARC_LOGS_ENABLED: It consists of below values: <ul style="list-style-type: none"> ◦ TRUE: ARC trace logs will be enabled for all the cdr records. ◦ FALSE: ARC trace logs will be disabled for all the cdr records. • TIMEZONE: Timezone name should be provided here, like Asia/Kolkata. |
| description(string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |

Example:

| name | value | description |
|------------------|--------------|-------------|
| BENEFIT_ORDER | EXPIRY_DATE | sample |
| ARC_LOGS_ENABLED | TRUE | sample |
| TIMEZONE | Asia/Kolkata | sample |

• *subscriber*

This table consists of all the subscriber information including phone number and rate plan. Below is the table details:

| Column | Description |
|------------------------|---|
| id(int) | It is the primary key of subscriber table. |
| name(string) | It stores the name of the subscriber. |
| phone_number(string) | It stores phone number of the subscriber. aNumber coming from the processed CDR record are first matched with this field in order to find the subscriber. |
| rate_planid(int) | It stores id of the rate plan that belongs to the subscriber. |
| active_status(boolean) | <p>It stores the active status of subscriber. It can be either True or False.</p> <ul style="list-style-type: none"> • If true, then it denotes that the subscriber is in active state. • If false, it denotes the subscriber is in inactive state. Rating or charging for such subscribers are disabled. |

| | |
|----------------------------|---|
| profile_id (int) | It denotes the profile where the particular subscriber configuration belongs to. |
| ref_code (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |
| description (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |
| addon_benefit_ids (string) | Its a foreign key from benefit_dfn table. Benefit definitions whose benefit_status is Add On are mapped with a specific subscriber through this field. Add On benefits are specific to subscriber and not attached to a rate plan. Its a comma seperated field where multiple benefit_ids can be mapped. |

Example:

| id | name | phone_number | rate_plan_id | active_status | profile_id | ref_code | de- scrip- tion | addon_benefit_ids |
|----|-------|---------------|--------------|---------------|------------|----------|-----------------------|-------------------|
| 1 | User1 | +919433436992 | 1 | Yes | 1 | code1 | sample | 3,4 |
| 2 | User2 | +919433436994 | 2 | No | 1 | code2 | sample | |
| 2 | User3 | +911234567890 | 3 | Yes | 1 | code3 | sample | |

- *rate_plan*

This table consists of rate plan name and id.

Below is the table details:

| Column | Description |
|------------------|---|
| id(int) | It is a primary key of rate_plan table. |
| name(string) | It stores rate plan name. |
| profile_id(int) | It denotes the profile where the particular rate_plan belongs to. |
| ref_code(string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |

| | |
|---------------------|---|
| description(string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |
|---------------------|---|

Example:

| id | name | profile_id | ref_code | description |
|----|----------|------------|-------------|-------------|
| 1 | Profile1 | 1 | sample_code | sample |
| 2 | Profile2 | 2 | sample_code | sample |

- *rate_plan_detail*

This table consists of all the information regarding a rate plan.

Below is the table detail:

| Column | Description |
|----------------------------|---|
| id (int) | It is primary key of rate_plan_details table. |
| service_type | It denotes the service type of a rate plan. It can have different services like Voice, SMS, Data. Whatever service type is configured in the rate plan details, gets matched to the service type of coming CDR. |
| record_type | It denotes the record type of a rate plan. It can have different record types like O for Outgoing calls and I for incoming calls. Whatever record type is configured in the rate plan details, gets matched to the record type of coming CDR. |
| call_type | It denotes the call type of a rate plan. It can have different call types like Local, International, Roaming. Whatever call type is configured in the rate plan details, gets matched to the call type of coming CDR. |
| rate_plan_id (int) | Its a foreign key from rate_plan table. A single rate plan can have multiple rate plan details based on different service type, call type and record type. |
| tariff_card_id (int) | Its a foreign key from tariff_card table. A rate plan detail configuration should be associated with a tariff card. |
| profile_id (int) | It denotes the profile where the particular rate_plan_detail belongs to. |
| is_instantaneous (boolean) | It denotes whether the rate plan is for event based calls or not. It should be true for event based rate plans where service_type is |

| | |
|--------------------------------|--|
| | SMS and Data. For rate plans with service type as Voice, it should be false. |
| ref_field_usage (string) | Its a reference field that represents usage of the rate plan. For rate plans with Service type as Voice, it should be "duration". For rate plans with Service type as SMS, it can be either "character-Length" - when usage in the CDR is coming as character length, or it can be "count" - when usage in the CDR is coming ascount of SMS. For rate plans with Service type as Data, it should be "usage" |
| ref_field_destination (string) | Its a reference field that represents the destination field to be considered for rating a CDR. It can be either bNumber or apn. For Voice and SMS, it should be bNumber and for Data it should be apn. |
| ref_code (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |
| description (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |

Example:

| id | service_type | record_type | call_type | rate_plan_id | tariff_card_id | profile_id |
|----|--------------|-------------|-----------|--------------|----------------|------------|
| 1 | SMS | O | Local | 1 | 1 | 1 |
| 2 | Data-Roam | | | 2 | 2 | 1 |
| 3 | Voice | O | Local | 3 | 3 | 1 |

| id | is_instantaneous | ref_field_usage | ref_field_destination | ref_code | description |
|----|------------------|-----------------|-----------------------|-------------|-------------|
| 1 | Y | count | bNumber | sample_code | sample |
| 2 | Y | usage | apn | sample_code | sample |
| 3 | N | duration | bNumber | sample_code | sample |

- *time_band*

This table consists of all the time band configurations.

Below is the table details:

| Column | Description |
|----------------------------|---|
| id (int) | It is primary key of time_band table. |
| name (string) | It denotes name of time band. |
| time_band_start (datetime) | It is staring time of a time band. date part is not considered for time band only time part is extracted. |
| time_band_end (datetime) | It is ending time of a time band, date part is not considered for time band only time part is extracted. |
| profile_id (int) | It denotes the profile where the particular time_band configuration belongs to. |
| days (string) | It denotes the days of the week that time_band belongs to. Its a comma seperated field, where multiple days can be configured. 0 represents Sunday 1 represents Monday 2 represents Tuesday 3 represnts Wednesday 4 represents Thursday 5 represents Friday 6 represents Saturday |
| ref_code (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |
| description (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |

Example:

| id | name | time_band_start | time_band_end | profile_id | days | ref_code | de- scrip- tion |
|----|------------------|------------------------|------------------------|------------|---------------|--------------|-----------------------|
| 1 | Stepped-Peak | 2023-09-01 00:00:00 | 2023-12-30 15:59:59 | 1 | 0,1,2,3,4,5,6 | sample__code | sample |
| 2 | Stepped-Off Peak | 2023-09-01 16:00:00 | 2023-12-30 23:59:59 | 1 | 0,1,2,3,4,5,6 | sample_code | sample |

- *dest_time*

This is mapping table where specific destinations (bNumber or apn) are mapped with time band, tariff dfn, benefit dfn and discount.

Below is the table details:

| Column | Description |
|-----------------------------|--|
| id (int) | It is primary key id of dest_time table |
| name (string) | It stores name of the destination_time configuration |
| destination_prefix (string) | It denotes the destination string, it can be either bNumber or apn based on the type of call. Destinations are matched with the bNumber or called number from the incoming CDR. |
| destination_code (int) | All the similar destinations have same destination_code. It is used for longest prefix match of destination. |
| time_band_id (int) | Its a foreign key from time_band table. Time bands are mapped with a specific destination through this field. |
| tariff_dfn_id (int) | Its a foreign key from tariff_dfn table. Tariff definitions are mapped with a specific destination through this field. |
| benefit_dfn_id (string) | Its a foreign key from benefit_dfn table. Benefit definitions are mapped with a specific destination through this field. Its a comma seperated field where multiple benefit_ids can be mapped. |
| discount_ids (string) | Its a foreign key from discount table. Discounts are mapped with a specific destination through this field. Its a comma seperated field where multiple discount_ids can be mapped. |
| profile_id (int) | It denotes the profile where the particular dest_time configuration belongs to. |
| ref_code (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC. |
| description (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC. |

Example:

| id | name | destination_prefix | destination_code | time_band_id | tariff_dfn_id |
|----|------|--------------------|------------------|--------------|---------------|
| 1 | SMS | +9184939234 | 1 | 1 | 1 |

| | | | | | |
|---|-------|-------------|---|---|---|
| 2 | Data | three.co.uk | 2 | 2 | 2 |
| 3 | Voice | +9184939234 | 3 | 3 | 3 |

| id | benefit_dfn_id | profile_id | discount_ids | ref_code | description |
|----|----------------|------------|--------------|-------------|-------------|
| 1 | 1 | 1 | 1,2 | sample_code | sample |
| 2 | 1,2,3 | 1 | | sample_code | sample |
| 3 | | 1 | | sample_code | sample |

- *tariff_card*

This table consists of tariff card name and id. Its a reference field that connects a rate plan with a specific combination of destination, time and tariff definition.

Below is the table details:

| Column | Description |
|---------------------|---|
| id(int) | It is primary key of tariff_card table. |
| name(string) | It stores name of a tariff card. |
| profile_id(int) | It denotes the profile where the particular tariff_card belongs to. |
| ref_code(string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |
| description(string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |

Example:

| id | name | profile_id | ref_code | description |
|----|------|------------|-------------|-------------|
| 1 | tc1 | 1 | sample_code | sample |

- *tariff_card_dest_map*

Its a mapping table that maps a tariff card with dest_time table.

Below is the table details:

| Column | Description |
|----------------------|--|
| id(int) | It is primary key of tariff_card_dest_map table. |
| name(string) | It stores name of a tariff card destination map. |
| dest_time_id (int) | Its a foreign key from dest_time table. Multiple dest_time table configurations can be mapped to a tariff card. |
| tariff_card_id (int) | Its a foreign key from tariff_card table. |
| profile_id(int) | It denotes the profile where the particular tariff_card_dest_map configuration belongs to. |
| ref_code (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC. |
| description (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC. |

Example:

| id | name | dest_time_id | tariff_card_id | profile_id | ref_code | description |
|----|------|--------------|----------------|------------|-------------|-------------|
| 1 | tc1 | 1 | 1 | 1 | sample_code | sample |

- tariff_definition**

This table consists of all the tariff related information.

Below is the table details:

| Column | Description |
|----------------------|---|
| id(int) | It is Primary key id of tariff_definition table. |
| name (string) | It stores name of the Tariff. |
| description (string) | Description of the tariff definition |
| unit (string) | It denotes the unit of the defined pulse. Units of the usage coming in CDR are defined here. For Voice it iss seconds, for SMS it is characters, and for data it is bytes. |
| pulse_size (int) | It defines the size of the pulse |

| | |
|---------------------------|---|
| pulse_rounding (string) | It denotes pulse rounding configuration to calculate rate. It can be either UP or DOWN |
| rate_type (string) | It denotes the rating type of a tariff definition. Supported rating types are Flat, Tiered, Stepped and Telescopic. For SMS and DATA records, rating type is always Flat. |
| Precision (int) | It represents the decimal precision of final calculated charge. |
| setup_fee (float) | Its the optional setup amount to be added to the final charge. |
| profile_id (int) | It denotes the profile where the particular tariff_definition configuration belongs to. |
| ref_code (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |
| code_description (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |

Example:

| id | name | description | unit | pulse_size | pulse_rounding | rate_type |
|----|------|-------------|------------|------------|----------------|-----------|
| 1 | td1 | rating | characters | 1 | UP | Flat |
| 2 | td2 | rating | byte | 10240 | UP | Flat |
| 3 | td3 | rating | seconds | 10 | UP | Flat |

| id | rate_rounding | precision | setup_fee | profile_id | ref_code | code_description |
|----|---------------|-----------|-----------|------------|-------------|------------------|
| 1 | UP | 2 | 0 | 1 | sample_code | sample |
| 2 | UP | 2 | 0 | 1 | sample_code | sample |
| 3 | UP | 2 | 0 | 1 | sample_code | sample |

- tariff_rate**

It consists of tariff rate related information

Below is the table details:

| Column | Description |
|--------|-------------|
|--------|-------------|

| | |
|----------------------|--|
| id(int) | It is primary key of tariff_rate table. |
| tariff_dfn_id(int) | Its a foreign key from tariff_definition table. Its a mapping field that denotes the tariff rate belongs to which tariff definition. |
| rate(float) | It denotes the rate of the tariff |
| pulse_count(int) | It denotes the count of the pulse for which tariff rate has been given. Suppose the tariff rate is Rs.3/minute and pulse size in tariff_definition table is given as 10. Then pulse count for Rs.3 tariff rate will be 6. |
| profile_id(int) | It denotes the profile where the particular tariff_card_dest_map configuration belongs to. |
| ref_code (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC. |
| description (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC. |

Example:

| id | tariff_dfn_id | rate | pulse_count | profile_id | ref_code | description |
|----|---------------|----------|-------------|------------|-------------|-------------|
| 1 | 1 | 1.5 | 1 | 1 | sample_code | sample |
| 2 | 2 | 0.017578 | 1 | 1 | sample_code | sample |
| 3 | 3 | 1 | 1 | 1 | sample_code | sample |

- subscriber_account*

This table consists of total computed charge for each subscriber.

Below is the table details:

| Column | Description |
|-------------------------|---|
| id (int) | Primary key id of subscriber_account table. |
| phone_number (string) | Its denotes the phone number of all the subscribers |
| unbilled_charge (float) | It denotes the unbilled charge for all the subscribers. |

| | |
|-----------------------|---|
| billed_charge (float) | It denotes the billed charge for all the subscribers. Cureently it is not used for computation, so value will be 0. |
| profile_id (int) | It denotes the profile where the particular subscriber_account configuration belongs to. |
| ref_code (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |
| description (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |

Example:

| id | phone_number | unbilled_charge | billed_charge | profile_id | ref_code | description |
|----|---------------|-----------------|---------------|------------|-------------|-------------|
| 1 | +919433436992 | 2 | 0 | 1 | sample_code | sample |
| 2 | +919433436992 | 2 | 0 | 1 | sample_code | sample |
| 3 | +911234567890 | 2 | 0 | 1 | sample_code | sample |

- **account_benefit**

Its an account related table that keeps a record of subscriber benefit bucket.

Below is the table details:

| Column | Description |
|--------------------------------|--|
| id(int) | Primary key id of account_benefit table. |
| phone_number (string) | Phone muber of the subscribers who has rate plans with benefits |
| benefit_definition_id (int) | Its a foreign key from benefit_definition table. It denotes the benefit definiton that belongs to each subscriber. |
| value (float) | It denotes the current value of the benefit bucket for each subscriber. Initial value of the value field will be same as benefit defintion value. |
| profile_id (int) | It denotes the profile where the particular account_benefit configuration belongs to. |
| valid_from (time-stamp) | It denotes the start date of benefit validity for the given subscriber as timestamp date (yyyy-mM-dd hh:mm:ss). |

| | |
|----------------------|--|
| valid_to (timestamp) | It denotes the end date of benefit validity for the given subscriber as timestamp date (yyyy-mM-dd hh:mm:ss). |
| ref_code (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC. |
| description (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC. |

Example:

| id | phone_number | benefit_definition_id | value | profile_id | valid_from | valid_to | ref_code | description |
|----|---------------|-----------------------|-------|------------|-------------------------|-------------------------|-------------|-------------|
| 1 | +919433436991 | 3 | 2 | 1 | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 | sample_code | sample |

- benefit_definition**

It consists of benefit related information.

Below is the table definition:

| Column | Description |
|------------------------|---|
| id (int) | Primary key id of benefit_definition table. |
| type (string) | It denotes the type of the benefit. For non-monetary usage related benefits, it should be USAGE For monetary benefits, it should be CHARGE |
| priority (string) | It denotes the priority order(ranking) based on which benefits will be applied , starts from 1 and so on. Benenfit definitions with lower rank have highest priority, and in ascending order benefit priority decreases. |
| value (float) | It denotes the value of the benefit. |
| unit (string) | It denotes the measure unit of the benefit. For SMS, it should be count. For Voice usage benefits, it can be seconds, minutes, hour, days. For Data usage benefits, it can be bytes, kiloBytes, megaBytes, gigaBytes. |
| valid_from (timestamp) | It denotes the start date of benefit validity as timestamp date (yyyy-mM-dd hh:mm:ss) |

| | |
|--------------------------------|---|
| valid_to (time-stamp) | It denotes the end date of benefit validity as timestamp date (yyyy-mM-dd hh:mm:ss) |
| profile_id (int) | It denotes the profile where the particular benefit_definition configuration belongs to. |
| ref_code (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |
| description (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |
| benefit_status (string) | Benefit Status can be of 2 types: Standard : Benefit definition which are part of the rate plan are considered as Standard benefit Add On : Benefit definition which are taken by subscriber as an add on benefit, which is mapped to subscriber and they are not part of the rate plan |

Example:

| id | type | priority | value | unit | valid_from | valid_to | profile_id |
|----|--------|----------|-------|-----------|-------------------------|-------------------------|------------|
| 1 | US-AGE | 1 | 2 | count | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 | 1 |
| 2 | US-AGE | 1 | 25 | megaBytes | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 | 1 |
| 3 | US-AGE | 1 | 2 | minutes | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 | 1 |
| 4 | US-AGE | 2 | 5 | minutes | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 | 1 |
| 5 | US-AGE | 1 | 10 | Rs | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 | 1 |

| id | ref_code | description | benefit_status |
|----|-------------|-------------|----------------|
| 1 | sample_code | sample | Standard |
| 2 | sample_code | sample | Standard |

| | | | |
|---|-------------|--------|--------|
| 3 | sample_code | sample | Add On |
| 4 | sample_code | sample | Add On |
| 5 | sample_code | sample | Add On |

- *account_discount*

Its a mapping table that maps a dicount to a specific subscriber.

Below is the table details:

| Column | Description |
|-------------------------------------|---|
| id(int) | Primary key id of account_discount table. |
| phone_number (string) | Phone muber of the subscribers who has rate plans with discounts |
| discount_id (string) | Its a foreign key from discount table. It denotes the discount definition that belongs to each subscriber. |
| profile_id (int) | It denotes the profile where the particular account_discount configura- tion belongs to. |
| valid_from (time- stamp) | It denotes the start date of discount validity for the given subscriber as timestamp date (yyyy-mM-dd hh:mm:ss) |
| valid_to (time- stamp) | It denotes the end date of discount validity for the given subscriber as timestamp date (yyyy-mM-dd hh:mm:ss) |
| ref_code (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |
| description (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |

Example:

| id | phone_number | discount_id | profile_id | valid_from | valid_to | ref_code | de- scrip- tion |
|----|---------------|-------------|------------|----------------------------|----------------------------|-------------|-----------------------|
| 1 | +919433436992 | 1 | 1 | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 | sample_code | sam- ple |

- *discount*

It consists of discount related information.

Below is the table definition:

| Column | Description |
|------------------------|--|
| id(int) | Primary key id of discount table. |
| priority (string) | It denotes the priority order(ranking) based on which discounts will be applied, starts from 1 and so on. Discounts with lower rank have highest priority, and in ascending order discounts priority decreases. |
| value (float) | It denotes the discount percentage to be applied on the final calculated charge. |
| unit (string) | It denotes the measure unit of discount. It should be percentage. |
| valid_from (timestamp) | It denotes the start date of discount validity as timestamp date (yyyy-mM-dd hh:mm:ss) |
| valid_to (timestamp) | It denotes the end date of discount validity as timestamp date (yyyy-mM-dd hh:mm:ss) |
| profile_id (int) | It denotes the profile where the particular discount configuration belongs to. |
| ref_code (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |
| description (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |

Example:

| id | priority | value | unit | valid_from | valid_to | profile_id | ref_code | description |
|----|----------|-------|------------|-------------------------|-------------------------|------------|-------------|-------------|
| 1 | 1 | 5 | percentage | 2023-09-01 00:00:00.000 | 2023-12-30 00:00:00.000 | 1 | sample_code | sample |

- arc_profile_config*

Its a mapping table that maps the field names coming from the input CDR with ARC specific input field names.

Below is the table definition:

| Column | Description |
|------------------------------|---|
| id(int) | Primary key id of arc_profile_config table. |
| ref_fields (string) | <p>Its a mapping field where field names coming from the input CDR can be mapped with ARC specific input field names. In Parser , we need to configure input CDR field names. For example, in input CDR field name for aNumber is callingNumber, so we need to map it as aNumber=callingNumber, bNumber=calledNumber, serviceType=servicingType, callType=callingType, startTime=startingTime, count=smsCount, usage=dataUsage, apn=dataApn, isCUG=isCUG, isFNF=isFNF, isSPE=isSPE, duration=vDuration, recordType=recordoType, characterLength=scharacterLength, destinationCode=destinationCode</p> <p>Note: dsitinationCode field name coming from lpm name in Mapping operator, should also be mapped here.</p> |
| profile_id (int) | It denotes the profile where the particular arc_profile_config configuration belongs to. |
| ref_code (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |
| descrip- tion (string) | Its an alphanumeric field, an extra column used to input any data for the given configuration. Not used internally by ARC |

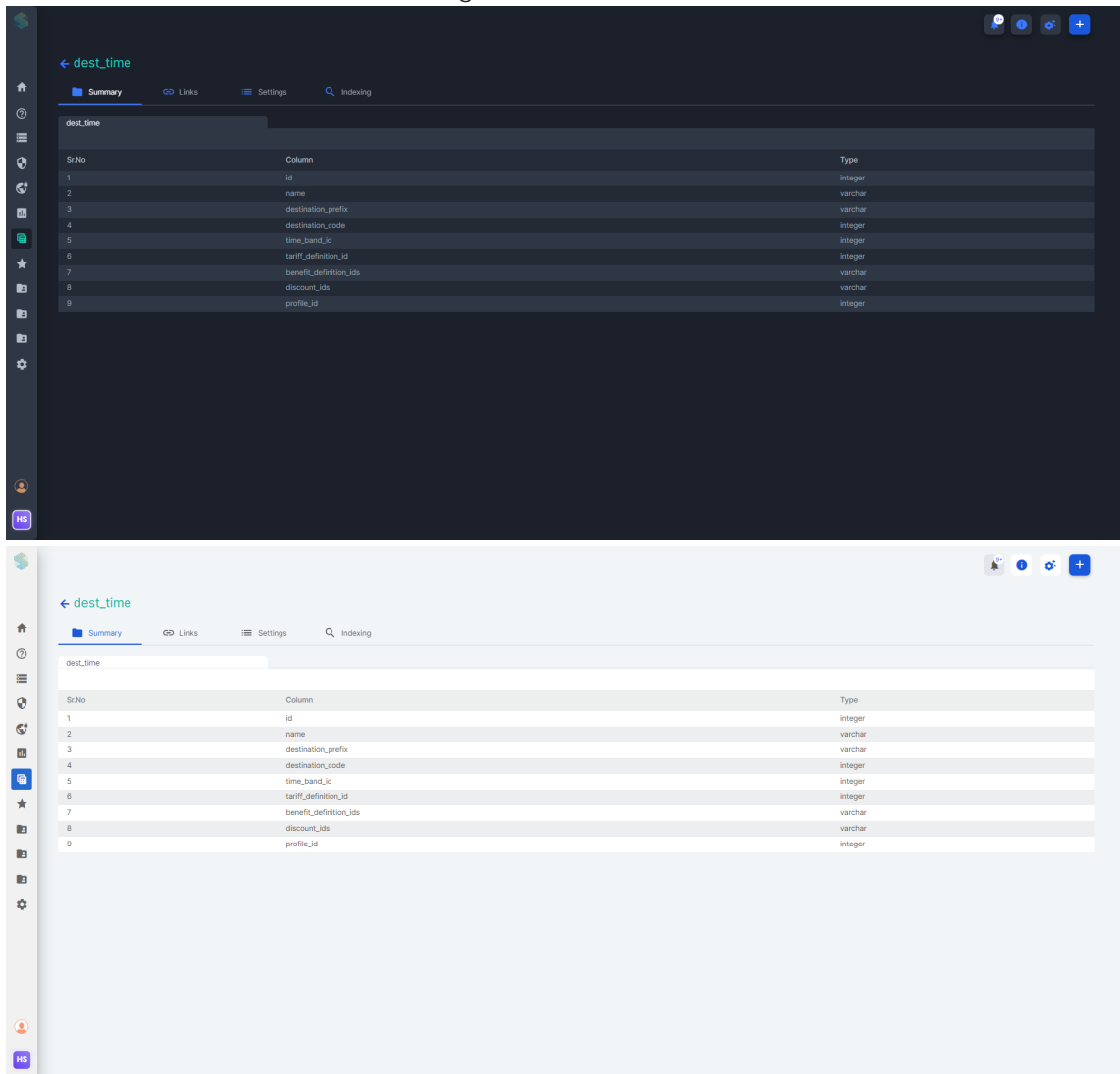
Example:

| id | ref_fields | profile_id | ref_code | de- scrip- tion |
|----|--|------------|-------------|-----------------------|
| 1 | aNumber=callingNumber, bNumber=calledNumber, serviceType=servicingType, callType=callingType, startTime=startingTime, count=smsCount, usage=dataUsage, apn=dataApn, isCUG=isCUG, isFNF=isFNF, isSPE=isSPE, duration=vDuration, recordType=recordoType, characterLength=scharacterLength, destinationCode=destinationCode | 1 | sample_code | sample |

2. Create **dest_time** cache.

- *Steps to create dest_time cache*

1. Navigate to Data Catalog and Create Table. The column names to be configured should be matched with `dest_time`. For more information, see [Create Table](#).
2. Ensure that the columns of the configured table is same as backend table as below:



| Sr.No | Column | Type |
|-------|------------------------|---------|
| 1 | id | integer |
| 2 | name | varchar |
| 3 | destination_prefix | varchar |
| 4 | destination_code | integer |
| 5 | time_band_id | integer |
| 6 | tariff_definition_id | integer |
| 7 | benefit_definition_ids | varchar |
| 8 | discount_ids | varchar |
| 9 | profile_id | integer |

3. There are three backend tables. `subscriber`, `subscriber_account`, `account_benefit`. The data from these three tables has to be populated into the ignite cache. In order to do that, you have to **create three different cache pipelines**. ARC will do the cache lookup for the created cache pipelines.

Configure the following three cache pipelines in Data Management Studio:

- ***subscriber_rateplan_cache***

The subscriber rate plan cache configuration:

As a part of subscriber rate plan cache configuration, follow the below steps:

1. While configuring the **RDBMS** source configuration, select the postgres database details where the table actually exist.
2. In **Collection Details** tab, provide the required columns in SQL Query pane for subscriber table.

3. In the **Parser**, all the details get reflected in **Record Structure Definition** tab.
4. In **Mapping**, the system add the **recordNumber** and **recordAddress**. Since this information is not required in cache, clear the checkboxes of **recordNumber** and **recordAddress** and click **Apply**.
5. In **Cache** sink configure the following fields.
 - a. **Connection URL**: Select the **Ignite Cache URL: Port** name as "ignite.staging:10800 in **Connection URL** field.
 - b. **Cache Name**: Cache name is hard coded and enter **subscriber_rateplan_cache** as cache name.
 - c. **Expiry Duration**: Enter the expiry duration (87654000000000) in seconds.
 - d. **Cache Value Update Strategy**: Select the **Cache Value Update Strategy** as **OVERRIDE**.
 - e. Configure the **Key:Value** as follows:

Key: \$phone_number:: \$active_status:: \$profile_id

Value: \$rate_plan_id:: \$addon_benefit_ids:: \$id:: \$name

- **account_benefit_cache**

The account benefit cache configuration:

As a part of account benefit cache configuration, follow the below steps:

1. While configuring the **RDBMS** source configuration, select the postgres database details where the table actually exist.
2. In **Collection Details** tab, provide the required columns in SQL Query pane for subscriber table.
3. In the **Parser**, all the details get reflected in **Record Structure Definition** tab
4. In **Record Integrity** Tab select the disable radio button.
5. In **Mapping**, the system add the **recordNumber** and **recordAddress**. Since this information is not required in cache, clear the checkboxes of **recordNumber** and **recordAddress** and click **Apply**.
6. In **Cache** sink configure the following fields.
 - a. **Connection URL**: Select the **Ignite Cache URL: Port** name as "ignite.staging:10800 in **Connection URL** field.
 - b. **Cache Name**: Cache name is hard coded and enter **account_benefit_cache** as cache name.
 - c. **Expiry Duration**: Enter the expiry duration (87654000000000) in seconds.
 - d. **Cache Value Update Strategy**: Select the **Cache Value Update Strategy** as **OVERRIDE**.
 - e. Configure the **Key:Value** as follows:

Key: \$phone_number:: \$benefit_definition_id:: \$profile_id

Value: \$value:: \$valid_from:: \$valid_to:: \$id

- *subscriber_account_cache*

The subscriber account cache configuration:

As a part of subscriber account cache configuration, follow the below steps:

1. While configuring the **RDBMS** source configuration, select the postgres database details where the table actually exist.
2. In **Collection Details** tab, provide the required columns in SQL Query pane for subscriber table.
3. In the **Parser**, all the details get reflected in **Record Structure Definition** tab.
4. In **Mapping**, the system add the **recordNumber** and **recordAddress**. Since this information is not required in cache, clear the checkboxes of **recordNumber** and **recordAddress** and click **Apply**.
5. In **Cache** sink configure the following fields.
 - a. Connection URL: Select the **Ignite Cache URL: Port** name as "ignite.staging:10800 in Connection URL field.
 - b. Cache Name: Cache name is hard coded and enter **subscriber_account_cache** as cache name.
 - c. Expiry Duration: Enter the expiry duration (87654000000000) in seconds.
 - d. Cache Value Update Strategy: Select the Cache Value Update Strategy as **OVERRIDE**.
 - e. Configure the **Key:Value** as follows:

Key: \$phone_number:: \$profile_id

Value: \$unbilled_charge:: \$billed_charge:: \$id

4. Once cache pipelines are created, create first pipeline using Local File Sink.

- *Steps to create pipeline*

It is first pipeline to be created. It is created with help of Data Source, Transformation Operator (Parser / Mapping), and Local File Sink.

Steps to configure first pipeline is listed below:

1. Drag and drop the Data Source and Parser into Canvas.
2. Establish connection between them.
3. Configure the **Data Source**.
4. **Configure Parser**. Below are the points to be remembered while configuring Parser:

- a. While configuring **Parser**, Datetime Column on which record is sorted should be in date time format. Rest all other datetime / timestamp column should be converted to String Datatype. For more information, see [Record Structure Definition](#).
 - b. For **ASCII Parser**, Mapping should be added. While configuring Mapping, uncheck **Record Address and Record Length**. To configure Mapping, see [Mapping](#). For more information on ASCII Parser, see [ASCII Parser](#).
 - c. For **CSV Parser**, Mapping is optional. For more information on CSV Parser, see [CSV Parser](#).
5. Once Transformation Operator is configured, drag and drop Local Sink. Configure it.
 6. After Configuration, save the pipeline and schedule the pipeline.
5. Once first pipeline is created, create second pipeline with File Sorting, Transformation Operator(Parser/ Mapping/ Sampling/ ARC) and Sink.

- *Steps to create pipeline*

It is created with help of File Sorting, Transformation Operator (Parser/ Mapping), ARC and Sink.

Steps to configure second pipeline is listed below:

1. Link the local cache for LPM with the dataset dest_time that is created in Data Catalog. For more information, see [Configuring Local Cache Settings](#).
2. Drag and drop the File Sorting and Parser into Canvas.
3. Establish connection between them.
4. Configure the **File Sorting**. For more information, see [File Sorting](#).
5. **Configure Parser. Select the required Parser to parse the incoming CDR record. In Parser**, incoming CDR records should be added into suitable format to charge the CDR. Below are the few column details that should be added under Record Structure Definition:
 - a. **aNumber**: It refers to calling number.
 - b. **callType**: It refers to call type either local or international.
 - c. **recordType**: It refers to record type either incoming (I) or outgoing (O). Rating and Charging is carried out for outgoing calls.
 - d. **serviceType**: It refers to service type i.e., Voice, SMS, or GPRS.
 - e. **bNumber**: It refers to called number or destination number.
 - f. **startTime**: It refers to start time of call.
 - g. **duration**: It refers to duration of the call. It is used for service type Voice.
 - h. **characterLength**: It refers to character length of the message. It is used for service type SMS.

- i. **count:** It refers to count of the message. It is used for service type SMS. Message is counted as 1 when SMS length is 140 character.
- j. **usage:** It refers to the usage of data cdr. It is used for service type GPRS.
- k. **apn:** It refers to apn number. It is used as destination for GPRS records.
- l. **IsCUG:** CUG refers to Closed User Group. If it is flagged then the incoming CDR is closed user group call.
- m. **IsFNF:** FNF refers to Friends and Family. If it is flagged then the incoming CDR is having details of friends and family.
- n. **IsSPE:** SPE refers to Special Calls. If it is flagged then the incoming CDR is having detail of special calls such as toll free numbers.
- o. **arc_logs:** It provides you the trace of logs for backend table.

Note:

While configuring Parser, set datetime format of sorting column as "yyyy-MM-dd HH:mm" format. Rest all other datetime / timestamp column should be converted to String Datatype. For more information, see [Record Structure Definition](#).

- 6. After Parser configuration, drag and drop the Mapping from Transformation Operator. Establish connection between Parser and Mapping.
 - 7. Mapping operator is used for matching destination code. Use apply function **\$longest-match()** to match the destination code.
 - 8. After Mapping configuration, drag and drop the ARC from Transformation Operator. Establish connection between Mapping and ARC.
 - 9. ARC operator is used for rating and charging CDR. Click on **Vertical Ellipsis > Edit**. **Configure:ARC** window is displayed. For configuration, see [ARC Operator Configuration](#).
 - 10. Finally the charged CDR is pushed into Data Sink.
6. After creating and configuring Pipelines, now deploy them.
- *Steps to deploy pipeline*
 - 1. Run the following three cache pipelines to makes sure that the data for lookup is available in cache. For more information to run the pipeline, see [Run the Pipeline](#).
 - a. **subscriber_rateplan_cache**
 - b. **account_benefit_cache**
 - c. **subscriber_account_cache**
 - 2. After running the cache pipelines, run the **first pipeline**.
 - 3. Once first pipeline is run, run the second pipeline.

Note:

Scheduler supports CRON driven. Minimum CRON time should be equal or greater than 1 minutes. Timer Driven Schedule is not supported. For more information, [see Pipeline Scheduler](#).

Configuring Local Cache Settings

Link the local cache for LPM with the dataset `dest_time` that is created in Data Catalog. While configuring the local cache settings, ensure the below steps to be followed. For detailed information on configuring local cache, see [Configure Local Cache](#).

1. Configure the local cache settings by selecting **dest_time table** from **Dataset Name** dropdown and name the cache as `dest_time_cache` under Cache Name field.
2. Configure **Cache Type** as **Longest Prefix Match**. Set the **destination_prefix** as **Key** and the **destination_code** as **Value**.
3. Schedule the local cache to one minute. To schedule the cache, see [Schedule](#).

ARC Operator Configuration

ARC window has two tabs to configure: ARC Configuration, and Import / Export Configuration.

1. **ARC Configuration:** In this tab, complete the details to fetch the profile, and lookup tables to rate and charge the CDR.

Complete the below details:

- a. **Select Profile:** In this select the require profile and version number for rating and charging.
 - i. **Profile:** Each profile contains the set of tables required to rate and charge the incoming CDR. It keeps varying as per business requirement. Select the required profile from the dropdown list. Contact administrator to create profile.
 - ii. **Version:** It displays the version number for profile. Select the profile from the dropdown list.
- b. **Config Connection Details:**
 - i. **JDBC URL:** Enter the postgres database url link to fetch the table configuration details. It is used to fetch the profiles created.
 - ii. **User Name:** Enter the postgres database user name.
 - iii. **Password:** Enter the postgres database password.
- c. **Ignite Cache Connection Details:** Enter the Ignite URL to cache the tables. It helps to improve the performance.

- d. **Enable Charging:** If enabled, for particular CDR benefit is applied while rating and charging. If disabled, for particular CDR only rating will happen and arc_charge output is 0.
 - e. **Click Done to save the configuration.**
2. **Import / Export Configuration:** This tab helps you to import the new ARC configuration or export the existing ARC configuration.

Complete the below details:

- a. **Import / Export Directory:** Enter the path directory where ARC configuration table details to be imported or exported. It will help you to either import or export the specific ARC configuration.
- b. **Import With New Profile:** If selected, it allows you to import the profile with new name. Select the checkbox, and Profile Name field is enabled.
- c. **Profile Name:** It is enabled once Import With New Profile checkbox is selected. Enter the Profile Name.
- d. **Import:** Click Import to import the new ARC configuration. You should import complete ARC configuration tables not single table.
- e. **Export:** Click Export to export the existing ARC configuration.

Go back to the [Pipeline Configuration Page](#) to know how to add and configure Data Sink to the pipeline.

Filter

Last updated on February 5, 2024

A Filter component can be used to remove certain input records from an output data file, or to route information to a particular output data file. For example, a Filter can be set up to route zero duration call records to one output data file and all other call records to a different output file. You can provide the expression to validate, based on which the data is filtered. The expression validation is based on the Boolean value.



This page provides step-wise configuration about the Transformation Operator. After you configure the Transformation Operator, go back to the [Pipeline Configuration Page](#) to know how to add and configure **Data Sink** to your pipeline.

To configure Filter, you must configure the following:

1. In Canvas, expand **Transformation Operator**. Drag and drop **Filter** in canvas.
2. Provide a datasource output link to the Filter.
3. Click the vertical ellipsis and select **Edit**.

4. The **Configure: Filter** window is displayed. Click on the **expand** button to set conditions.
5. Condition pane is displayed. Click on **Wizard** to open the condition settings.
6. Filter component has following options. Configure the condition settings accordingly.
 - a. **Rule Name:** Provide the name for the rule.
 - b. **Condition 1:** Click **Choose Wizard based Rule** to reveal the condition parameters.
 - i. **Select Column:** Allows you to select the data column from the input source.
 - ii. **Operation:** Allows you to set the filter operation criteria. It includes **Contains**, **Equalignore-case**, **Equals**, **Greater than**, and **Greater than Equals**.
 - iii. **Type:** Allows you to choose the type of data. It includes **Value** and **Column**. In case if **value** is selected, you need to provide compare expression. And if you have selected the type as **column**, then compare column needs to be selected.
 - iv. **Compare:** Allows you to add the comparison value based on data type of the column you are comparing to.
Note: For **datetime** datatype, the value is accepted in milliseconds. For example, **Datetime Type** for **2021-01-10**, the value should be **1610254403000** milliseconds.
 - v. **Select Column:** Choose the type of column for comparison.
 - vi. **And/Or:** Allows you to define relation between multiple conditions.
 - c. **Condition 2:** To add multiple condition, click **Choose Wizard based Rule** and provide the details as explained in **Condition 1**.
7. After configuring the filter conditions, click **Apply** to save.
8. For adding another Rule, click **Add Rule** and follow the same condition setup.
9. Configured examples for **Column** type and **Value** Type:
10. Go back to the [Pipeline Configuration Page](#) to know how to add and configure Data Sink to the pipeline.

Flattener

Last updated on May 19, 2025

This operator allows you to store data in one or a few tables containing all the information. It can flatten specific list or map. It flattens the entire record to produce a record with no nested fields. It works in similar way as Parser. It can handle json and xml file format.

JSON Flattener

This Transform Operator flattens JSON format data into table format. To configure **JSON Flattener**, you must configure the following:

1. In Canvas, expand **Transformation Operator**. Drag and drop **Flattener** in canvas.

2. Provide an input link to the Flattener. Input link can be from a Datasource or ASN Parser (Transformation Operator).
 - a. **Case 1:** When the input link is from a Datasource
 - i. Click the vertical ellipsis and select **Edit**. A **Select File Type** window will be displayed.
 - ii. Select **Data Flattener (JSON)** option from **File Type** drop-down. You can either upload a sample json file or add structure manually.
 - iii. To add the structure manually, refer section [Add Manually](#)
 - iv. To upload json sample file, click **Upload Sample File (Optional)** field to browse and select the json file.
 - v. After selecting the file, click **Next**. A **Configure: Data Flattener (JSON)** window is displayed.
 - b. **Case 2:** When the input link is from ASN Parser
 - i. Click the vertical ellipsis and select **Edit**. A **Configure: Data Flattener (JSON)** window is displayed.
3. Configure the flattener details in the **Configure: Data Flattener (JSON)** window. It has two tabs: [File Structure](#) and [Record Integrity](#).
4. **File Structure Tab:** This tab displays the structure of the data.

- *Steps to configure*

To save the structure, you need to select the level in **Details** field. By default, **None** is selected.

You can add more data structure. For adding more structure definitions manually, configure the following details:

1. **Add Node:** Hover the cursor over the data row, a **Add Node** button appears at the right end of the row. It allows you to add more nodes manually.
 - a. Click **Add Node** and choose from **Record**, **Sequence Of**, or **Field**.
 - b. **Record:** Enter the record name.
 - c. **Sequence Of:** A **Sequence Of** repeats until all data records of a particular record reference are parsed and exits once the interpreter registers that no data record has been parsed.
 - d. **Field:** Commonly used to refer to a column in a database or a field in a data entry form or web form. The field may contain data to be entered as well as data to be displayed. Configure the following:
 - i. **Field Name:** Enter the field name.
 - ii. **Type:** Choose the type of data from the drop-down.
 - iii. **Column Name:** Column name is filled automatically after you have entered the field name. You can edit and enter as required.
2. **Enable Validation:** It allows you to validate the input file as per defined schema. It helps to avoid situations such as data contamination with incorrect records or failure during the

transformation process. Select the checkbox to perform the validation. Once selected, the system will check the input data against the specified schema while processing. The below options are displayed:

- a. **Skip Record:** It allows you to skip the records that are not matching with the File Structure. It looks for data type mismatches. For example, you have defined the timestamp column in "dd-mm-yyyy" format, and the input file has a few records with the timestamp column in "yy-mm-dd" format. In this case, such records will be skipped during the validation process.
- b. **Skip File:** It allows you to skip the complete input file if it is not matching with the File Structure.
- c. The below table shows the datatype accepted by the flattener when processing the incoming file/record:

| File Structure Data Type (Defined) | Data Type of Incoming File / Record (Accepted) |
|------------------------------------|---|
| String | String, Integer, Decimal, Long, Boolean, Datetime |
| Integer | Integer |
| Decimal | Decimal, Integer |
| Long | Integer |
| Boolean | Boolean |
| Datetime | Datetime (format should match) |

3. Click the **Record Integrity** tab to configure.
5. **Record Integrity Tab:** This tab allows you to perform the missing record sequence check and duplicate record from the file.

Complete the configuration:

- a. **Record Sequence Check:** This check allows you to find the missing record sequence from the file. Select the **Enable** radio button to configure missing record sequence check. For more information, see [Missing Record Sequence Check](#).
- b. **Duplicate Record Check:** This check allows you to find the duplicate records from the file. For more information, see [Duplicate Record Check](#).
- c. Click **Apply** to save the configuration.

Note: Record Integrity functionality is same for JSON and XML Flattener.

6. Actions available in window:

- a. **Delete:** Allows you to delete the node.
- b. **Copy:** Allows you to copy the node.
- c. **Paste:** Allows you to paste the copied entry.
- d. **Cut:** Allows you to cut the node.
- e. **Upload:** Allows you to [upload](#) a JSON file. It can handle a JSON file format.
- f. **Reset:** Allows you to reset the modified or added file structure and revert back to the initial structure.
- g. **Move Up:** Allows you to move the record upwards in the structure definition.
- h. **Move Down:** Allows you to move the record downwards in the structure definition.

XML Flattener

This Transform Operator flattens XML format data into table format. To configure **XML Flattener**, you must configure the following:

1. In Canvas, expand **Transformation Operator**. Drag and drop **Flattener** in canvas.
2. Provide an input link to the **Flattener** from a Datasource.
3. Click the vertical ellipsis and select **Edit**. A **Select File Type** window will be displayed.
4. Select **Data Flattener (XML)** option from **File Type** drop-down. You can either upload a sample file or add structure manually
 - a. To add the structure manually, refer section [Add Manually](#).
 - b. To upload sample xml file, click **Upload Sample File (Optional)** field to browse and select the xml file.
 - c. Click **Next**. A **Configure: Data Flattener (XML)** window is displayed.
5. Configure the flattener details in the **Configure: Data Flattener (XML)** window. It has three tabs: **XML**, **Structure Definition**, and **Record Integrity**.
6. **XML Tab:** This tab displays the XML file details.
 - *Steps to configure*
 1. Once the file is uploaded, Click the expand icon to view xml file preview.
 2. In case if any modifications are required in the xml file, you can perform the edit on this interface.
 3. Click **Decode** to check for any exception or error in the file. While decoding the xml file to tree structure, if there is any error or issue in the file, it will prompt in the error log panel.
 4. If there is any syntax or exception error in the xml file, then it is displayed in the error panel.

5. While decoding, if there is no error, you are navigated to **Structure Definition** tab.
7. **Structure Definition Tab:** This tab displays the structure of the data. Flattener has created the **structure for the uploaded XML file. The nested XML document is flattened into a simple key/value pair document.**

- *Steps to configure*

To save the structure, you need to select the level in **Details** field. By default, **None** is selected. Maximum two levels are supported.

You can add more data structure. For adding more structure definitions manually, configure the following details:

1. **Add Node:** Hover the cursor over the data row, a **Add Node** button appears at the right end of the row. It allows you to add more nodes manually.
 - a. Click **Add Node** and choose from **Record**, **Sequence Of**, or **Field**.
 - b. **Record:** Enter the record name
 - c. **Sequence Of:** A **Sequence Of** repeats until all data records of a particular record reference are parsed and exits once the interpreter registers that no data record has been parsed.
 - d. **Field:** Commonly used to refer to a column in a database or a field in a data entry form or web form. The field may contain data to be entered as well as data to be displayed. Configure the following:
 - i. **Field Name:** Enter the field name.
 - ii. **Type:** Choose the type of data from the drop-down.
 - iii. **Column Name:** Column name is filled automatically after you have entered the field name. You can edit and enter as required.
 - iv. Example:
2. **Enable Validation:** It allows you to validate the input file as per defined schema. It helps to avoid situations such as data contamination with incorrect records or failure during the transformation process. Select the checkbox to perform the validation. Once selected, the system will check the input data against the specified schema while processing. The below options are displayed:
 - a. **Skip Record:** It allows you to skip the records that are not matching with the **File Structure**. It looks for data type mismatches. For example, you have defined the timestamp column in "dd-mm-yyyy" format, and the input file has a few records with the timestamp column in "yy-mm-dd" format. In this case, such records will be skipped during the validation process.
 - b. **Skip File:** It allows you to skip the complete input file if it is not matching with the **File Structure**.
 - c. The below table shows the datatype accepted by the flattener when processing the incoming file/record:

| File Structure Data Type (Defined) | Data Type of Incoming File / Record (Accepted) |
|------------------------------------|---|
| String | String, Integer, Decimal, Long, Boolean, Datetime |
| Integer | Integer |
| Decimal | Decimal, Integer |
| Long | Integer |
| Boolean | Boolean |
| Datetime | Datetime (format should match) |

3. Click **Record Integrity** tab to configure.

8. **Record Integrity:** This tab allows you to perform the missing record sequence check and duplicate record from the file.

Complete the configuration:

- Record Sequence Check:** This check allows you to find the missing record sequence from the file. Select the **Enable** radio button to configure missing record sequence check. For more information, see [Missing Record Sequence Check](#).
- Duplicate Record Check:** This check allows you to find the duplicate records from the file. For more information, see [Duplicate Record Check](#).
- Click **Save** to save the configuration.

9. Actions available in window:

- Delete:** Allows you to delete the node.
- Copy:** Allows you to copy the node.
- Paste:** Allows you to paste the copied entry.
- Cut:** Allows you to cut the node.
- Upload:** Allows you to [upload](#) a XML file.
- Reset:** Allows you to reset the modified or added file structure and revert back to the initial structure.
- Move Up:** Allows you to move the record upwards in the structure definition.
- Move Down:** Allows you to move the record downwards in the structure definition.

Add Structure Manually (JSON Flattener)

You can add the structure manually by entering the details on the configuration widow.

1. In Canvas, expand **Transformation Operator**. Drag and drop **Flattener** in canvas.
2. Provide a input link to the Flattener.
3. Click the vertical ellipsis and select **Edit**.
4. A **Select File Type** window is displayed. Select **Data Flattener (JSON)** option from **File Type** drop-down and click **Next**.
5. A **Configure: Data Flattener (JSON)** window is displayed.
6. Hover the cursor over the **ROOT** data row, a **Add Node** button appears at the right end of the row. It allows you to add file structure definitions manually.
7. Click **Add Node** and choose from **Field**, **Record**, or **Sequence Of**.
 - a. **Field**: Commonly used to refer to a column in a database or a field in a data entry form or web form. The field may contain data to be entered as well as data to be displayed. Configure the following:
 - i. **Field Name**: Enter the field name.
 - ii. **Type**: Choose the type of data from the drop-down.
 - iii. **Column Name**: Column name is filled automatically after you have entered the field name. You can edit and enter as required.
 - b. **Record**: Enter the record name.
 - c. **Sequence Of**: A **Sequence Of** repeats until all data records of a particular record reference are parsed and exits once the interpreter registers that no data record has been parsed.
8. Click on minus icon to collapse, and plus icon to expand and view the file structure.

Add Structure Manually (XML Flattener)

You can add the structure manually by entering the details on the configuration widow.

1. In Canvas, expand **Transformation Operator**. Drag and drop **Flattener** in canvas.
2. Provide a input link to the Flattener.
3. Click the vertical ellipsis and select **Edit**.
4. A **Select File Type** window is displayed. Select **Data Flattener (XML)** option from **File Type** drop-down and click **Next**.
5. A **Configure: Data Flattener (XML)** window is displayed.
6. **File Structure** tab is displayed. It allows you to create a simple key/value pair data file structure.
7. Hover the cursor over the **ROOT** data row, a **Add Node** button appears at the right end of the row. It allows you to add file structure definitions manually.

8. Click **Add Node** and choose from **Field**, **Record**, or **Sequence Of**.
 - a. **Field**: Commonly used to refer to a column in a database or a field in a data entry form or web form. The field may contain data to be entered as well as data to be displayed. Configure the following:
 - i. **Field Name**: Enter the field name.
 - ii. **Type**: Choose the type of data from the drop-down.
 - iii. **Column Name**: Column name is filled automatically after you have entered the field name. You can edit and enter as required.
 - b. **Record**: Enter the record name.
 - c. **Sequence Of**: A **Sequence Of** repeats until all data records of a particular record reference are parsed and exits once the interpreter registers that no data record has been parsed.
 9. Click on minus icon to collapse, and plus icon to expand and view the file structure.
 10. Click **Save**.
-

Missing Record Sequence Check

In Data Management pipeline, after loading the data, if any records are missed in between file processing, record sequence check feature identifies the missing records from the input files and generates an alert notification for missing record sequence. You can view the triggered alert in Triggered alerts screen.

To configure Record Sequence:

1. From the configure: **Data File Parser** Screen, Click the **Record Integrity** tab. By default, record sequence check option is disabled.
2. Click **Enable**. The Record Sequence window will be expanded automatically.
3. Configure the following details and click **Save**.
 - a. **General Details**
 - i. **Integrity Name**: This is a general name given to integrity file. Enter a name for the Integrity check.
 - b. **Record Sequence Check Configuration**
 - i. **Select Column**: This provides the list of columns for selection. Click the **Select Columns** dropdown and select the required column. **Select Column** dropdown displays only integer and long data type columns. After selecting the **Select Column**, you can not edit/modify the selected column in record structure definition.
 - ii. **Record Start Number**: Record start number is the sequence number from where the record sequence need to start.
 - iii. **Record End Number**: Record end number is the sequence number where the record sequence ends.

- iv. **Wait Time:** Wait Time is the time duration after loading a file that a system holds before raising an alert. Enter the required wait time.
- v. **Time Frame:** Time Frame quantifies the units of wait time. Click the time frame dropdown and select the required time frame(**seconds/minutes/hours**).

Configured Example:

- In the above configured example, if you configure Record Start number as 10 and Record End number as 20 and your input file has the records only from 10 to 15. In this scenario, as per record sequence configuration, the records must exist from 10 to 20 but they exist only from 10 to 15 and the remaining records from 16 to 20 are missing. Since the wait time for the record sequence is configured as one hour, after loading the file, system waits for one hour and generates an alert for the missing record sequence.
- If any missing record sequence is detected, an alert is generated and you can view the alert in **Triggered Alerts** screen as shown below.

Note:

- In multiple record reference scenario, the dropdown for the record sequence identifier column will display only the columns that are present in all schema.

Duplicate Record Check

In Data Management Pipeline, while loading the data, if any duplicate record is found while processing the file, duplicate record check feature identifies the duplicate records from the input files and generates an alert notification.

To view the triggered alert either Click Bell icon in the Pipeline Repository or visit alert screen. For more information on Triggered Alerts, see [Triggered Alerts](#).

Note:

- Incoming columns should have Timestamp datatype to check and generate alert in case of duplicate records are identified. For more information on how to create alert, see [Create Alert](#).
- It will process -1 day records. **For example:** Suppose System Date is 30th September'24 15:30:00 and you perform duplicate record check, records are checked from 29th September '24 00:00:00 till 30th September'24 15:30:00.

To configure, click [Click Here](#). New Row is added.

Configure below details and Click **Save**.

1. **Record Type:** This field displays the name of incoming file. In case incoming file has multiple worksheets, worksheet names are displayed in the dropdown list. Select the worksheet in which duplicate record should be performed.

2. **Timestamp Columns:** This field displays the date columns for checking the duplicate record. Select the date column from the dropdown list.
3. **Columns:** This field displays the list of columns on which duplicate check to be performed. Select the columns from the dropdown list. You can select multiple columns.
4. Click **Add Row** when you want to perform duplicate record check on multiple worksheet.
5. Click **delete** icon to remove the rows.
6. **Configured example:**
7. If any duplicate record is detected, an alert is generated and you can view the alert in **Triggered Alerts** screen.

Note:

Click on URL link displayed in alert message to view the duplicate record details.

JSON Mapper

Last updated on April 1, 2025

JSON Mapper is used to perform all mapping functions directly on the output of ASN.1 Parser. Unlike **traditional mapper**, the **hierarchical structure is maintained in JSON Mapper** that we can use to access any arbitrary field. JSON Mapper contains local variables so that we can use fields from one **schema in another schema**. This feature allows you to **define/map the data to be displayed in specific format**. If the data is loaded in any format, and you want to change the record names, configure and map them in this window. JSON Mapper not only changes column names, it can change data type of a column and you can add new columns with string, numerical and Boolean operators acting on incoming columns.

To configure JSON Mapper, you must configure the following:

1. In Canvas, expand **Transformation Operator**. Drag and drop **JSON Mapper** in canvas.
2. **Provide the input link to the JSON Mapper.**
3. Click the vertical ellipsis, and select **Edit**.
4. A **Configure: JSON Mapper** window is displayed. It has three tabs: **Input Events**, **Mapping** and **Record Integrity**.
5. Configure the **Input Event** tab:
 - a. **Variable Name:** Enter the name for the variable.
 - b. **Data Type:** Select the required data type from the drop-down.
 - c. **Nullify Initial Value:** If selected, the initial value for that particular local variable will be null.

- d. **Initial Value:** This is enabled only when **Nullify Initial Value** is not selected. Although a Variable can derive a value from an Input Data Field within a Mapping Node, it must always be given an initial value. There are instances where a constant value is required. Example: In case if you want to do summation, you can give the initial value as zero.

Examples of initial values for each data type for Variable type are:

- i. **Boolean:** True and False.
- ii. **Integer:** 0
- iii. **String:** " (empty string)

Note : Initial value cannot be given to DateTime value.

- e. **Apply Function:** This column computes the operations available in the configured function and the value outcome is assigned to the **Variable Name** parameter. In this section, the parameters can be constant, variable or another column from any schema. If you want to access any particular column in the apply function then you need to give the schema name followed by column. e.g. \$toString(schemaname.column).

Note : Since input events are evaluated for every incoming record regardless of schema, it is recommended to use the **\$isNull()** function to avoid the value getting reset. This will allow us to retain existing value inside local variable in case the column value is null.

- f. Click on Mapping tab.

- 6. **Mapping Tab:** This tab has list of schema to map the input event.

- *Steps to configure*

Schema Configuration

1. Select the required schema and click on **Edit** icon.
2. Configure the following fields in **Mapping** tab:
 - a. **Old Column Name:** Displays the name of the column to be mapped.
 - b. **Old Data Type:** Displays the data type to be mapped.
 - c. **New Column Name:** Displays the column name for the mapped column.
 - d. **New Data Type:** Displays the data type for the new column.
 - e. **Apply Function:** Displays the configured apply function.
 - f. **Add Field:** Click to add the new field.
 - i. **New Column Name:** Enter the name of the column to be mapped.
 - ii. **New Column Data Type:** Select the data type for the new column from the drop down.
 - iii. **Apply Function:** This column computes the operations available in the configured function and the value outcome is assigned to the **New column Name** parameter. In this section, the parameters can be constant, variable or another column from the same schema. If you want to access any particular column in the apply func-

tion then you need to specify complete hierarchy by using dot operator e.g. `$toString(fileCreationTimestamp.localTimestamp)`.

Note: Whenever there is a mismatch in the datatype of the apply function and column, a snackbar message is displayed when you save the configuration. For example, you want to display the current datetime using the apply function "`$now()`". The datatype for this function is datetime. You create a new column with a decimal datatype, which is a mismatch. When you try to save the configuration, a snackbar message is displayed: *"Invalid configuration for: 'JSON Mapper.'"*

- iv. **Example:** If the function configured is `$multiply ($add (const 30,const 20), salary)` then the parameter **salary** must be defined. To use the existing table instances, functions or columns, click **Ctrl+Space**, it displays a pop up to select the configurations.
- v. Click **Done**. New column is added at the bottom of the column list.
- g. **Edit:** To edit any column, click the **Edit** icon.
- h. **Delete:** To delete a column, click the **Delete** icon.
- i. **Remove All Columns:** Click to remove all the columns in the grid.
- j. **Move Up/Move Down :** Use the move up and move down icon to change the sequence of the columns.

Enable Validation

It allows you to validate the run time exceptions in added mapping functions due to schema mismatch in function return type. It helps to avoid situations such as data contamination with **incorrect records or failure during the transformation process**. Select the checkbox to perform the validation. The below options are displayed:

1. **Skip Record:** It allows you to skip the records where run time exception occurs in added mapping function. It looks for function schema mismatches. For example, mismatch in function return type, and invalid input parameters.
2. **Skip File:** It allows you to skip the complete input file if it is not matching with the function schema.
3. The below table shows the datatype accepted by the JSON mapper when processing the incoming file/record:

| File Structure Data Type (Defined) | Data Type of Incoming File / Record (Accepted) |
|------------------------------------|--|
| String | String, Integer, Decimal, Long, Boolean, Date-time |
| Integer | Integer |

| | |
|----------|--------------------------------|
| Decimal | Decimal, Integer |
| Long | Integer |
| Boolean | Boolean |
| Datetime | Datetime (format should match) |

Click the **Record Integrity** tab to configure.

7. **Record Integrity Tab:** This tab allows you to perform the missing record sequence check and duplicate record from the file.

Complete the configuration:

- a. **Record Sequence Check:** This check allows you to find the missing record sequence from the file. Select the **Enable** radio button to configure missing record sequence check. For more information, see [Missing Record Sequence Check](#).
- b. **Duplicate Record Check:** This check allows you to find the duplicate records from the file. For more information, see [Duplicate Record Check](#).
- c. Click **Done** to save the configuration.

Missing Record Sequence Check

In Data Management pipeline, after loading the data, if any records are missed in between file processing, record sequence check feature identifies the missing records from the input files and generates an alert notification for missing record sequence. You can view the triggered alert in Triggered alerts screen.

To configure Record Sequence:

1. From the configure: **Data File Parser** Screen, Click the **Record Integrity** tab. By default, record sequence check option is disabled.
2. Click **Enable**. The Record Sequence window will be expanded automatically.
3. Configure the following details
 - a. **General Details**
 - i. **Integrity Name:** This is a general name given to integrity file. Enter a name for the Integrity check.
 - b. **Record Sequence Check Configuration**
 - i. **Select Column:** This provides the list of columns for selection. Click the **Select Columns** dropdown and select the required column. **Select Column** dropdown displays only integer and long data type columns. After selecting the **Select Column**, you can not edit/modify the selected column in record structure definition.

- ii. **Record Start Number:** Record start number is the sequence number from where the record sequence need to start.
- iii. **Record End Number:** Record end number is the sequence number where the record sequence ends.
- iv. **Wait Time:** Wait Time is the time duration after loading a file that a system holds before raising an alert. Enter the required wait time.
- v. **Time Frame:** Time Frame quantifies the units of wait time. Click the time frame dropdown and select the required time frame(**seconds/minutes/hours**).

Configured Example:

- In the above configured example, if you configure Record Start number as 10 and Record End number as 20 and your input file has the records only from 10 to 15. In this scenario, as per record sequence configuration, the records must exist from 10 to 20 but they exist only from 10 to 15 and the remaining records from 16 to 20 are missing. Since the wait time for the record sequence is configured as one hour, after loading the file, system waits for one hour and generates an alert for the missing record sequence.
- If any missing record sequence is detected, an alert is generated and you can view the alert in **Triggered Alerts** screen as shown below.

4. Click **Done** to save the configuration.

Note:

- In multiple record reference scenario, the dropdown for the record sequence identifier column will display only the columns that are present in all schema.

Duplicate Record Check

In Data Management Pipeline, while loading the data, if any duplicate record is found while processing the file, duplicate record check feature identifies the duplicate records from the input files and generates an alert notification.

To view the triggered alert either Click Bell icon in the Pipeline Repository or visit alert screen. For more information on Triggered Alerts, see [Triggered Alerts](#).

Note:

- Incoming columns should have Timestamp datatype to check and generate alert in case of duplicate records are identified. For more information on how to create alert, see [Create Alert](#).
- It will process -1 day records. **For example:** Suppose System Date is 30th September'24 15:30:00 and you perform duplicate record check, records are checked from 29th September '24 00:00:00 till 30th September'24 15:30:00.

To configure, click [Click Here](#). New Row is added.

Configure below details and Click **Save**.

1. **Record Type:** This field displays the name of incoming file. In case incoming file has multiple worksheets, worksheet names are displayed in the dropdown list. Select the worksheet in which duplicate record should be performed.
2. **Timestamp Columns:** This field displays the date columns for checking the duplicate record. Select the date column from the dropdown list.
3. **Columns:** This field displays the list of columns on which duplicate check to be performed. Select the columns from the dropdown list. You can select multiple columns.
4. Click **Add Row** when you want to perform duplicate record check on multiple worksheet.
5. Click **delete** icon to remove the rows.
6. **Configured example:**
7. If any duplicate record is detected, an alert is generated and you can view the alert in **Triggered Alerts screen**.

Note:

Click on URL link displayed in alert message to view the duplicate record details.

LDC

Last updated on December 16, 2024

LDC refers to Long Duration Call (LDC). This feature stitches the **partial CDRs** of a long duration call into a single CDR. These **partial CDRs** are called legs. LDC picks the partial CDR from the Local file source and loads the stitched CDRs into **data sink**. Partial CDR are identified based on the Correlation ID associated with each partial record and the LDC Indicator. It identifies and handles the exceptions such as, missing legs, irregular order, and so on.



This page provides step-wise configuration about the Transformation Operator. After you configure the Transformation Operator, go back to the *Pipeline Configuration Page* to know how to add and configure **Data Sink** to your pipeline.

To configure LDC follow below steps:


1. In Canvas, expand **Transformation Operator**. Drag and drop LDC in canvas.
2. Connect LDC with data sink.
3. Click **Vertical Ellipsis of LDC > Click Edit**.
4. **Configure:** LDC window is displayed.
5. Complete the below details:
 - a. **General:** Complete the below general details:

- i. **Duration Field:** This column stores the CDR call duration. It stores the sum of all Partial CDRs duration after stitching. Select the required column from the dropdown list.
 - ii. **Status Field:** The Status field is used by the LDC task to indicate the status of a particular stitch. Set 1 if all CDRs got fully stitched; Set 2 if CDRs got stitched partially. Select the required column from the dropdown list.
 - iii. **Partial Record Count:** This field stores the total Number of Partial CDRs got Stitched. Select the required column from the dropdown list.
 - iv. **Correlation ID:** It is a unique identifier to detect the partial cdrs which belong to same call. This can be a single column or group of columns. Select the required column from the dropdown list.
 - v. **Stitched CDR Output:** Select all the columns of first Partial CDR or last Partial CDR. Select the required column from the dropdown list.
- b. **LDC Identification Type:** Complete the below identification type:
- i. **Field Type:** The LDC Indicator Field represents the field that contains the leg identifier (first, intermediate, or last leg) of the partial CDRs. Select a column from the dropdown list.
 - ii. **Initial Value:** Enter a value that identifies the initial leg. Ex: 0
 - iii. **Intermediate Value:** Enter a value that identifies an intermediate leg. Ex: 1
 - iv. **End Value:** Enter a value that identifies a final leg. Ex: 2
- c. **Ordering and Missing Detection:** Complete the below details:
- i. **Timeout Field:** It is used to calculate timeout expiry, tolerance and order the partial CDRs for a correlation ID. Select the timestamp column from the dropdown list.
 - ii. **Timeout (Seconds):** All the Partial CDRs needs to be arrived within the Timeout Period for stitching else it would be a stitched separately in new Group LDC. Enter the expiry period for stitching. The expiry duration must be entered only in seconds. **Example:** If the expiry period is 24 hrs, enter 86400 seconds.
 - iii. **Tolerance (Seconds):** This is the acceptable time gap between two consecutive partial CDRs within the same correlation ID.
- d. **Duplicate Check:** Duplicate Check dropdown lists all the configured columns to check the duplicate values in the selected column(s). This field is optional.
Complete the following dropdown:
- i. **Columns:** Column dropdown allows you to select the required columns to check for duplicate CDR values. Select the required column from dropdown and click **Add**. The duplicate cdrs are calculated based on configured columns from LDC configuration. All the duplicate cdrs are ignored during LDC stitching.
- e. **Configured Example:**
6. Click **Save** to save the configuration.

LookUp

Last updated on February 5, 2024

This operation allows you to look and match for any record value in the local or remote cache database tables.

 This page provides step-wise configuration about the Transformation Operator. After you configure the Transformation Operator, go back to the *Pipeline Configuration Page* to know how to add and configure **Data Sink** to your pipeline.

Consider an example: **Subscriber A** is an Airtel customer and is a fraudster who makes fraud calls. We must find the total number of matches found in the outgoing calls from this number. Lookup property and value must be configured and perform a match based on the required criteria.

Distributed Cache Configuration

Allows to choose a distributed map cache client to retrieve the value associated to a key. The coordinates that are passed to the lookup must contain the key 'key'. To configure Lookup as Distributed Cache type, you must configure the following:


1. In Canvas, expand **Transformation Operator**. Drag and drop **Lookup** in canvas.
2. Click the vertical ellipsis and select **Edit**.
3. A **Select Cache Type** window is displayed. Select the **Distributed Cache** type and click **Next**.
4. A **Configure Lookup** window is displayed.
5. This tab allows you to configure the look up details. Configure the following:
 - a. **Configuration Details**
 - i. **Lookup Behaviour**: Select the lookup behavior on what needs to be performed. Example:
Return first matching row - will return the value of first match for the configure property.
Return count of matching rows - will return the count of all matching rows for the configured property.
 - b. **Pipelines Containing Cache Sink**
 - i. **Cache Sink Configured**: Pipelines that include Cache as sink component are displayed in drop-downs.
 - c. **Specify Lookup Key Column**: Select the lookup key column from the selected table, which you have to match or compare.
 - d. **Specify Return Columns**: It displays the cache value, select from **Yes** or **No** if you want to include the cache values or not. Enter the name for the return value column.
 - e. **Prefix Name**: Enable this option and enter the prefix to add it to the data that is matched.
 - f. **Target Field Name**: Applicable only for **Return a count of matching rows** and **Return true if matches exist, otherwise false** lookup behaviour. The column name for the returned value can be defined here.
6. Here is a configured example for Distributed Cache:
- 7.

8. Go back to the [Pipeline Configuration Page](#) to know how to add and configure Data Sink to the pipeline.

Mapping

Last updated on June 10, 2025

This window allows you to define/map the data to be displayed in specific format. If the data is loaded in any format, and you want to change the record names, configure and map them in this window. **Mapper not only changes column names, it can change data type of a column and you can add new columns with string, numerical and Boolean operators acting on incoming columns.**

 This page provides step-wise configuration about the Transformation Operator. After you configure the Transformation Operator, go back to the [Pipeline Configuration Page](#) to know how to add and configure **Data Sink** to your pipeline.

To configure Mapping, you must configure the following:

1. In Canvas, expand **Transformation Operator**. Drag and drop **Mapping** in canvas.
2. **Provide the input link to the Mapping.**
3. Click the vertical ellipsis, and select **Edit**.
4. A **Configure: Mapping** window is displayed.
Complete the following details:
 - a. **Add Field**: Click to add the new field.
 - b. **Incoming Column Name**: Enter the name of the column to be mapped.
 - c. **Incoming Data Type**: Enter the data type to be mapped.
 - d. **New Column Name**: Enter the column name to be provided for the mapped column.
 - e. **New Column Data Type**: Select the data type for the new column from the drop down.
 - f. **Apply Function**: This column computes the operations available in the configured function and the value outcome is assigned to the **New column Name** parameter. In this section, the **parameters used must be configured in the Data Transform Mapping pane. The system validates the entered function. If the function is invalid, system will not allow to save the pipeline component. In some exception case, if any existing pipeline which are running with invalid function, the state of pipeline will change to failed status.**
 - g. **Example**: If the function configured is \$multiply (\$add (const 30,const 20), salary) then the parameter **salary** must be defined. To use the existing table instances, functions or columns, click Ctrl+Space, it displays a pop up to select the configurations.
 - h. **Add All Columns**: Allows you to add all the columns from the parser into this window for mapping.
 - i. **Remove All Columns**: Click to remove all the columns in the grid.

Example 1: Concatenate Two Properties

Task:

You have an object with *FirstName*, *LastName*, and *Title* properties. You want to receive an object with *FullName* and *Title* properties instead.

```
{
  "FirstName": "Mike"
  "LastName": "David"
  "Title": "Analyst"
} → {
  "FullName": "Mick David"
  "Title": "Analyst"
}
```

1. Add a new Field.
2. Enter the new column name **FullName** for the mapped column.
3. Select the new column data type as **String** from the drop down.
4. In **Apply Function** column, type **\$concat(ARG1, ARG2)** or press Ctrl+Space to open the list of functions and select.
5. Replace ARG1 with **FirstName** and ARG2 with **LastName**, the configured apply function would be **\$concat(FirstName, LastName)**.
6. The output table will contain one new column as **FullName**.

General Functions

| Function | Description |
|----------------------|--|
| \$isNull(ARG1, ARG2) | <p>If ARG1 is null then isNull function returns ARG2. Return type should match with ARG1 and ARG2 type.</p> <ul style="list-style-type: none"> • ARG1 and ARG2 holds string, boolean, number, datetime • Return type is 'string, boolean, number, datetime' <p>Note: return type, ARG1 and ARG2 should be of same type. Example: \$isNull(firstname, lastname) if firstname is null then we will get lastname otherwise firstname.</p> |
| \$\$fileName | <p>This function is supported wherever string arguments are accepted.</p> <p>This function can be used as argument inside other functions. This function will fetch your file name that is coming from your source and return it as a string argument.</p> <p>Example:</p> <ol style="list-style-type: none"> 1. \$toLowerCase(\$\$fileName) |

| | |
|--------------------|--|
| | 2. \$substring(\$\$fileName,const 2, const 5) |
| \$\$recordType | <p>This function is supported wherever string arguments are accepted. This function will fetch your record type or record reference that you defined in Parser/Flattener and return it as a string argument.</p> <p>Example:</p> <ol style="list-style-type: none"> 1. \$toLowerCase(\$\$recordType) 2. \$substring(\$\$recordType,const 2, const 5) |
| \$\$recordNumber | <p>\$\$recordNumber is a sequence number and this function is supported wherever number arguments are accepted. This function can be used as argument inside other functions. \$\$recordNumber type returns the record number for each record. Examples:</p> <p>\$add(salary, \$\$recordNumber);</p> <p>\$abs(\$\$recordNumber);</p> <p>Return type is 'long'.</p> |
| \$\$recordAddress | <p>\$\$recordAddress gives the address for each record and this function is supported wherever number arguments are accepted. This function can be used as argument inside other functions. The first row record address will be 0. The second row address will be previous record length + previous record address + 1 and so on. Examples:</p> <p>\$add(salary, \$\$recordAddress);</p> <p>\$abs(\$\$recordAddress);</p> <p>Return type is 'long'.</p> |
| \$\$primaryKey | <p>\$\$primaryKey is a sequence number which returns the unique incremental number for each record and this function is supported wherever number arguments are accepted. This function can be used as argument inside other functions but it is unique across all the mapping components in the pipeline. The values generated are specific to a single pipeline. Examples:</p> <p>\$add(salary, \$\$primaryKey)</p> <p>\$abs(\$\$primaryKey)</p> <p>Return type is 'long'.</p> |
| \$generateId(ARG1) | <p>GenerateId Function returns unique incremental values in a sequence across pipelines. Example: \$generateId(ARG1) \$generateId(const"DMS_12") ARG1 holds string, it should always be of constant type.</p> |

| | |
|---------------------------------|---|
| | <p>ARG1 should satisfy below conditions.</p> <ol style="list-style-type: none"> 1. It should start with <code>_</code> or <code>[a-z][A-Z]</code> and it should not start with number or any other special characters. 2. The allowed characters are <code>[a-z][A-Z],_,[0-9]</code> 3. First argument should be const String and no column names are allowed. You can use either unique or duplicate sequence names in the first argument. <ol style="list-style-type: none"> 1. If you use same sequence within/across mapper/pipeline, then it gives the unique incremental values with splitting across columns. Example: <code>\$generateId(const"Mapper");</code> <code>\$generateId(const"Mapper").</code> 2. If you use unique sequence, then it gives unique incremental values for each columns. Example: <code>\$generateId(const"generate");</code> <code>\$generateId(const"Function_1").</code> |
| <code>\$hash(ARG1, ARG2)</code> | <p>Hash function returns the hash value of given variables, Constants or combination of both. Number of arguments are not restricted. You can give any number of arguments. 1. ARG1 and ARG2 hold all data types. 2. Return Type is Long.</p> <p>Example: <code>\$hash(name, const 12345)</code></p> <p>Let ARG1 be an incoming column say 'name' containing value "subex" and ARG2 be a constant data "12345".</p> <p>Output - 891529369</p> |

Lookup Functions

| Function | Description |
|----------------------------------|---|
| <code>\$lookup(ARG1,ARG2)</code> | <p>This function holds two string arguments.</p> <ul style="list-style-type: none"> • ARG1: cache name ex: const "cache_name" • ARG2: incoming column which you want to match(lookup) with cache • return type is string <p>Configured example: 1 key example: <code>\$lookup(const "cache_name",name)</code> 2 keys example: <code>\$lookup(const "cache_name", \$concat(\$concat(name, const "::"),city))</code> I need to configure lookup function in the same way.</p> |
| <code>\$get(ARG1,ARG2)</code> | <p>This function holds two arguments. This function will be used in case of I get the expected return values in cache.</p> |

| | |
|--|---|
| <p>\$constructMap(ARG1, ARG2)</p> | <p>This function holds two string arguments.</p> <ul style="list-style-type: none"> • ARG1 holds String, i.e cache column name. • ARG2 holds String, which can be constant value or the incoming column <p>Configured example: Example:</p> <ol style="list-style-type: none"> 1. \$constructMap(const "id", const 22) |
| <p>\$combineMap(ARG1,ARG2)</p> | <p>CombineMap function is used to combine two \$constructMap functions.</p> <ul style="list-style-type: none"> • ARG1 holds String. • ARG2 holds String. <p>Configured example:</p> <p>\$combineMap(\$constructMap(const "id",const22),\$constructMap(const</p> |
| <p>\$dateRangeLookup(String cacheName, DateTime searchDate, String returnValue)</p> | <p>This function holds three arguments.</p> <ul style="list-style-type: none"> • ARG1 holds String, that is cache name ex: const "cache_name" • ARG2 holds datetime data type, that is incoming date range value which • ARG3 is return value column that holds String. i.e 'cache value' column <p>Configured example: Example:</p> <ol style="list-style-type: none"> 1. \$dateRangeLookup(const "date_cache",\$dateTime(const 2001,const C 2. \$dateRangeLookup(const "date_cache",range_date1,const "id"). Here, range_date1 is the incoming column. |
| <p>\$dateRangeLookup(String cacheName, DateTime searchDate, String filter, String returnValue)</p> | <p>This function holds four arguments.</p> <ul style="list-style-type: none"> • ARG1 holds String, that is cache name ex: const "cache_name" • ARG2 holds datetime datatype, that is incoming date range column or in cache setting. • ARG3 holds String, that is constructMap function which accepts param you have two where conditions, then combineMap function can be use • ARG4 is return value column that holds String. i.e 'cache value' column <p>Configured example: Example:</p> <ol style="list-style-type: none"> 1. \$dateRangeLookup(const "date_cache",range_date1,\$constructMap(c 2. \$dateRangeLookup(const "date_cache",range_date1,\$combineMap(\$ "city",const "bengaluru")),const "id"). |
| <p>\$constructMap(ARG1, ARG2)</p> | <p>This function holds two string arguments.</p> <ul style="list-style-type: none"> • ARG1 holds String. i.e cache column name. • ARG2 holds String, which can be constant value or the incoming column |

| | |
|--|--|
| | <p>Configured example: Example:</p> <ol style="list-style-type: none"> 1. \$constructMap(const "id", const 22) |
| \$combineMap(ARG1,ARG2) | <p>CombineMap function is used to combine two \$constructMap functions.</p> <ul style="list-style-type: none"> • ARG1 holds String. • ARG2 holds String. <p>Configured example:</p> <p>\$combineMap(\$constructMap(const "id",const22),\$constructMap(const</p> |
| \$longestMatch(String cacheName, String searchValue, String validValue, String returnValue) | <p>This function holds four string arguments.</p> <ul style="list-style-type: none"> • ARG1 holds String, that is cache name ex: const "longest_Cache1" • ARG2 holds String, that is column/constant where longest match operates • ARG3 holds String, that is valid value. Longest matcher value should s • ARG4 is with return type as String. This is to which cache column it sh <p>Configured Examples: 1. \$longestMatch(const "longest_cache1", const "longest_cache1", phone_number, const "+0123456789",const "id"). Here</p> |
| \$longestMatch(String cacheName, String searchValue, String validValue, String returnValue, condition) | <p>This function helps to perform the longest prefix match for the multiple k</p> <ul style="list-style-type: none"> • ARG1 holds String, that is cache name ex: const "longest_Cache1". • ARG2 holds String, that is column/constant where longest match operates • ARG3 holds String, that is valid value. Longest matcher value should s • ARG4 is with return type as String. This is to which cache column it sh • ARG5 is condition in which one or more columns are passed for which <p>Configured Example:</p> <ul style="list-style-type: none"> • In Cache Setting, Cache configured as "lpm_match" which has Keys - Value1, Value2. • Apply function is \$longestMatch(const "lpm_match", c_name, const "c <p>Explanation:</p> <p>In Apply Function: ARG1(lpm_match) is cache name, ARG2(c_name) is se key(number2) from the cache. ARG3 ("0123456789") is the valid value. F the return value which is returned when match happens. ARG5 (c_calling with cache key (name, number1) respectively.</p> |
| \$getFilteredObject(String ARG1, String ARG2, String ARG3) | <p>This apply function is used to fetch the specific object from the Object A</p> <ul style="list-style-type: none"> • ARG1 holds Object Array as a string. • ARG2 holds the key of specific object to be searched. • ARG3 holds value of the key to be matched. |

| | <ul style="list-style-type: none">Return Type is a string value that gives a specific object based on key <p>Example: <code>\$getFilteredObject(input_array,const "parameterID",const"16778333")</code></p> <p>Explanation: Here parameterID is key whose value is 16778333, is searched to a specific object are displayed.</p> | | | | | | | | | |
|---|---|--------|------------------|-------|---|-----------------|-------|---|-----------------|-------|
| <code>\$getObject(ARG1, ARG2)</code> | <p>This apply function is used to fetch the specific object from the Object Array.</p> <ul style="list-style-type: none">ARG1 holds the Object Array.ARG2 holds the position to return the specific object. <p>Example: <code>\$getObject(onlineCreditControlRecord_creditControlRecords_array,1)</code></p> <p>Explanation: Here onlineCreditControlRecord is an Object array that has a creditControlRecord is another Object array. It has 2 object: 0 and 1. \$getObject</p> | | | | | | | | | |
| <code>\$getAllNestedValues(ARG1, ARG2, ARG3, ARG4)</code> | <p>This apply function is used to fetch the value from the Object Array.</p> <ul style="list-style-type: none">ARG1 holds the Object Array.ARG2 holds the reference object to fetch the value.ARG3 holds delimiter tag to differentiate the object value.ARG4 refers to action to be performed on reference object. It supports <p>Example: <code>\$getAllNestedValues(onlineCreditControlRecord_creditControlRecords_array,"add")</code></p> <p>Explanation: Here onlineCreditControlRecord is an Object array that has a creditControlRecord is another Object array. It has 2 object: 0 and 1. Both</p> <table><tr><th>Object</th><th>Reference Object</th><th>Value</th></tr><tr><td>0</td><td>totalOctetsUnit</td><td>20591</td></tr><tr><td>1</td><td>totalOctetsUnit</td><td>47987</td></tr></table> <ul style="list-style-type: none">If ARG4 is add, then output is 68578. It is addition of reference objectIf ARG4 is concat, then output is 20591 47987. | Object | Reference Object | Value | 0 | totalOctetsUnit | 20591 | 1 | totalOctetsUnit | 47987 |
| Object | Reference Object | Value | | | | | | | | |
| 0 | totalOctetsUnit | 20591 | | | | | | | | |
| 1 | totalOctetsUnit | 47987 | | | | | | | | |
| <code>\$multiKeyLongestMatch(cacheName, validValue, returnValue, ...searchValue Const "EXACT" condition)</code> | <p>This function helps to perform the longest prefix match by selecting multiple</p> <ul style="list-style-type: none">ARG1 holds cache name as a string.ARG2 holds valid value as a string for comparison. (Valid value can be KkLIMmNnOoPpQqRrSsTtUuVvWwXxYyZz)ARG3 holds cache's return value as a string.ARG4 holds search value as a string which are incoming columns whichARG5 holds condition for exact match which are incoming columns which | | | | | | | | | |

| | |
|--|--|
| | <p>Example: <code>\$multiKeyLongestMatch(const "multikeylpm_cache22", const "MmNnOoPpQqRrSsTtUuVvWwXxYyZz", const "market_code", imsi,msisdn)</code></p> <p>Here, "multikeylpm_cache22" is the cache configured and "market_code" is the market code. imsi and msisdn are key columns which perform LPM with cache key columns. "apn, apn_copy" are input columns which performs exact match with cache keys.</p> <p>Note:</p> <p>User can proceed without providing an exact condition in cache settings. User should provide the exact match condition and use null in the 5th argument.</p> <p>Example: <code>\$multiKeyLongestMatch(const "multikeylpm_cache22", const "MmNnOoPpQqRrSsTtUuVvWwXxYyZz", const "market_code", imsi,msisdn, null)</code></p> |
|--|--|

Boolean Functions

| Function | Description |
|---|--|
| <code>\$notEquals(Object ARG1, Object ARG2)</code> | <p>This function checks if the two arguments passed are not equal to each other. where,</p> <ul style="list-style-type: none"> • ARG1 indicates the object1 to be compared with. • ARG2 indicates the object2 to be compared to. • Return type is Boolean value. <p>Example: <code>notEquals(1,2)</code> returns TRUE.</p> |
| <code>\$regexMatches(String ARG1, String ARG2)</code> | <p>This function matches the string parameter to the given regular expression. If matched then it returns true. where,</p> <ul style="list-style-type: none"> • ARG1 is Regular Expression. • ARG2 is the string on which the regular expression must be matched. • Return type is boolean value. <p>Example: <code>regexMatches(*abc,xyzabc)</code> returns true as the abc, as the suffix in the string is matched.</p> |
| <code>\$equals(ARG1, ARG2)</code> | <p>It is a logical function that returns TRUE if two input values are equal. where,</p> <ul style="list-style-type: none"> • ARG1 and ARG2 are string, integer, long, decimal or boolean values. • Return type is boolean value. <p>Both the arguments must be of same type. Example: <code>equals(1,1)</code> is TRUE.</p> |

| | |
|---|---|
| <code>\$not(ARG1)</code> | <p>It is a logical function that returns the negation of a Boolean expression. where,</p> <ul style="list-style-type: none"> • ARG1 is a boolean value. • Return type is boolean value. <p>Example: not(true) is false</p> |
| <code>\$greaterThan(ARG1, ARG2)</code> | <p>This is a logical function that returns TRUE if the first argument (Input1) is greater than the second argument (Input2). where,</p> <ul style="list-style-type: none"> • ARG1 and ARG2 are Integer, Long or Decimal values. • Return type is Boolean value. <p>Example: greaterThan(2,1) is TRUE.</p> |
| <code>\$greaterThanEqual(ARG1, ARG2)</code> | <p>It is a logical function that returns TRUE if the first argument (Input1) is greater than OR equal to the second argument (Input2). where,</p> <ul style="list-style-type: none"> • ARG1 and ARG2 are Integer, Long or Decimal values. • Return type is Boolean value. <p>Example: greaterThan(2,2) is TRUE.</p> |
| <code>\$lessThan(ARG1, ARG2)</code> | <p>This function returns true if input1 value is less than input2 value. where,</p> <ul style="list-style-type: none"> • ARG1 and ARG2 are Integer, Long or Decimal values. • Return type is Boolean value. <p>Example: lessThan(1,2) is TRUE</p> |
| <code>\$lessThanEqual(ARG1, ARG2)</code> | <p>It is a logic function that returns TRUE if the first (Input1) argument is LESS than or EQUAL to the second (Input2) argument. where,</p> <ul style="list-style-type: none"> • ARG1 and ARG2 are Integer, Long or Decimal values. • Return type is Boolean value. <p>Example: lessThan(2,2) is TRUE</p> |
| <code>\$xor(ARG1, ARG2)</code> | <p>The Xor function returns the logical exclusive OR (Boolean) of two Boolean inputs.</p> <ul style="list-style-type: none"> • ARG1 and ARG2 holds boolean • Return type is 'boolean' <p>Example : \$xor(\$greaterThan(const 10, const 20), const true) output : true</p> |
| <code>\$or(ARG1, ARG2)</code> | <p>Or is a logic function that returns TRUE if any expression parameter set is TRUE.</p> |

| | <ul style="list-style-type: none">• ARG1 and ARG2 holds boolean• Return type is 'boolean' <p>Example : \$or(\$greaterThan(const 10, const 20), const true)</p> <p>output : true</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------------------|---|-------|-------|-------|-------|------|-----|-------|-------|------|-------|-------|-------|-------|------|------|-------|------|------|------|-------|-------|-------|------|------|------|------|-------|------|------|-------|
| <p>\$And(ARG1, ARG2)</p> | <p>The And function returns the logical And (Boolean) of two Boolean parameters.</p> <ul style="list-style-type: none">• ARG1 and ARG2 holds boolean• Return type is 'boolean', <p>Example : \$and(\$greaterThan(const 10, const 20), const true)</p> <p>output : false</p> <table><tr><th>cond1</th><th>cond2</th><th>-x</th><th>x&y</th><th>xory</th><th>xor</th></tr><tr><td>false</td><td>false</td><td>true</td><td>false</td><td>false</td><td>false</td></tr><tr><td>false</td><td>true</td><td>true</td><td>false</td><td>true</td><td>true</td></tr><tr><td>true</td><td>false</td><td>false</td><td>false</td><td>true</td><td>true</td></tr><tr><td>true</td><td>true</td><td>false</td><td>true</td><td>true</td><td>false</td></tr></table> | cond1 | cond2 | -x | x&y | xory | xor | false | false | true | false | false | false | false | true | true | false | true | true | true | false | false | false | true | true | true | true | false | true | true | false |
| cond1 | cond2 | -x | x&y | xory | xor | | | | | | | | | | | | | | | | | | | | | | | | | | |
| false | false | true | false | false | false | | | | | | | | | | | | | | | | | | | | | | | | | | |
| false | true | true | false | true | true | | | | | | | | | | | | | | | | | | | | | | | | | | |
| true | false | false | false | true | true | | | | | | | | | | | | | | | | | | | | | | | | | | |
| true | true | false | true | true | false | | | | | | | | | | | | | | | | | | | | | | | | | | |

String Functions

| Function | Description |
|-------------------------------|--|
| \$concat(ARG1, ARG2) | <p>This function concatenates the argument 1 with argument 2.</p> <ul style="list-style-type: none"> • ARG1 and ARG2 holds the string value. • Return type is a string. |
| \$substring(ARG1, ARG2, ARG3) | <p>This function extracts the substring from the string.</p> <ul style="list-style-type: none"> • ARG1 is string type • ARG2 and ARG3 holds numbers • Return type is string |
| \$padLeft(ARG1, ARG2, ARG3) | <p>This function holds three arguments. It helps to pad string1 (arg1) to the left of string2 (arg2) based on the length (arg3) defined.</p> <ul style="list-style-type: none"> • ARG1: Indicates the string parameter that must be padded to left of argument 2. • ARG2: It is string type. It indicates the string parameter to the left of it, argument 1 will be padded. • ARG3: This is an integer, which indicates the length of return string. <p>Below are a few cases for better understanding:</p> |

| | |
|--|---|
| | <ol style="list-style-type: none"> When ARG3 > combine length of ARG1 and ARG2 Consider an example <code>\$padLeft(const "John", const "Mathew", const 15)</code>. Here, string length of John is 4 and string length of Mathew is 6. Combine length of ARG1 and ARG2 is 10, and length of return string is 15. We have to pad John to the left of Mathew. So, the padded string looks like MathewMatheJohn. When length of ARG3 < combine length of ARG1 and ARG2 Consider an example <code>\$padLeft(const "John", const "Mathew", const 6)</code>. Here, string length of John is 4 and string length of Mathew is 6. Combine length of ARG1 and ARG2 is 10, and length of return string is 6. We have to pad John to the left of Mathew. So, the padded string looks like MaJohn. When length of ARG3 = combine length of ARG1 and ARG2 Consider an example <code>\$padLeft(const "John", const "Mathew", const 10)</code>. Here, string length of John is 4 and string length of Mathew is 6. Combine length of ARG1 and ARG2 is 10, and length of return string is 10. We have to pad John to the left of Mathew. So, the padded string looks like MathewJohn. When length of ARG3 < length of ARG1 Consider an example <code>\$padLeft(const "John", const "Mathew", const 3)</code>. Here, string length of John is 4, string length of Mathew is 6, and length of return string is 3. In this case, function will return ARG1. So, the padded string looks like John. |
| <p><code>\$padRight(ARG1, ARG2, ARG3)</code></p> | <p>This function holds three arguments. It helps to pad string1 (arg1) to the right of string2 (arg2) based on the length (arg3) defined.</p> <ul style="list-style-type: none"> • ARG1: Indicates the string parameter that must be padded to right of argument 2. • ARG2: It is string type. It indicates the string parameter to the right of it, argument 1 will be padded. • ARG3: This is an integer, which indicates the length of return string. <p>Below are a few cases for better understanding:</p> |

| | |
|--|--|
| | <ol style="list-style-type: none"> 1. When length of ARG3 > combine length of ARG1 and ARG2 Consider an example <code>\$padRight(const "John", const "Mathew", const 15)</code>. Here, string length of John is 4 and string length of Mathew is 6. Combine length of ARG1 and ARG2 is 10, and length of return string is 15. We have to pad John to the right of Mathew. So, the padded string looks like JohnMathewMathe. 2. When length of ARG3 < combine length of ARG1 and ARG2 Consider an example <code>\$padRight(const "John", const "Mathew", const 6)</code>. Here, string length of John is 4 and string length of Mathew is 6. Combine length of ARG1 and ARG2 is 10, and length of return string is 6. We have to pad John to the right of Mathew. So, the padded string looks like JohnMa. 3. When length of ARG3 = combine length of ARG1 and ARG2 Consider an example <code>\$padRight(const "John", const "Mathew", const 10)</code>. Here, string length of John is 4 and string length of Mathew is 6. Combine length of ARG1 and ARG2 is 10, and length of return string is 10. We have to pad John to the right of Mathew. So, the padded string looks like JohnMathew. 4. When length of ARG3 < length of ARG1 Consider an example <code>\$padRight(const "John", const "Mathew", const 3)</code>. Here, string length of John is 4, string length of Mathew is 6, and length of return string is 3. In this case, function will return ARG1. So, the padded string looks like John. |
| <code>\$toLowerCase(ARG1)</code> | This function converts the passed string argument to the lower case. <code>\$toLowerCase(ARG1)</code> Example: <code>toLowerCase(JOHN)</code> Converted string will be john. |
| <code>\$toUpperCase(ARG1)</code> | This function converts the passed string argument to the upper case. Example: <code>toLowerCase(john)</code> Converted string will be JOHN. |
| <code>\$left(String ARG1, Integer ARG2)</code> | <p>This function returns the substring from the left-most set of characters in the string. where,</p> <ul style="list-style-type: none"> • ARG1 indicates the string on which operation is performed. • ARG2 is integer that indicates the length of the string. |

| | |
|------------------------------------|---|
| | <ul style="list-style-type: none"> Return type is string. <p>Example: left(SQL Server,3) returns SQL.</p> |
| \$right(String ARG1, Integer ARG2) | <p>This function returns the substring from the right-most set of characters in the string. where,</p> <ul style="list-style-type: none"> ARG1 indicates the string on which operation is performed. ARG2 is integer that indicates the length of the string. <p>Return type is string. Example: right(SQL Server,6) returns Server.</p> |
| \$trim(ARG1) | <p>This function trims the input string of any leading or trailing space characters. where,</p> <ul style="list-style-type: none"> ARG1 is a string parameter from which space characters must be trimmed. Return type is a string value after the spaces are trimmed. <p>Example: trim(subex) returns subex.</p> |
| \$length(ARG1) | <p>This function returns the length of the string value passed. where,</p> <ul style="list-style-type: none"> ARG1 is the string parameter. Return value is an integer. <p>Example: length(subex) = 5</p> |
| \$regexReplace | <p>This function matches the string to the given regular expression and performs the replacement. Syntax -> \$regexReplace(String Arg1, String Arg2, String Arg3) Arg1 -> Regular Expression Type -> String Arg2 -> String on which regex is to be matched Type -> String Arg3 -> Replacement String Type -> String Return Type is String</p> <p>Ex: .*abc,xyzabc,def def</p> |
| \$token(ARG1, ARG2, ARG3) | <p>The Token function returns a substring of a string after splitting on a delimiter.</p> <ul style="list-style-type: none"> ARG1 and ARG2 holds string, and ARG3 holds integer Return type is 'string', <p>Note: ARG3 starts from 0 and ARG2 should be single char string. Example : \$token(const 'abc;xyz;123', const ';', const 1) output : xyz</p> |
| \$toFlag(ARG1) | <p>The ToFlag turns a Boolean input into a Y/N String.</p> |

| | |
|-----------------------------------|---|
| | <ul style="list-style-type: none"> • ARG1 holds boolean • Return type is 'string', <p>Example : \$toFlag(const true) output : Y</p> |
| \$nullString() | The NullString function returns a null. returnType: 'string' |
| \$if(ARG1, ARG2, ARG3) | <p>If function returns the second argument if the condition ARG1 is true, otherwise returns the third argument.][Return type should match with ARG2 and ARG3 type.</p> <ul style="list-style-type: none"> • ARG1 and ARG2 holds boolean, and ARG3 holds string, boolean, number, datetime • Return type is 'string, boolean, number, datetime' <p>returnType: 'string, boolean, number, datetime' Note: return type, ARG1 and ARG2 should be of same type. Example : \$if(\$greaterThan(const 10, const 20), const 10, const 20) output : 20</p> |
| \$applyStringRule(ARG1,ARG2,ARG3) | <p>This function validates the sample string element with configured match key and returns the matched vaule with that string rule.</p> <ul style="list-style-type: none"> • ARG1 holds string, (const"srsname") that is string rule set name. • ARG2 holds String, (const"matchkey name") that is the name of the matchkey names when you configure the matchkey. • ARG3 holds String, (const"name") that is on which string you want to pass the argument. <p>Example: \$applyStringRule(const"srsname", const"matchkey name",const"name") is the function.</p> <p>The below is the applied function: \$applyStringRule(const"Sample", const"Test1", 1234123987689) For example, if you define a rule like "(....123)(765)(456)". The first string entry is (....123), that means it takes first four number as any number before 123. The second string entry is (765). The third string entry is (456). Let us assume the sample number "1234123987689" for validation. In this case, If you give \$1 in output field, it returns 1234123 as it satisfies the match key format (....123).</p> |

| | |
|---------------------|--|
| | If you give \$2 , then it returns 765 as output. If you give \$3 , then it returns 456 as output. |
| \$stringToHex(ARG1) | <p>\$stringToHex function converts string to Hexa decimal values. \$stringToHex(ARG1) ARG1 holds String, it can be either constant or column values Return Type is String.</p> <p>Examples: \$stringToHex(Name) \$stringToHex(const "IDT")</p> |
| \$hexToString(ARG1) | <p>\$hexToString function converts Hexa decimal values to string value. \$hexToString(ARG1) ARG1 holds String, it can be either constant or column values Return Type is String. Example: \$hexToString(const "7375626578") will give output as "subex".</p> |

Mathematical Functions

| Function | Description |
|------------------------|---|
| \$add(ARG1, ARG2) | <p>This function adds the argument values.</p> <ul style="list-style-type: none"> • ARG1 and ARG2 holds the numbers. • Return type is a number. |
| \$subtract(ARG1, ARG2) | <p>This function returns the different between the argument values.</p> <ul style="list-style-type: none"> • ARG1 and ARG2 holds the numbers. • Return type is a number. |
| \$multiply(ARG1, ARG2) | <p>This function multiplies the argument values.</p> <ul style="list-style-type: none"> • ARG1 and ARG2 holds the numbers. • Return type is a number. |
| \$divide(ARG1, ARG2) | <p>This function divides the argument 1 with argument 2.</p> <ul style="list-style-type: none"> • ARG1 and ARG2 holds the numbers. • Return type is a number |
| \$abs(ARG1) | <p>This function returns the absolute (positive) value of a numeric value. where,</p> <ul style="list-style-type: none"> • ARG1 is Integer, Long or Decimal value. |

| | |
|---------------------|---|
| | <ul style="list-style-type: none"> Return value id Integer, Long or Decimal respectively. <p>Example: abs(-1) is 1.</p> |
| \$mod(ARG1, ARG2) | <p>This function divides one number by other and returns the remainder. Where,</p> <ul style="list-style-type: none"> arg1 and arg2 are Integer, Long or Decimal values. Return values are Integer, Long or Decimal values respectively. <p>Example: mod(10,4) is $10/4 = 2$.</p> |
| \$round(ARG1, ARG2) | <p>This function rounds a decimal to an integer or a given number of decimal places. It returns data type as integer. Supported functions are UP, DOWN, CEILING, FLOOR, HALF UP, HALF DOWN, and HALF EVEN.</p> <ul style="list-style-type: none"> ARG1 data type is integer. ARG2 data type is a string. <p>Example: \$round(const 1.6, const 'up'), the returned value here is 2. \$round(const 1.6, const 'down'), the returned value here is 1. \$round(const 1.6, const 'ceiling'), the returned value here is 2. \$round(const 1.6, const 'floor'), the returned value here is 1. \$round(const 1.6, const 'half up'), the returned value here is 2. \$round(const 1.6, const 'half down'), the returned value here is 2. \$round(const 1.6, const 'half even'), the returned value here is 2. Half Even returns the even number nearest to the first digit.</p> |
| \$pow(ARG1, ARG2) | <p>The Power function raises the first number to the power of the second number.</p> <ul style="list-style-type: none"> ARG1 and ARG2 holds number Return type is 'number', |
| \$nullDecimal() | The NullDecimal function returns a null returnType: 'decimal' |
| \$nullInteger() | The NullInteger function returns a null. returnType: 'integer' |
| \$negate(ARG1) | <p>The Negate function returns the negation of the numeric input.</p> <ul style="list-style-type: none"> ARG1 holds number Return type is 'number', <p>Example : \$negate(const 10) output : -10</p> |
| \$toString(ARG1) | <p>The toString function returns the string as a datatype after passing any datatype.</p> <ul style="list-style-type: none"> ARG1 holds integer, decimal, boolean |

| | |
|------------------------------|--|
| | <ul style="list-style-type: none"> Return type is 'string' |
| \$hexToIPv4String(ARG1) | <p>The HexToIPv4String function converts an 8 character hex string into a separated IP v4 string.</p> <ul style="list-style-type: none"> ARG1 holds string Return type is 'string' <p>Example : \$hexToIPv4String(const "4035FFFF") output : 64.53.255.255 The arg1 should be hexa string and length should be 8 chars.</p> |
| \$roundPrecision(ARG1, ARG2) | <p>This function returns the decimal value upto selected decimal place on passing a decimal value.</p> <ul style="list-style-type: none"> ARG1 holds Integer, Long, Decimal, that is the value datatype coming from parser. This can be a Constant value or Column value. ARG2 holds integer value that is number of decimal values you want to limit the ARG1 value. <p>The return type is decimal value.</p> <p>Example:</p> <ol style="list-style-type: none"> \$roundPrecision(const'125.12345', const '2'). It returns "125.12". \$roundPrecision(Salary, number of decimal values you want to limit the salary value) \$roundPrecision(25559.257222, 3). It returns "25559.257". If the salary value is 25559.25788 then it returns "25559.258" since the fourth value after decimal is greater than five. \$roundPrecision(\$divide(Salary,const12), Const3) <ol style="list-style-type: none"> \$roundPrecision(\$divide(62000,12), Const3). Here divide function performs divide operation and gives \$roundPrecision(5166.66666, Const3) and now it returns the value "5166.667". \$roundPrecision(\$divide(60000,12), Const3). Here divide function performs divide operation and gives \$roundPrecision(5000, Const3) and now it returns the value "5000" since it is a integer value. |
| \$hexToInteger(ARG1) | <p>\$hexToInteger function converts Hexa decimal values to Decimal.</p> <p>\$hexToInteger(ARG1) ARG1 holds String, Return Type is Integer. Example: \$hexToInteger(const "3ADE") Will give output as "15070".</p> |

| | |
|-----------------------|---|
| \$toLong(String ARG1) | <p>This apply function converts String to Long value. ARG1 holds String; Return Type is Long.</p> <ul style="list-style-type: none"> • Example 1: Converting constant value to long. Function: \$toLong(\$toString(const "93423443988907")) Output: 93,423,443,988,907 • Example 2: Suppose b_id is an argument which holds string value 934568. Function: \$toLong(b_id) Output: 934,568 <p>This apply function can be used with other apply functions such as add, subtract, multiply etc. Few examples are listed as below:</p> <ul style="list-style-type: none"> • \$add(\$toLong(String ARG1), const 1) • \$subtract(\$toLong(String ARG2), const 2) |
|-----------------------|---|

DateTime Functions

| Function | Description |
|--------------------------|--|
| \$addMillis(ARG1, ARG2) | This function adds milliseconds to Date Time and returns a new Date Time. Example: If the argument a holds the value 10/01/2020 10:10:10 and we want to add 1000 milliseconds for this date and time, then the apply function is addMillis(a,1000). The returned value would be: 10/01/2020 10:10:11 |
| \$addSeconds(ARG1, ARG2) | This function adds Seconds to Date Time and returns a new Date Time. Example: If the argument a holds the value 10/01/2020 10:10:10 and we want to add 2 seconds for this date and time, then the apply function is addSeconds(a,2). The returned value would be: 10/01/2020 10:10:12 |
| \$addMinutes(ARG1, ARG2) | It adds minutes to Date Time and returns a new Date Time. Example: If the argument a holds the value 10/01/2020 10:10:10 and we want to add 3 minutes for this date and time, then the apply function is addMinutes(a,3). The returned value would be: 10/01/2020 10:13:10 |
| \$addHours(ARG1, ARG2) | This function adds hours onto a Date Time and returns a new Date Time. Example: If the argument a holds the value 10/01/2020 10:10:10 and we want to add 1 hour for this date and time, then the apply function is addHours(a,1). The computed value would be: 10/01/2020 11:10:10 |
| \$addDays(ARG1, ARG2) | This function adds days to Date Time and returns a new Date Time. Example: If the argument a holds the value 10/01/2020 10:10:10 and we want to add 1 day for this date and time, then the |

| | |
|-------------------------------|--|
| | <p>apply function is addDays(a,1). The computed value would be: 11/01/2020 10:10:10</p> |
| \$addWeeks(ARG1, ARG2) | <p>This function adds weeks to Date Time and returns a new Date Time. Example: If the argument a holds the value 10/01/2020 10:10:10 and we want to add 1 week for this date and time, then the apply function is addWeeks(a,1). The computed value would be: 17/01/2020 10:10:10</p> |
| \$addYears(ARG1, ARG2) | <p>This function adds years to Date Time and returns a new Date Time. Example: If the argument a holds the value 10/01/2020 10:10:10 and we want to add 1 year for this date and time, then the apply function is addYears(a,1). The computed value would be: 10/01/2021 10:10:10</p> |
| \$secondOfMinute(ARG1) | <p>This function returns the seconds component of datetime from the argument passed. Example: If the argument a holds the value 10/01/2020 11:11:10 and we want to fetch the seconds of a minute in the argument passed, secondOfMinute(10/01/2020 11:11:10), it returns 10 seconds</p> |
| \$minuteOfHour(ARG1) | <p>This function returns the minute component of datetime from the argument passed. Example: If the argument a holds the value 10/01/2020 10:11:10 and we want to fetch the minutes of an hour in the argument passed, minuteOfHour(10/01/2020 10:1:10), it returns 10 minutes</p> |
| \$dayOfMonth(ARG1) | <p>This function returns day of the month of the configured date and time in ARG1. Example: If the argument a holds the value 10/01/2020 11:12:10 and we want to fetch the day of month in the argument passed, dayOfMonth(10/01/2020 11:12:10), it returns 10, which means 10th day of a month</p> |
| \$subtractSeconds(ARG1, ARG2) | <p>The subtractSeconds function returns the datetime after subtracting number of seconds on the given date.</p> <ul style="list-style-type: none"> • ARG1 holds datetime and ARG2 holds number. • Return type is datetime |
| \$secondsDiff(ARG1, ARG2) | <p>The SecondsDiff function returns seconds in long type after passing the datetime earlydate as first argument and datetime latedate as second argument.</p> <ul style="list-style-type: none"> • ARG1 and ARG2 holds datetime. • Return type is 'long' <p>Example: \$secondsDiff(dob, \$subtractSeconds(dob, const 1000)) output: 1000</p> |

| | |
|-----------------------------|---|
| \$todate(ARG1) | This function converts milliseconds date type to actual date type format. It returns date type in arg1. Example: In input data, you are getting a date value in the form of millisec (integer), the argument holds the value 3738383993 and we want to fetch the actual date format, then it returns 1970-02-13 11:56:23 |
| \$monthOfYear(ARG1) | <p>Returns the month of the year of the Date Time.</p> <ul style="list-style-type: none"> • ARG1 holds datetime • Return type is 'number', <p>Ex: column a: 10/01/2020 10:10:10 01</p> |
| \$hourOfDay(ARG1) | <p>Returns the hour of the day of the Date Time.</p> <ul style="list-style-type: none"> • ARG1 holds datetime • Return type is 'number', <p>Ex: column a: 10/01/2020 11:10:10 11</p> |
| \$nullDateTime() | The NullDateTime function returns a null. returnType: 'datetime' |
| \$weekOfYear(ARG1) | <p>The WeekOfYear function returns the day of the week of the year.</p> <ul style="list-style-type: none"> • ARG1 holds datetime • Return type is 'integer' <p>Note: The outcome will be in the range of 1 to 52.</p> |
| \$dayOfYear(ARG1) | <p>The DayOfYear function returns the day of the year of the Date Time.</p> <ul style="list-style-type: none"> • ARG1 holds datetime • Return type is 'integer', <p>Note: The outcome will be in the range of 1 to 366.</p> |
| \$dayOfWeek(ARG1) | <p>The DayOfWeek function returns the day of the week of the Date Time.</p> <ul style="list-style-type: none"> • ARG1 holds datetime • Return type is 'integer', <p>Note: (1 = Monday, ..., 7 = Sunday)</p> |
| \$ParseDateTime(ARG1, ARG2) | <p>The ParseDateTime function return datetime type after passing the string type argument.</p> <ul style="list-style-type: none"> • ARG1 and ARG2 holds string |

| | |
|--------------------------------|--|
| | <ul style="list-style-type: none"> Return type is 'datetime' <p>Example: \$ParseDateTime(const "dd/MM/yyyy HH:mm:ss",const "17/03/2002 10:12:13") output: 17/03/2002 10:12:13</p> |
| \$millisOfSecond(ARG1) | <p>millisOfSecond returns the millis component of date time.</p> <ul style="list-style-type: none"> ARG1 holds datetime Return type is 'integer' <p>Example : \$millisOfSecond(\$now()) output : 890</p> |
| \$time(ARG1, ARG2, ARG3, ARG4) | <p>The Time function builds a time from its component parts.</p> <ul style="list-style-type: none"> ARG1, ARG2, ARG3, and ARG4 holds integer Return type is 'datetime' <p>Note: The given time is added to 1/1/1970 by default. Example : \$time(const 10, const 20, const 40, const 25) output : 1970-01-01 10:20:40:025</p> |
| \$subtractDays(ARG1, ARG2) | <p>The subtractDays function returns the datetime after subtracting number of days on the given date.</p> <ul style="list-style-type: none"> ARG1 holds datetime and ARG2 holds number Return type is 'datetime' |
| \$buildDateTime(ARG1, ARG2) | <p>The BuildDatetime function builds a date from a date and time component.</p> <ul style="list-style-type: none"> ARG1 and ARG2 holds datetime Return type is 'datetime' <p>Example : \$buildDateTime(\$now(), \$time(const 10, const 20, const 40, const 25)) (18-11-2020 10:30:20 010 , 1970-01-01 10:20:40:025) output : 18-11-2020 10:20:40:025</p> |
| \$dateTime(ARG1, ARG2) | <p>The BuildDatetime function builds a date from a date and time component.</p> <ul style="list-style-type: none"> ARG1 holds integer and ARG2 holds datetime Return type is 'datetime' <p>Functionality: if days_of_year is less than or equal to the no of days in datetime argument, then the days_of_year is added to starting of the year. Exmaple: \$dateTime(const 120, dob) If dob is 1990-10-20, here 120 is less than 292 (no of days till 1990-10-20) Output: 1990-04-30 if days_of_year is greater than no of days in</p> |

| | |
|---|---|
| | <p>datetime argument, then the days_of_year is added to starting of the previous year. Exmaple: \$dateTime(const 275, dob) If dob is 01/10/2006, here 275 is greater than 274</p> <p>Output: 02/10/2005</p> |
| <p>\$dateTime(YEAR, MONTH, DAY, HOUR, MINUTE, SEC, MILLI)</p> | <p>The BuildDatetime function builds a date from a date and time component. returnType: 'datetime', arguments: [{</p> <pre> text: 'YEAR', type: 'integer' }, { text: 'MONTH', type: 'integer' }, { text: 'DAY', type: 'integer' }, { text: 'HOUR', type: 'integer' }, { text: 'MINUTE', type: 'integer' }, { text: 'SEC', type: 'integer' }, { text: 'MILLI', type: 'integer' }] </pre> <p>Example : \$dateTime(const 2005, const 12, const 23, const 10, const 30, const 50, const 609) Output: 2005-12-23 10:30:50 609</p> |
| <p>\$parseDuration(ARG1, ARG2)</p> | <p>The ParseDuration function returns seconds in decimal type after passing the string format as first argument and string duration as second argument.</p> <ul style="list-style-type: none"> • ARG1 and ARG2 holds string • Return type is 'decimal' <p>Example : \$parseDuration(const "HH:mm:ss.SSS", const "00:00:29.101")</p> <p>output : 29.101</p> |

| | |
|------------------------------------|--|
| <p>\$roundDateTime(ARG1, ARG2)</p> | <p>The roundDateTime function returns Date in Datetime type after passing the datetime format as first argument and integer type as second argument. If second argument is greater than by millisecond of first argument then it increase second of first argument by and function returns dd/MM/yyyy HH:mm:ss format 1 else function returns in dd/MM/yyyy HH:mm:ss format.</p> <ul style="list-style-type: none"> • ARG1 holds datetime and ARG2 holds number • Return type is 'datetime' <p>Example : \$roundDateTime(dob, const 10) (where dob= 12/11/2007 12:11:30:300) 300 > 10 5 < 10 output : 12/11/2007 12:11:31</p> |
| <p>\$addMonths(ARG1, ARG2)</p> | <p>The AddMonths function adds months to a Date Time and returns a new Date Time.</p> <ul style="list-style-type: none"> • ARG1 holds datetime and ARG2 holds number • Return type is 'date' <p>Example: \$addMonths(\$now(), const 2)</p> |
| <p>\$date(ARG1, ARG2, ARG3)</p> | <p>The Date function builds a date from its component parts, those being Year, Month and Day.</p> <ul style="list-style-type: none"> • ARG1, ARG2, and ARG3 holds number • Return type is 'date' <p>Example: \$date(const 2020, const 11, const 21)</p> |
| <p>\$millis(ARG1)</p> | <p>The Millis function returns the number of milliseconds since 01/01/1970. This is useful for doing arithmetic between two or more dates</p> <ul style="list-style-type: none"> • ARG1 holds datetime • Return type is 'long' • \$millis(date_of_birth) <p>Example: \$date(const 2020, const 11, const 21)</p> |
| <p>\$now()</p> | <p>The Now function returns the current data time when called.</p> <ul style="list-style-type: none"> • Return type is 'date' |
| <p>\$parseDateTime(ARG1, ARG2)</p> | <p>The ParseDateTime function parses a date time from a String input provided with a specified Format</p> <ul style="list-style-type: none"> • ARG1 and ARG2 holds string • Return type is 'date' |

| | |
|---|--|
| | <p>Example: <code>\$parseDateTime(const 'dd/MM/yyyy HH:mm:ss',const '17/03/2002 10:12:13')</code></p> |
| <code>\$seconds(ARG1)</code> | <p>The Seconds function returns the second component of a date-time input.</p> <ul style="list-style-type: none"> • ARG1 holds datetime • Return type is 'long' <p>Example: <code>\$seconds(date_of_birth)</code></p> |
| <code>\$year(ARG1)</code> | <p>The Year function returns the year component from the provided DateTime value.</p> <ul style="list-style-type: none"> • ARG1: A Datetime Argument • Return Type: long <p>Example: <code>\$year(date_of_birth)</code></p> |
| <code>\$formatDateTime(ARG1, ARG2)</code> | <p>This function allows you to accept the column that has date format of any type and converts into the format that you pass on second argument.</p> <ul style="list-style-type: none"> • ARG1 holds DateTime. • ARG2 holds String. That is into which format you want to convert the datetime. • Return Type is String. <p>ARG1 should be Column Type whereas ARG2 can be of Column or Constant Type.</p> <p>Example: <code>\$formatDateTime(Incoming Parser Column of Type DateTime, Incoming Parser Column which hold new date format)</code> and <code>\$formatDateTime(Incoming Parser Column of Type DateTime, Const"dd.mm.yyyy")</code></p> <p>1. columnToBechanged,newFormat 1997-03-23 12:05:44,dd.MM.yyyy 1997-03-23 12:05:44,dd-MMM-yyyy</p> <p>Configured Example: <code>\$formatDateTime(existing_date_format_column_name,const 'new_date_format')</code></p> <p>1. <code>\$formatDateTime(ParserColumn,const "dd.MM.yyyy")</code> 2. <code>\$formatDateTime(ParserColumn1,ParserColumn2)</code> It returns "23.03.1997" as the return format is "dd.MM.yyyy". In the above example, if you give column name in ARG2, It returns the new date format coming from parser.</p> |

Configured Example:

Global Variables

Global Variables allows you to make use of the column data of one record reference in a different record reference as required. Here ASCII Parser is used in parser configuration. ASCII Parser identifies different record references from grouped set of references from the source. Global Variables can be defined in two ways:

1. "Record_Reference.column_name".
Example: "moc.name"
2. Using incoming column names in existing mapping functions.
Example: Concat(moc.name,const "Subex") where moc.name is column coming from Ascii Parser.
3. The configured global variable can be used in apply functions in mappings tab and output can be seen in Sink components.

Pre requisite ASCII Parser configuration for Global Variables

1. The **File Structure Definition** in ASCII parser is of two record references (taken as example) as shown in below figure.
2. The Record structure Definition can be defined as shown in below figure.
3. For more information to configure ASCII parser, see [Configuring ASCII Parser](#).

Configuring Mapping for Global Variables

To configure Mapping component, you must configure the following:

1. In Canvas, expand **Transformation Operator**. Drag and drop **Mapping** in canvas.
2. **Connect Parser** to **Mapping** component. The **Select Schema** drop-down window is displayed. Select the Schemas and click **Save** to save schema selection.
3. Click the vertical ellipsis and select **Edit**.
4. The **Configure: Merge Mapper** window is displayed when we connect multiple schemas from parser to mapping component. We have **Global Variables** configuration and **Mappings configuration** tab.
5. Click the **Add Field** icon to configure global variables. You can click **Add Field** further to add as many global variables required.
6. **Configure the following fields under Global Variable** tab and click **Apply**.
 - a. **Variable Name**: Enter the variable name.
 - b. **Data Type**: Select the data type from **Select Data Type** dropdown.
 - c. **Initial Value**: Global Variable holds this value when no data is passed from incoming Column. (Example - if incoming record does not have value for the column that is used in Global Variable configuration then Initial Value will be assigned to it).

- d. **Apply Function:** Incoming Parser columns can be used here as **Record_Reference.Column_Name** along with Apply functions.
Example - `$$concat(moc.name, const "Subex")`.
7. Click Apply, the **Mappings** tab is displayed. In order to map the value of global variable to required record reference, we create a new column and assign global variable to it.
8. The expanded window is displayed. Click **Add Field**.
9. A new row is added. Configure the following fields and click **Apply**.
 - a. **New Column Name:** Enter a new name to the column in which the global variable is to be assigned.
 - b. **New Column Data Type:** Select the data type of the new column from dropdown.
 - c. **Apply Function:** Enter the global variable name. Global Variable can also be given via apply functions here.
 - d. Below is the configured example using global variable name in apply function:
 - e. Below is the configured example where global variable is used in apply function.
10. Go back to the [Pipeline Configuration Page](#) to know how to add and configure Data Sink to the pipeline.

Parser

Overview

Last updated on May 12, 2023

This operator allows you to convert the input data to ASCII, ASN.1, XLS/XLSB/XLSX, CSV, and Binary format. Following are the types of parse supported:

- [Data File Parser\(ASCII\)](#)
- [Data File Parser\(ASN.1\)](#)
- [Data File Parser\(Binary\)](#)
- [Data File Parser \(CSV\)](#)
- [Data File Parser \(XLS/XLSX/XLSB\)](#)
- [Data File Parser \(XML\)](#)

If you perform any alter table operations like adding a new column, deleting a column, changing the column name and so on in the parser, then you can see the modified changes in the Table Details (incoming columns) of the sink that is connected to the parser. All the sinks that are connected to the parser are affected.

ASCII Parser

Last updated on May 19, 2025

ASCII parser is used to read and interpret ASCII-encoded data. ASCII is a character encoding standard that represents text characters using a 7-bit binary code. ASCII Parser is used to parse the CSV and text files which columns are separated by Delimiter of any character, Delimited Tagged and Fixed Length records. It can also parse records to particular schema using identifier when having multiple schema's.



This page provides step-wise configuration about the Transformation Operator. After you configure the Transformation Operator, go back to the [Pipeline Configuration Page](#) to know how to add and configure [Data Sink](#) to your pipeline.

To configure the ASCII Data File Parser:

1. In Canvas, expand **Transformation Operator**. Drag and drop **Parser** in canvas.
2. Provide the **Data Source** input to the **Parser**.
3. Click the vertical ellipses and select **Edit**.
4. A **Select Parser File Type** window is displayed. Select the file type - **Data File Parser (ASCII)**.
5. Considering the ASCII format to define the Parser and grammar, you must configure [File Structure Definition](#), [Record Structure Definition](#), and [Record Integrity](#). For more information on configuring Data File Parser, see [Configure Data File Parser](#).
6. After the **File Structure**, **Record Structure Definition** and **Record Integrity** is configured, click **Save Parser**.

Properties and Actions in Configure Data File Parser(ASCII) Window

Configure ASCII Data File Parser (ASCII) window has the following properties and tabs:

1. **File Structure Definition:** This tab allows you to define the file structure definition. All the configurations in this window defines the structure of the file to be parsed. For more information on the properties, actions and attributes, see [File Structure Definition](#). For more information on configuration, see [File Structure Definition Configuration](#).
2. **Record Structure Definition:** This tab allows you to define the record structure definition. This tab allows you to define the data field in the records for the file loaded. Example: If we have any records with the following fields: ID, name, location, contact number. We have to define the fields in this window to parse the data in the specified format. For more information on configuration, see [Record Structure Definition](#).
3. **Record Integrity:** This tab allows you to perform the missing record sequence check and duplicate record from the file. For more information, see [Record Integrity](#).

Common Actions

Common actions in **File Structure Definition** and **Record Structure Definition** window:

1. **Edit:** Allows you to edit the entry.
2. **Delete:** Allows you to delete the record.
3. **Copy:** Allows you to copy the record.
4. **Paste:** Allows you to paste the copied entry.
5. **Cut:** Allows you to cut the record.
6. **Add Converter:** This option helps you in converting the type of selected record.
7. **Move Up:** Allows you to move the record upwards in the structure definition.
8. **Move Down:** Allows you to move the record downwards in the structure definition.

File Structure Definition

Component Panel

Component Panel has the attributes that are used to define the logic and provide instructions to the parsing engine on parse start or run.

1. **File Start Point:** File Start Point provides the instructions for parser about where to start and how to run through parsing the file. To configure the File Start Point, Drag and drop the File Parser elements on to the right file structure space.
2. **Logic Attributes:** Logic attributes are used to define the structure of a data file and/or record structure that is/are used to control the flow of data through the parser by implementing the following controls. It allows the definition of header records, trailer records, block structures, record structures and field structures to be configured. Below Logic Attributes are available:
 - a. **Sequence:** Adding a Sequence allows you to control the flow of the parser by adding either nested Record References or other Logic commands. It follows top to bottom flow in the nested tree. A sequence would be the highest operator of a logic tree. Sequences are used within File Parser, Blocks or Records with Sub Records. A sequence is executed only once.
 - b. **Sequence Of:** A Sequence Of repeats until all data records of a particular record reference are parsed and exits once the interpreter registers that no data record has been parsed.
 - c. **Choice:** Choice acts in a manner similar to a Sequence and is typically used for Binary Data Definition Files. In the Choice Logic Element, the parser works through the Record References to find a matching Record Reference, before moving to the next Record Reference.
 - d. **Record Reference:** Record References can be added to a Sequence, Sequence Of or Choice Logic Operators and refers the parser to a specific Record Type entry within the parser.
 - i. *Steps to Add Record Reference*

1. Drag and drop **Record Reference** into the grid, following configuration window is displayed.
2. To use the existing record reference, select the existing record definition from the drop-down and click **Add**.
3. To create a new Record Reference, click **Add New**. **Add Record Reference** window is displayed.

Configure the following:

- a. **Record Name:** Enter the record name for which you have to define the structure.
- b. **Record Type:** Select the record type to handle the selection of records for parsing. Based on the selected record type, the configurations vary.
 - i. **Delimited:** If selected, the records are selected to parse based on the selected delimiters. Example: Comma separated values.
 - ii. **Fixed Length:** If selected, the records are selected to parse based on the length configured. Example: String of length 15
 - iii. **Delimited Tagged:** If selected, the records are selected to parse based on the delimited tags. Example: The delimiter is comma separated value and the tag configured is 1, it parses the data based on that values.
- c. **Tag Delimiter:** This parameter is applicable when record type is **Delimited Tagged**. Select the tag delimiter from the list, for the parser to read the data. You can also add your own tag delimiters into the list using **Create Tag Delimiter**.
For Example: Consider that data has 1 ID, 2 Name, 3 address. The tag delimiter to be configured for a record is space.
- d. **Parse Quote Characters:** This parameter is used to read the data while parsing. Select the parameter from the dropdown list as per business requirement:
 - i. **Single Quote:** Single Quote refers reading incoming data having strings with double quotes while parsing. **For Example:** Suppose the incoming data has name and company name like, "John Mathew,Albert", and Network Pvt Ltd respectively. If you select Single Quote, then parser will read John Mathew,Albert as single field and will parse the data to name field. In absence of quotes parser will read "John Mathew" as name field and "Albert" as company field.
 - ii. **Double Quote:** Double Quote refers reading incoming data having strings with single quotes while parsing. **For Example:** Suppose the incoming data has name and company name like, 'John Mathew,Albert', and Network Pvt Ltd respectively. If you select Double Quote, then parser will read John Mathew,Albert as single field and will parse the data to name field. In absence of quotes parser will read "John Mathew" as name field and "Albert" as company field.
 - iii. **Both Quotes:** Both Quotes refers reading incoming data having strings with single and double quotes while parsing. In this, both single quote(') and double quote(") is considered as record for parsing data. **For Example:** Suppose the in-

coming data has name and company name like, 'John "Mathew', and Network Pvt Ltd respectively. If you select Both Quotes, then parser will read 'John "Mathew' as single field and will parse the data to name field.

Note:

- Both Quote is applicable to Record Type "Delimited".

e. **Field Delimiter:** Select the required field delimiter from **Field Delimiter** dropdown. In the **Field Delimiter** dropdown, you can enter the delimiter to separate the field values.

For Example: If the delimiter is "comma", then the field values of imported csv file are separated with delimiter 'comma'.

f. **Record Delimiter:** Select the required record delimiter from **Record Delimiter** dropdown.

- The record delimiter allows you to select the terminator to separate the records. **Terminator acts as record separator when multiple records are available.** You can create terminator by selecting Create Record Delimiter option.
- Enter any special character as dropdown item that acts as terminator.** Here '@' acts as terminator.
- Select the configured terminator using the **Record Delimiter** dropdown and click **Done**.

g. **Generate:** If enabled, it allows you to parse the data.

ii. *Configured Example for Record Reference -Delimited*

Consider an example of input data is as mentioned below:

1=John, 2=321A, 3=Bengaluru
1=Thomas, 2=322A, 3=Delhi
2=32A, 3=Chennai
3=Mumbai

Before we parse this data, we must define the record reference for a Parser to read it. From the data, we see that it has 4 records. Records have 1-3 columns/content categories separated with comma. Each column has ID and respective value assigned. Configure the following:

- Select the record type Delimited Tagged.
- Select the Tag delimiter "=".
- Select the delimiter as comma.
- Configured Example:**

iii. *Configured Example for Record Reference - Fixed Length*

Consider an example of input data is as mentioned below: david2601011990 Before we parse this data, we must define the record reference for a Parser to read it. From the data, we see that it has 3 records; name, age, and date of birth. Configure the following:

1. Select the record type as Fixed Length.
2. Select the Parse Quote Character as Single Quote or Double Quote as required.
3. Select the length count as 15.
4. **Configured Example:**

File Structure Definition Configuration

This tab allows you to define the file structure definition. All the configurations in this window defines the structure of the file to be parsed. For more information on how to create new file structure definition, see [Create New FSD](#).

In File Structure Definition tab you will find below functionality:

1. **Skip Header and Skip Footer:** This feature allows you to skip the required number of rows from available records. Steps to Skip Header and Footer are listed below:
 - a. From the **Configure Data File Parser(ASCII)** window, select the **Skip Header** and **Skip Footer** check boxes available.
 - b. Enter the required number of rows to skip in **Count of Header Rows to Skip** and **Count of Footer Rows to skip** text boxes respectively.
 - c. In the above example, as configured, the first 2 header rows of available records are skipped and the last 5 footer rows of available records are skipped.
2. **ASCII Revamp:** For faster performance process, turn on the toggle bar available. By default the toggle bar is turned On.

Record Structure Definition

This tab allows you to define the record structure definition for records in the file loaded. Example: If we have any records with the following fields: ID, name, location, contact number, and so on; To parse the data in the specified format, we have to define the fields in this tab in the specific structure.

Note:

- You can also create multiple record reference without adding identifier for record type - 'Delimited'.
- To parse the record, select the check box associated with the **Generate** for any record. If not, then data will not be parsed.

In Record Structure Definition you need to configure with Field, Identifier and Converter. Click to know how to add the Field / Identifier/ Converter:

1. *Add Field*

Complete the following to add fields:

1. Click **Add Field or Identifier**, and select **Field**.
2. A **New Data Field** window is displayed.
3. Complete the below details:
 - a. **Field Name**: Enter the name for the unique value in the records.
 - b. **Encoding**: Select the encoding formats from the drop down. It will help you to parse the data in the selected format. Select the encoding technique from dropdown list. Available options are:
 - i. **ASCII**: Allows you to encode the file into ASCII format to parse the data. This technique does not support accented letters used in Western European Language such as é, ñ, ç.
 - ii. **Windows-1252**: This technique is also known as CP1252. It supports characters as per ASCII format and accented letters used in Western European Language such as é, ñ, ç. When this encoding technique is selected, Validation message pops up. Once saved, all record references and columns are encoded into Windows1252 format.
 - c. **Trim White Space**: Enable this option to trim or eliminate the white spaces in the record while parsing.
 - d. **Trim Quotes**: Enable this option to trim or eliminate the quotes in the record while parsing. Example: If the record to be parsed is name , after parsing, it will look like name.
 - e. **Trim White Space Inside Quotes**: Enable this option if you want to trim the white spaces inside the single quotes of any record. Example: If the record to be parsed is 'na me ', after parsing, it will look like 'name'.
 - f. **Pad Character**: Enter the character to be trimmed in the prefix while parsing. It deletes all the occurrences of configured prefix. Example: If the data to be parsed is xxName, and we have to trim the data by eliminating x, configure the pad character as x.
 - g. **Term Character**: Enter the character to be trimmed in the suffix while parsing. It deletes all the occurrences of configured suffix. Example: If the data to be parsed is Nameyy, and we have to trim the data by eliminating y, configure the term character as y.
 - h. **Null Value**: Configure any parameter that you want to replace with the null value. Example: If the configured value is **Mary**, all the occurrences of **Mary** in the data, will be replaced with Null.
4. Click **Done**.

2. *Add Identifier*

If there are multiple record references, then to identify that, you must configure the identifiers.

Steps to add identifiers are listed below:

1. Click **Add Field or Identifier** and select **identifier**.

2. A New Data Identifier window is displayed.
3. Complete the below details:
 - a. **Offset name:** Enter the numeric value for the offset.
 - b. **Length:** Enter the length.
 - c. **Trim White Space Inside Quotes:** Enable this option if you want to trim the white spaces inside the single quotes of any record. Example: If the record to be parsed is 'na me ', after parsing, it will look like 'name'.
 - d. **Trim Quotes:** Enable this option to trim or eliminate the quotes in the record while parsing. Example: If the record to be parsed is name , after parsing, it will look like name.
 - e. **Trim White Space:** Enable this option to trim or eliminate the white spaces in the record while parsing.
 - f. **Pad Character:** Enter the character to be trimmed in the prefix while parsing. It deletes all the occurrences of configured prefix. Example: If the data to be parsed is xxName, and we have to trim the data by eliminating x, configure the pad character as x.
 - g. **Term Character:** Enter the character to be trimmed in the suffix while parsing. It deletes all the occurrences of configured suffix. Example: If the data to be parsed is Nameyy, and we have to trim the data by eliminating y, configure the term character as y.
 - h. **Id Value:** Configure any parameter that you want to replace with the Id value. Example: If the configured value is Mary, all the occurrences of Mary in the data, will be replaced with the entered value.
4. Click **Done**.

3. *Add Converter*

To add a Converter, hover the mouse cursor over the record row, you can see **Add Converter** option at the right end of the respective record.

Complete the following to add converter:

1. Click to add New Converter and select the data type from the drop-down. Available options are; Datetime, Decimal, Boolean, Integer, and Long.
2. Click **Done**.
4. You can use the *common actions* available in the window.
5. **Enable Validation:** It allows you to validate the input file with the parser schema defined. It helps to avoid situations such as data contamination with incorrect records or parser failure. Select the checkbox to perform the validation. When checkbox is selected, below options are displayed:
 - a. **Skip Record:** It allows you to skip the records those are not matching the Record Structure definition.
 - b. **Skip File:** It allows you to skip the complete input file if it is not matching with Record Structure definition.

6. Click **Save Parser**.
7. If you wish to configure for missing file sequence and duplicate record check, Click [Record Integrity](#) tab.
8. **Configured Example:**

Note:

Error Handling: This feature will reject the wrong(data type mismatch) or corrupted records in parser and process correct files. If one of the data record contains an error value, then the system skips that error and continues to fetch the subsequent records without interrupting the current record fetching process.

Consider below records example:

| Name | Age | DOB |
|------|-----|------------|
| abc | 32 | 29/08/1989 |
| efg | 20 | 09/12/2002 |
| abc | 32 | xyz |
| abc | 32 | 23/08/1989 |

In the 3rd record, date of birth (dob) is in the wrong data type, so the system rejects that record and fetches the other subsequent records.

Record Integrity

This tab allows you to perform the missing record sequence check and duplicate record from the file. Complete the configuration:

1. **Record Sequence Check:** This check allows you to find the missing record sequence from the file. Select the **Enable** radio button to configure missing record sequence check. For more information, see [Missing Record Sequence Check](#).
2. **Duplicate Record Check:** This check allows you to find the duplicate records from the file. For more information, see [Duplicate Record Check](#).

Create File Structure Definition

1. When you drag and drop the **file start point** elements, you must place it inside the **Add File Start Point** frame. You will see the enabled placeholder while you drop the element.
2. When you drag and drop the **logic** attributes, you must place it inside the **File Start Point** frame (Example: File Parser). You will see the enabled placeholder while you drop the element.
3. In **File Structure Definition** tab, click **Add File Start Point** and select [File Start Point](#).

or

You can drag and drop the File Start Point element into [File Structure Definition\(Ascii\)](#) frame.

- a. Drag and drop the logic on to the file start point. For more information on the logic attributes, see [Logic Attributes](#).
 - b. You can also add the logic using **Add Logic** option associated with each attribute.
 - c. On mouse hover of subjective root element, you can see **Add Logic option at the right end of** respective logic component.
 - d. Click plus icon, supported type of logic list is displayed.
 - e. If **Sequence** is selected, the next logic that can be added is **Record Reference**.
 - f. If **Sequence of** is selected, the next logic that can be added are **Choice and Record Reference**. Enter the required configurations for the selected operator.
 - g. You can use the common actions available at the top left side of the window. For more information go to [common actions](#).
4. **Configured Example:** Here is an example of File structure defined with the two record references demo and test.

Missing Record Sequence Check

In Data Management pipeline, after loading the data, if any records are missed in between file processing, record sequence check feature identifies the missing records from the input files and generates an alert notification for missing record sequence. You can view the triggered alert in Triggered alerts screen.

To configure Record Sequence:

1. From the configure: **Data File Parser** Screen, Click the **Record Integrity** tab. By default, record sequence check option is disabled.
2. Click **Enable**. The Record Sequence window will be expanded automatically.
3. Configure the following details and click **Save**.
 - a. **General Details**
 - i. **Integrity Name:** This is a general name given to integrity file. Enter a name for the Integrity check.
 - b. **Record Sequence Check Configuration**
 - i. **Select Column:** This provides the list of columns for selection. Click the **Select Columns** dropdown and select the required column. **Select Column** dropdown displays only integer and long data type columns. After selecting the **Select Column**, you can not edit/modify the selected column in record structure definition.

Note:

- In multiple record reference scenario, the dropdown for the record sequence identifier column will display only the columns that are present in all schema.

- ii. **Record Start Number:** Record start number is the sequence number from where the record sequence need to start.
 - iii. **Record End Number:** Record end number is the sequence number where the record sequence ends.
 - iv. **Wait Time:** Wait Time is the time duration after loading a file that a system holds before raising an alert. Enter the required wait time.
 - v. **Time Frame:** Time Frame quantifies the units of wait time. Click the time frame dropdown and select the required time frame(**seconds/minutes/hours**).
- c. **Configured Example:**
- i. **In the above configured example, if you configure Record Start number as 10 and Record End number as 20 and your input file has the records only from 10 to 15.** In this scenario, as per record sequence configuration, the records must exist from 10 to 20 but they exist only from 10 to 15 and the remaining records from 16 to 20 are missing. Since the wait time for the record sequence is configured as one hour, after loading the file, system waits for one hour and generates an alert for the missing record sequence.
 - ii. If any missing record sequence is detected, an alert is generated and you can view the alert in **Triggered Alerts** screen as shown below.

Duplicate Record Check

In Data Management Pipeline, while loading the data, if any duplicate record is found while processing the file, duplicate record check feature identifies the duplicate records from the input files and generates an alert notification.

To view the triggered alert either Click Bell icon in the Pipeline Repository or visit alert screen. For more information on Triggered Alerts, see [Triggered Alerts](#).

To configure, click [Click Here](#). New Row is added.

Note:

- Incoming columns should have Timestamp datatype to check and generate alert in case of duplicate records are identified. For more information on how to create alert, see [Create Alert](#).
- It will process -1 day records. **For example:** Suppose System Date is 30th September'24 15:30:00 and you perform duplicate record check, records are checked from 29th September '24 00:00:00 till 30th September'24 15:30:00.

Configure below details and Click **Save**.

1. **Record Type:** This field displays the name of incoming file. In case incoming file has multiple worksheets, worksheet names are displayed in the dropdown list. Select the worksheet in which duplicate record should be performed.
2. **Timestamp Columns:** This field displays the date columns for checking the duplicate record. Select the date column from the dropdown list.

3. **Columns:** This field displays the list of columns on which duplicate check to be performed. Select the columns from the dropdown list. You can select multiple columns.
4. Click **Add Row** when you want to perform duplicate record check on multiple worksheet.
5. Click **delete** icon to remove the rows.
6. **Configured example:**
7. If any duplicate record is detected, an alert is generated and you can view the alert in **Triggered Alerts** screen.

Note:

Click on URL link displayed in alert message to view the duplicate record details.

Go back to the [Pipeline Configuration Page](#) to know how to add and configure Data Sink to the pipeline.

ASN.1 Parser

Last updated on February 5, 2024

ASN.1 parser reads ASN.1 encoded data structure and converts it into a format that can be easily analyzed and processed. An ASN.1 parser allows you to read and interpret ASN.1-encoded data by understanding the structure defined in ASN.1 notation. It enables you to decode and extract information from ASN.1-encoded messages, handle data types defined in the ASN.1 specification, and perform various operations on the parsed data. It provides functionality such as parsing, decoding, validating, extracting, encoding ASN.1 file. In HyperSense, ASN1 parser parse the records from ASN1 binary files using grammar file.



This page provides step-wise configuration about the Transformation Operator. After you configure the Transformation Operator, go back to the [Pipeline Configuration Page](#) to know how to add and configure **Data Sink** to your pipeline.

To configure the ASN.1 Data File Parser:

1. In Canvas, expand **Transformation Operator**. Drag and drop **Parser** in canvas.
2. Provide the **Data Source** input to the **Parser**.
3. Click the vertical ellipsis and select **Edit**. A **Select Parser File Type** window is displayed.
4. The **Configure: Data File parser (ASN.1)** is displayed. You can select the required grammar file type (**ASN.1 Grammar** or **Decoder XML**) and click **Upload** to upload the grammar file.
5. **Browse and select the grammar file for upload.** Click the **Grammar** tab row to check the data.

6. The data inside the grammar file is displayed.
7. If you select **Decoder XML** while selecting the required grammar file, click **Upload** to upload the **Decoder XML** grammar file.
8. Browse and select the grammar file for upload. After uploading the grammar file, click **Generate**.
9. If the data is loaded successfully, the system redirects to the next tab **Structure Definition**. It creates the json structure to the file. Click **Save**.
10. To check this functionality, you can also Upload a sample ASN1 file or Decoded XML file and verify. For information on uploading sample data, go to [Upload Sample File](#).
11. Connect the output of Parser to the Json flattener. For more information, refer to [Flattener](#).

Properties and Actions in Configure Data File Parser(ASN.1) Window

Configure ASN.1 Data File Parser (ASN.1) window has the following properties and tabs:

1. **Grammar:** This tab allows you to upload, download grammar file. All the configurations in this window defines the structure of the file to be parsed. For more information on configuration, see [Grammar](#).
2. **Structure Definition:** This tab allows you to upload sample data. For more information on configuration, see [Structure Definition](#).

Common Actions

Common actions in **Grammar** and **Structure Definition** window:

1. **Upload:** Allows to upload any grammar file.
2. **Download:** Allows to download the modified grammar file.
3. **Refresh:** Allows to revert the modifications done on the grammar file.

Grammar

This tab displays the uploaded grammar file data. As soon as the grammar file is uploaded, it will check for any exception or error in the file and displays in the error log panel.

1. The grammar file is uploaded on the initial configuration window of ASN.1 Parser. You can also upload the grammar file by clicking on the upload icon on this window.
2. Click **Generate** to check for any exception or error in the file. While decoding the grammar file to tree structure, if there is any error or issue in the file, it will prompt in the error log panel.
3. In case if any modifications are required in the grammar file, then you can perform the edit on this interface and click **Save**.
4. If there is any syntax or exception error in the grammar file, then it is displayed in the error panel.

Structure Definition

This tab displays the tree structure definition for grammar in the file loaded. The output data of this parser will be in JSON format, and this output will be the feed as an input to the field flattener. To verify if the parsing is done in proper format or not, you have to upload sample file with specific structure in this window.

Upload Sample File

This allows to validate in GUI whether the sample file is getting parsed or not.

Note:

The ASN.1 Parser file should be inline with the Grammar file upload in the Grammar tab.

1. Click **Upload Sample File** to browse and select sample file.
2. Raw data is displayed in the form of hexadecimal.
3. **After uploading, click Parse Data.** The actual structure of the data is in the **Configuration** panel. In the **Decoded** panel, the parsed data is on the left side and the raw data in on the right side.
4. Go back to the [Pipeline Configuration Page](#) to know how to add and configure Data Sink to the pipeline.

Note:

Skip Error: This feature will reject the wrong(data type mismatch) or corrupted records in parser and process correct files. If one of the data record contains an error value, then the system skips that error and continues to fetch the subsequent records without interrupting the current record fetching process.

Consider below records example:

| Name | Age | DOB |
|------|-----|------------|
| abc | 32 | 29/08/1989 |
| efg | 20 | 09/12/2002 |
| abc | 32 | xyz |
| abc | 32 | 23/08/1989 |


In the 3rd record, date of birth (dob) is in the wrong data type, so the system rejects that record and fetches the other subsequent records.

Binary Parser

Last updated on February 5, 2024

Binary parser is designed to read and interpret binary data. Binary data is represented using sequences of bits. It extract meaningful information from binary files by interpreting binary data according to predefined structure or format. The process of binary parsing involves analyzing the

structure of the binary data and extracting relevant information based on that structure. In **HyperSense**, **Binary Parser** parse records from binary file using decoder XML file.

 This page provides step-wise configuration about the Transformation Operator. After you configure the Transformation Operator, go back to the *Pipeline Configuration Page* to know how to add and configure **Data Sink** to your pipeline.

To configure the Binary Data File Parser:


1. In Canvas, expand **Transformation Operator**. Drag and drop **Parser** in canvas.
2. Provide the **Data Source** input to the **Parser**.
3. Click the vertical ellipses and select **Edit**.
4. A **Select Parser File Type** window is displayed. Select the file type - **Data File Parser (Binary)**.
5. Select **Decoder XML** file to upload. Click **Define Parser**.
6. The **Configure: Data File parser (Binary)** is displayed. You can select the required Decoder XML file and click upload to upload the file.
7. Browse and select the **Decoder XML** file for upload.
8. The data inside the decoder XML file is displayed.
9. For successful upload:
 - a. No error found message is displayed.
 - b. Click **Save** to save parser configuration.
10. For unsuccessful upload:
 - a. Error message is displayed.
 - b. **Save** Icon is disabled.
 - c. Select correct decoder xml file and upload.
11. If you try to upload file other than decoder xml file, upload is unsuccessful.
12. Go back to the *Pipeline Configuration Page* to know how to add and configure **Data Sink** to the pipeline.

CSV Parser

Last updated on May 19, 2025

CSV Parser, also known as Comma-Separated Values Parser, is used to read and process CSV files. It allows to read the data of CSV file and convert it into structured format that can be analyzed and processed further. It is used to parse the CSV files having any delimiter characters such as comma,

semicolon, tabs, spaces etc. By using CSV parser, you can automate process of reading and extracting data from CSV files.

 This page provides step-wise configuration about the Transformation Operator. After you configure the Transformation Operator, go back to the *Pipeline Configuration Page* to know how to add and configure **Data Sink** to your pipeline.

To configure the CSV Data File Parser:

1. In Canvas, expand **Transformation Operator**. Drag and drop **Parser** in canvas.
2. Provide the **Data Source** input to the **Parser**.
3. Click the vertical ellipsis and select **Edit**.
4. A **Select Parser File Type** window is displayed. Select the file type as **Data File Parser (CSV)**.
5. A configuration window is displayed.
6. Refer [File Structure Definition](#) for configuring File structure, Refer [Record Structure Definition](#) for configuring Record Definition, and Refer [Record Integrity](#) to configure the missing record sequence and duplicate record.
7. After Parser configuration, connect with other transformation operator or data sink.

Properties and Actions in Window

Configure Data File Parser (CSV) window has the following properties and tabs:

1. **File Structure Definition:** This tab allows you to define the file structure definition. All the configurations in this window defines the structure of the file to be parsed. For more information on the properties, actions and attributes, see [File Structure Definition](#).
2. **Record Structure Definition:** This tab allows you to define the record structure definition. This tab allows you to define the data field in the records for the file loaded. Example: If we have any records with the following fields: ID, name, location, contact number. We have to define the **fields in this window to parse the data in the specified format**. For more information on the properties, actions and attributes, see [Record Structure Definition](#).
3. **Record Integrity:** This tab allows you to perform the missing record sequence check and duplicate record from the file. For more information, see [Record Integrity](#).

Common Actions

Common actions in **File Structure Definition** and **Record Structure Definition** window:

1. **Delete:** Allows you to delete the node.

2. **Copy:** Allows you to copy the node.
3. **Paste:** Allows you to paste the copied entry.
4. **Cut:** Allows you to cut the selected entry.
5. **Upload:** Allows you to [upload](#) a CSV file.
6. **Reset:** Allows you to revert back and reset the settings.
7. **Move Down:** Allows you to move the field down in the structure.
8. **Move Up:** Allows you to move the field up in the structure.

File Structure Definition

This tab allows you to define the file structure definition.

Configure the following details in **File Structure Definition** tab:

1. **Field Delimiter:** Enter the delimiter to separate the field values.
Example: If the delimiter is ",", then the field values of imported csv file are separated with comma.
 - **Multi Character Delimiter:** This feature allows you to use multiple delimiter values while configuring **File Structure Definition**. In the **Field Delimiter** text box, you can enter multiple field delimiters to separate the field values.
Example: If the delimiter is "@#!", then the field values of imported csv file are separated with delimiter '@#!'
2. **Quote Character:** This ignores the delimiter if the delimiter is entered in selected quotes (Single Quote/Double Quote).
3. **Skip Header:** Select the **Skip Header** checkbox and enter the **Count of Header Rows to Skip** from your input file. This allows you to skip the number of rows from Header. In the above example, the first 5 rows of available records are skipped from header.
4. **Skip Footer:** Select the **Skip Footer** checkbox and enter the **Count of Footer Rows to Skip** from your input file. This allows you to skip the number of rows from Footer. In the above example, the last 10 rows of available records are skipped from footer.

Record Structure Definition

This tab allows you to define the record structure definition for records in the file loaded. Example: If we have any records with the following fields: ID, name, location, contact number, and so on; To parse the data in the specified format, we have to define the fields in this tab in the specific structure.

To configure Record Structure Definition:

1. In **Record Structure Definition** tab, click **Add Field**.

2. **Name:** Enter the name for the column.
3. **Datatype:** Select the datatype from the drop-down. Available options are: String, Date Time, Decimal, Boolean, Integer, Long.
4. **Description:** Enter any info related to the column.
5. **Mandatory:** If selected, the column must contain a value.
6. **Enable Validation:** It allows you to validate the input file with the parser schema defined. It helps to avoid situations such as data contamination with incorrect records or parser failure. Select the checkbox to perform the validation. When checkbox is selected, below options are displayed:
 - a. **Skip Record:** It allows you to skip the records those are not matching the Record Structure definition.
 - b. **Skip File:** It allows you to skip the complete input file if it is not matching with Record Structure definition.
7. To add more field, follow the same above steps.
8. You can use the common actions available at the top left side of the window. For more information go to [common actions](#).
9. Click **Save Parser**.
10. If you wish to configure for missing file sequence and duplicate record check, Click [Record Integrity](#) tab.

Note:

Skip Error: This feature will reject the wrong(data type mismatch) or corrupted records in parser and process correct files. If one of the data record contains an error value, then the system skips that error and continues to fetch the subsequent records without interrupting the current record fetching process.

Consider below records example:

| Name | Age | DOB |
|------|-----|------------|
| abc | 32 | 29/08/1989 |
| efg | 20 | 09/12/2002 |
| abc | 32 | xyz |
| abc | 32 | 23/08/1989 |

In the 3rd record, date of birth (dob) is in the wrong data type, so the system rejects that record and fetches the other subsequent records.

Here is a configured example:

Record Integrity

This tab allows you to perform the missing record sequence check and duplicate record from the file. Complete the configuration:

1. **Record Sequence Check:** This check allows you to find the missing record sequence from the file. Select the **Enable** radio button to configure missing record sequence check. For more information, see [Missing Record Sequence Check](#).
2. **Duplicate Record Check:** This check allows you to find the duplicate records from the file. For more information, see [Duplicate Record Check](#).

Upload a sample CSV File

You can upload a CSV or .txt file in the Configure: Data File Parser (CSV) Window. To upload a file:

1. Go to configuration page of Data File Parser and click Upload.
2. Configure the following:
 - a. ***Upload file:** Allows you to browse and import CSV or .txt file.
 - b. **Header:** Select the checkbox to consider the first row as the header for the data. It will include header in the file that contains the column heading and subsequent rows contain data records.
 - c. ***Field Delimiter:** Enter the delimiter to separate the field values.
Example: If the delimiter is ",", then the field values of imported CSV file are separated with comma.
 - d. ***Record Delimiter:** Default value, cannot be edited. It indicates an end of line between the records.
4. Selected CSV file is imported and the data preview is displayed.
5. You can also add **Description** and make the fields as **Mandatory** to fetch only the fields which contains value.
6. To add more Fields, click **Add Field**.
7. After adding fields, click [Record Integrity](#) tab.

Missing Record Sequence Check

In Data Management pipeline, after loading the data, if any records are missed in between file processing, record sequence check feature identifies the missing records from the input files and generates an alert notification for missing record sequence. You can view the triggered alert in Triggered alerts screen.

To configure Record Sequence:

1. From the configure: **Data File Parser** Screen, Click the **Record Integrity** tab. By default, record sequence check option is disabled.

2. Click **Enable**. The Record Sequence window will be expanded automatically.
3. Configure the following details and click **Save**.
 - a. **General Details**
 - i. **Integrity Name:** This is a general name given to integrity file. Enter a name for the Integrity check.
 - b. **Record Sequence Check Configuration**
 - i. **Select Column:** This provides the list of columns for selection. Click the **Select Columns** dropdown and select the required column. **Select Column** dropdown displays only integer and long data type columns. After selecting the **Select Column**, you can not edit/modify the selected column in record structure definition.

Note:

 - In multiple record reference scenario, the dropdown for the record sequence identifier column will display only the columns that are present in all schema.
 - ii. **Record Start Number:** Record start number is the sequence number from where the record sequence need to start.
 - iii. **Record End Number:** Record end number is the sequence number where the record sequence ends.
 - iv. **Wait Time:** Wait Time is the time duration after loading a file that a system holds before raising an alert. Enter the required wait time.
 - v. **Time Frame:** Time Frame quantifies the units of wait time. Click the time frame dropdown and select the required time frame(**seconds/minutes/hours**).
 - c. **Configured Example:**
 - i. In the above configured example, if you configure Record Start number as 10 and Record End number as 20 and your input file has the records only from 10 to 15. In this scenario, as per record sequence configuration, the records must exist from 10 to 20 but they exist only from 10 to 15 and the remaining records from 16 to 20 are missing. Since the wait time for the record sequence is configured as one hour, after loading the file, system waits for one hour and generates an alert for the missing record sequence.
 - ii. If any missing record sequence is detected, an alert is generated and you can view the alert in **Triggered Alerts** screen as shown below.

Duplicate Record Check

In Data Management Pipeline, while loading the data, if any duplicate record is found while processing the file, duplicate record check feature identifies the duplicate records from the input files and generates an alert notification.

To view the triggered alert either Click Bell icon in the Pipeline Repository or visit alert screen. For more information on Triggered Alerts, see [Triggered Alerts](#).

To configure, click [Click Here](#). New Row is added.

Note:

- Incoming columns should have Timestamp datatype to check and generate alert in case of duplicate records are identified. For more information on how to create alert, see [Create Alert](#).
- It will process -1 day records. **For example:** Suppose System Date is 30th September'24 15:30:00 and you perform duplicate record check, records are checked from 29th September '24 00:00:00 till 30th September'24 15:30:00.

Configure below details and Click **Save**.

1. **Record Type:** This field displays the name of incoming file. In case incoming file has multiple worksheets, worksheet names are displayed in the dropdown list. Select the worksheet in which duplicate record should be performed.
2. **Timestamp Columns:** This field displays the date columns for checking the duplicate record. Select the date column from the dropdown list.
3. **Columns:** This field displays the list of columns on which duplicate check to be performed. Select the columns from the dropdown list. You can select multiple columns.
4. Click **Add Row** when you want to perform duplicate record check on multiple worksheet.
5. Click **delete** icon to remove the rows.
6. **Configured example:**
7. If any duplicate record is detected, an alert is generated and you can view the alert in **Triggered Alerts** screen.

Note:


[Click on URL link displayed in alert message to view the duplicate record details.](#)

Go back to the [Pipeline Configuration Page](#) to know how to add and configure Data Sink to the pipeline.

XLSX/XLS/XLSB Parser

[Last updated on May 19, 2025](#)

This Parser is used to parse the `xlsx/XLS/XLSB` files, it will create structure using sample `XLSX/XLS/XLSB` file and we can create structure manually as well to handle `XLSX/XLS/XLSB` files. Schema structure can be created based on with and without header according to the requirement. We can merge multiple files if schema structure is same and can load to single table.

 This page provides step-wise configuration about the Transformation Operator. After you configure the Transformation Operator, go back to the *Pipeline Configuration Page* to know how to add and configure **Data Sink** to your pipeline.

To configure the XLSX/XLS/XLSB Data File Parser:

1. In Canvas, expand **Transformation Operator**. Drag and drop **Parser** in canvas.
2. Provide the **Data Source** input to the **Parser**.
3. Click the vertical ellipsis and select **Edit**.
4. A **Select Parser File Type** window is displayed. Select the file type as **Data File Parser (XLSX/XLS/XLSB)**.
5. A configuration window is displayed.
6. Refer [File Structure Definition](#) for configuring File structure, Refer [Record Structure Definition](#) for configuring Record Definition, and Refer [Record Integrity](#) to configure the missing record sequence and duplicate record.
7. After Parser configuration, connect with other transformation operator or data sink.

Note:

- If you have configured a xls/xlsx/xlsb parser with multiple sheet(schema) in it, and you try to connect it with another component, then it will prompt to select the sheet(schema).
- The selected sheet(schema) will be displayed on the connection node.

Properties and Actions in Window

Configure Data File Parser (XLSX/XLS/XLSB) window has the following properties and tabs:

1. **File Structure Definition:** This tab allows you to define the file structure definition. All the configurations in this window defines the structure of the file to be parsed. For more information on the properties, actions and attributes, see [File Structure Definition](#).
2. **Record Structure Definition:** This tab allows you to define the record structure definition. This tab allows you to define the data field in the records for the file loaded. Example: If we have any records with the following fields: ID, name, location, contact number. We have to define the fields in this window to parse the data in the specified format. For more information on the properties, actions and attributes, see [Record Structure Definition](#).
3. **Record Integrity:** This tab allows you to perform the missing record sequence check and duplicate record from the file. For more information, see [Record Integrity](#).

Common Actions

Common actions in **File Structure Definition** and **Record Structure Definition** window:

1. **Delete:** Allows you to delete the node.
2. **Copy:** Allows you to copy the node.
3. **Paste:** Allows you to paste the copied entry.
4. **Cut:** Allows you to cut the selected entry.
5. **Upload:** Allows you to upload a XLSX or XLS file. Go to [Upload a Sample file](#) for more information.
6. **Revert:** Allows you to revert back and reset the settings.
7. **Move Down:** Allows you to move the field down in the structure.
8. **Move Up:** Allows you to move the field up in the structure.

File Structure Definition

This tab allows you to define the file structure definition.

Configure the following details in **File Structure Definition** tab:

1. **Skip Header:** Select the Skip Header checkbox and enter the count of Header Rows to Skip from your input file. This allows you to skip the number of rows from Header. In the above example, the first 5 rows of available records are skipped from header.
2. **Skip Footer:** Select the Skip Footer checkbox and enter the Count of Footer Rows to Skip from your input file. This allows you to skip the number of rows from Footer. In the above example, the last 5 rows of available records are skipped from footer.
3. **Merge:** It is enabled when it detects a xlsx or xls file with multiple identical sheets. It allows you to merge and combine two or more different sheets with same data in the xlsx or xls file. If the file has multiple sheets, and the sheets data has identical column and data type, then it will merge them together to generate a single output schema.
4. **Output Scheme Name:** Enter the desired output schema name.
5. Click [Record Structure Definition](#) tab.

Record Structure Definition

This tab allows you to define the record structure definition for records in the file loaded. **Example:** If we have any records with the following fields: ID, name, location, contact number, and so on; To **parse the data in the specified format**, we have to define the fields in this tab in the specific structure.

To configure Record Structure Definition:

1. In **Record Structure Definition** tab, click Add Field.

2. **Name:** Enter the name for the column.
3. **Datatype:** Select the datatype from the drop-down. Available options are:
 - a. String
 - b. Date time
 - c. Decimal
 - d. Boolean
 - e. Integer
 - f. Long
4. **Description:** Enter any info related to the column.
5. **Mandatory:** If selected, the column must contain a value.
6. **Enable Validation:** It allows you to validate the input file with the parser schema defined. It helps to avoid situations such as data contamination with incorrect records or parser failure. Select the checkbox to perform the validation. When checkbox is selected, below options are displayed:
 - a. **Skip Record:** It allows you to skip the records those are not matching the Record Structure definition.
 - b. **Skip File:** It allows you to skip the complete input file if it is not matching with Record Structure definition.
7. To add more fields, follow the same above steps.
8. You can also rename the sheet name as per incoming excel file.
9. You can use the common actions available at the top left side of the window. For more information go to [common actions](#).
10. Click **Save Parser**.
11. If you wish to configure for missing file sequence and duplicate record check, Click [Record Integrity](#) tab.

Note:

Skip Error: This feature will reject the wrong(data type mismatch) or corrupted records in parser and process correct files. If one of the data record contains an error value, then the system skips that error and continues to fetch the subsequent records without interrupting the current record fetching process.

Consider below records example:

| Name | Age | DOB |
|------|-----|------------|
| abc | 32 | 29/08/1989 |
| efg | 20 | 09/12/2002 |
| abc | 32 | xyz |

abc 32 23/08/1989

In the 3rd record, date of birth (dob) is in the wrong data type, so the system rejects that record and fetches the other subsequent records.

Here is a configured example:

Record Integrity

This tab allows you to perform the missing record sequence check and duplicate record from the file.

Complete the configuration:

1. **Record Sequence Check:** This check allows you to find the missing record sequence from the file. Select the **Enable** radio button to configure missing record sequence check. For more information, see [Missing Record Sequence Check](#).
2. **Duplicate Record Check:** This check allows you to find the duplicate records from the file. For more information, see [Duplicate Record Check](#).

Upload a Sample XLSX/XLS/XLSB file

This feature allows to create record structure based on the uploaded sample file.

1. Click **Upload icon**.
2. **Upload Excel file dialog box is displayed. Click *Upload File .**
3. Browse and select sample xlsx or xls or xlsb file.
4. **Uploaded file is selected. If the required xls or xlsx or xlsb file contain a header row and you want to include the header, then select the header checkbox. Click **Import** to proceed.**
5. Record Structure Definition tab is displayed.
6. If the uploaded file contains multiple sheets, then the sheets are listed as tabs along the bottom in the above shown screen. One sheet represent one schema.
7. **Click on the sheet name to switch to different sheet. A sheet contain its own collection of column to help you create your schema. While configuring the parser with another component, you can select the required sheet(schema).**
 - a. To create a new sheet, click the plus icon.
 - b. To delete a sheet, click cross icon.
 - c. To rename the existing sheet, click on the sheet name.
8. **In the below example, we have two Sheet 1 & 2. Sheet 1 represents one schema with different column. Whereas, Sheet 2 represents another schema with different column.**

9. You can add or modify fields in **Record Structure Definition** tab. To add fields, click **Add Field** and configure the following details:
 - a. **Name:** Enter the name for the column.
 - b. **Datatype:** Select the datatype from the drop-down. Available options are:
 - i. String
 - ii. Date time
 - iii. Decimal
 - iv. Boolean
 - v. Integer
 - vi. Long
 - c. **Description:** Enter any info related to the column.
 - d. **Mandatory:** If selected, the column must contain a value.
10. To add more fields, follow the same above steps.
11. You can use the common actions available at the top left side of the window. For more information go to [common actions](#).
12. Click **Save Parser**.
13. If you wish to configure for missing file sequence and duplicate record check, Click [Record Integrity](#) tab.

Missing Record Sequence Check

In Data Management pipeline, after loading the data, if any records are missed in between file processing, record sequence check feature identifies the missing records from the input files and generates an alert notification for missing record sequence. You can view the triggered alert in Triggered alerts screen.

To configure Record Sequence:

1. From the configure: **Data File Parser** Screen, Click the **Record Integrity** tab. By default, record sequence check option is disabled.
2. Click **Enable**. The Record Sequence window will be expanded automatically.

Configure the following details and click **Save**.

- a. **General Details**
 - i. **Integrity Name:** This is a general name given to integrity file. Enter a name for the Integrity check.

b. Record Sequence Check Configuration

- i. **Select Column:** This provides the list of columns for selection. Click the **Select Columns** dropdown and select the required column. **Select Column** dropdown displays only integer and long data type columns. After selection, you can not edit/modify the selected column in record structure definition.

Note:

In multiple record reference scenario, the dropdown for the record sequence identifier column will display only the columns that are present in all schema.

- ii. **Record Start Number:** Record start number is the sequence number from where the record sequence need to start.
- iii. **Record End Number:** Record end number is the sequence number where the record sequence ends.
- iv. **Wait Time:** Wait Time is the time duration after loading a file that a system holds before raising an alert. Enter the required wait time.
- v. **Time Frame:** Time Frame quantifies the units of wait time. Click the time frame dropdown and select the required time frame(**seconds/minutes/hours**).

c. Configured Example:

- i. In the above configured example, if you configure **Record Start** number as 10 and **Record End** number as 20 and your input file has the records only from 10 to 15. In this scenario, as per record sequence configuration, the records must exist from 10 to 20 but they exist only from 10 to 15 and the remaining records from 16 to 20 are missing. Since the wait time for the record sequence is configured as one hour, after loading the file, system waits for one hour and generates an alert for the missing record sequence.
- ii. If any missing record sequence is detected, an alert is generated and you can view the alert in **Triggered Alerts** screen as shown below. For more information on Triggered Alerts, see [Triggered Alerts](#).

Duplicate Record Check

In Data Management Pipeline, while loading the data, if any duplicate record is found while processing the file, duplicate record check feature identifies the duplicate records from the input files and generates an alert notification.

To view the triggered alert either Click Bell icon in the Pipeline Repository or visit alert screen. For more information on Triggered Alerts, see [Triggered Alerts](#).

To configure, click Click Here. New Row is added.

Note:

- Incoming columns should have Timestamp datatype to check and generate alert in case of duplicate records are identified. For more information on how to create alert, see [Create Alert](#).

- It will process -1 day records. **For example:** Suppose System Date is 30th September'24 15:30:00 and you perform duplicate record check, records are checked from 29th September '24 00:00:00 till 30th September'24 15:30:00.

Configure below details and Click **Save**.

1. **Record Type:** This field displays the name of incoming file. In case incoming file has multiple worksheets, worksheet names are displayed in the dropdown list. Select the worksheet in which duplicate record should be performed.
2. **Timestamp Columns:** This field displays the date columns for checking the duplicate record. Select the date column from the dropdown list.
3. **Columns:** This field displays the list of columns on which duplicate check to be performed. Select the columns from the dropdown list. You can select multiple columns.
4. Click **Add Row** when you want to perform duplicate record check on multiple worksheet.
5. Click **delete** icon to remove the rows.
6. **Configured example:**
7. If any duplicate record is detected, an alert is generated and you can view the alert in **Triggered Alerts** screen.

Note:

Click on URL link displayed in alert message to view the duplicate record details.

Go back to the [Pipeline Configuration Page](#) to know how to add and configure Data Sink to the pipeline.

XML Parser

Last updated on December 7, 2023

XML Parser allows you to parse the XML file. It allows you to parse the multilevel xml file with multiple schema.



This page provides step-wise configuration about the Transformation Operator. After you configure the Transformation Operator, go back to the [Pipeline Configuration Page](#) to know how to add and configure **Data Sink** to your pipeline.

To configure the XML Parser:

1. Create a new pipeline (or) edit the existing pipeline.
2. Make sure Data Source is present and configured.

3. In Canvas, expand **Transformation Operator**. Drag and Drop **Parser**.
4. Provide the input connection to parser from data source.
5. Click on Vertical Ellipses > **Edit**.
6. Select **Parser File Type** window is displayed.

Complete the following:

- a. **File Type:** Select Data File Parser(XML) from the dropdown list.
 - b. **Upload XML Decoder File:** Select the XML decoder file from the system and upload.
 - c. Click **Define Parser**.
7. **Configure: Data File Parser(XML)** window is displayed.

Window has following properties:

- a. **Upload:** It allows you to upload Decoder XML file. In case you have refresh the window, the existing Decoder XML file is removed. You can click upload to upload the Decoder XML file.
- b. **Download:** It allows you to download the grammar of Decoder XML file. Downloaded file format is ".txt".
- c. **Refresh:** It allows you to refresh the uploaded Decoder XML file while defining parser. You can upload new Decoder XML file, Click on Upload.
- d. **Generate:** It allows you to generate the structure of Decoder XML file. Make sure you upload Decoder XML file with proper grammar and Click **Generate**.
- e. **Save:** It allows you save the parser configuration. Complete the Structure Definition and Click **Save**.
- f. **Tabs:** There are two tabs available in the configuration window, which are listed as below:
 - i. **Grammar:** This tab contains the grammar and error details of the uploaded Decoder XML file. It has below fields:
 - **Decoder XML:** It is a radio button to indicate uploaded file is Decoder XML file. When you upload Decoder XML file radio button is pre-selected, else it will be clear.
 - **Grammar:** It contains the grammar details of uploaded Decoder XML file. When you upload Decoder XML file, it will check for any exception or error in the file and displays in error log panel. You can expand and check the grammar details.
 - **Error Log Panel:** It displays the details regarding the error found in the Decoder XML file.
 - If file has no error, message displayed is "No error found".
 - If file has error, message displayed is "Error in File". You can expand the details to find out the error in file.

- ii. **Structure Definition:** This tab is enabled once you upload the Decoder XML file and Click **Generate**. This tab contains the grammar structure of the uploaded Decoder XML file and allows you to check the grammar format. To check the grammar format, upload the sample file and parse the data.

It has below properties:

- **Grammar:** It displays the grammar details for the uploaded Decoder XML file.
- **Upload Sample File:** To verify if the parsing is done in proper grammar format or not, you have to upload sample file with specific structure. Select the sample xml file from the system and upload. Once you upload the sample file, Raw data is displayed in the form of hexadecimal. Next step is to **Parse Data**.
- **Parse Data:** Once you upload the sample file, Click **Parse Data**. In the **Decoded** panel, the parsed data is on the left side and the raw data in on the right side.

8. Click **Save** to keep the configuration.
9. Output from the XML Parser is Json. Connect the Parser with Flattener. To know more about **Flattener Configuration**, refer to [Flattener](#).

Pre Parser

Last updated on February 5, 2024

This is a Transformation Operator component where we select Specific Jar Configured in Custom JAR Registration to operate on the incoming data along with properties file. The output is validated with the Output Schema file which is uploaded in same Jar Registration and gives output only when schema is matched.

To configure Pre Parser:

1. In Canvas, expand **Transformation Operator**. Drag and drop **Pre Parser** in canvas.
2. Connect the **Data Source** to **Pre Parser** and click ellipsis and click **Edit**.
3. The **Configure Pre Parser** window is displayed.
4. Select the required JAR and click **Save**. If you select the JAR file with schema, then you can connect the Preparser directly to other transformation operators or to the **Data Sink**. If you select the JAR file without schema, then you need to connect the **Pre Parser** to the **CSV Parser** as the output of pre parser is comma separated.
5. Connect the Pre Parser to Data Sink.
6. Upload appropriate input file, and then deploy and run the pipeline to see output in Data Sink.

Note:

Preparser can also support files which have key and values enclosed with flower braces. To configure this, you must register a jar in custom jar registration pane. For more information on how to register a jar, see [Custom Jar Registration](#).

Sample File:

```
{
1=91818191
2=1819
3=191
}
{
1=373782
2=929
3=0
}
```

Record Processor

Last updated on February 5, 2024

Record Processor will process the incoming records from **Subscriber Mapper**, and execute the actions as per the defined rule. Actions are executed according to defined rules and result is updated in the subscriber and account table. Input to Record Processor is from Subscriber Mapper. While connecting from Subscriber Mapper, you should select one of the schema. Output from **Record Processor is connected to RDBMS Sink**. In order to handle Record Processing a postgres table is maintained which holds meta data of incoming records from a kafka topic. After 10 days the processed records will be cleaned up from this table.



This page provides step-wise configuration about the **Transformation Operator**. After you configure the **Transformation Operator**, go back to the *Pipeline Configuration Page* to know how to add and configure **Data Sink** to your pipeline.

Below are few example:

1. If **Account** is identified as **Blacklist**, then account status is updated as "Blacklist" in account table.
2. If **Subscriber** is identified as **Blacklist**, then subscriber status is updated as "Blacklist" in subscriber table.

Configuration

Steps to configure Record Processor as follows:

1. In Canvas, expand **Transformation Operator**. Drag and drop **Record Processor** in canvas.
2. Connect **Subscriber Mapper** to the **Record Processor**. Select the schema for configuration and Save.
3. Click on **Vertical Ellipsis > Edit**.
4. **Configure: Record Processor** window is displayed. You will find below details in window:
 - a. **Search Bar**: Allows you to search specific rule from the list.

- b. **Add Rule:** Click to add new rule. You can add multiple rules as per requirement.
 - c. **No Condition Set:** This displays the rule which are defined. Initially, no rule(condition) is defined. You should expand and complete the details to set the rule.
 - d. **Done:** Click to save the record processor configuration.
5. Click on expand icon to define Rule.
6. Rule Configuration pane is displayed in expanded view.

Complete below details to set rule:

- a. **Rule Name:** Rule name is unique name which helps to identify the rule that you have defined. Enter the unique Rule Name. This is mandatory field.
- b. **Expression Box:** This box allows you to enter the expression and define rule to process incoming record.
- c. **Validate:** Click to validate the expression. This is mandatory action to save rule.
 - i. If expression is correct, Valid expression message is displayed.
 - ii. If expression is incorrect, Invalid expression message is displayed.
- d. **Format:** Click to auto adjust the syntax of the expression.
- e. **Clear:** Click to clear the written expression. Clear Button is enabled once the expression is written.
- f. **Action to be taken:** Select the actions from the dropdown to perform actions on the basis of defined expression. Available options are:
 - i. **Blacklist Subscriber:** When expression matches with the incoming record, then status column in subscriber table is updated as "Blacklist". In the absence of such record, it gets inserted into subscriber table. For an account which has single subscriber and subscriber status is updated as blacklist, then account is updated as blacklist.
 - ii. **Blacklist Account:** When expression matches with the incoming record, then **Accountstatus** column in account table is updated as "Blacklist". In the absence of such record, it gets inserted into account table. When account is blacklisted, all associated subscribers are blacklisted.
 - iii. **Close Case as Fraud:** When expression matches, all the cases associated with subscriber or account gets closed as fraud.
 - iv. **Close Case as Non-Fraud:** When expression matches, all the cases associated with subscriber or account gets closed as non-fraud.

Note:

When you select either "Close Case as Fraud" or "Close Case as Non-Fraud" Action, in **Process Automation Studio** all the cases associated with the entity value are closed as Fraud or Non-Fraud depending upon the action taken. An entity may be a direct field or an alias field.

- v. **Reject Record:** When expression matches, the record gets rejected. Record is not updated into subscriber or account table.
 - vi. Below actions can be used in combination while setting rules:
 - Blacklist Subscriber & Close Case As Fraud
 - Blacklist Subscriber & Close Case As Non Fraud
 - Blacklist Account & Close Case As Fraud
 - Blacklist Account & Close Case As Non Fraud
 - Blacklist Subscriber & Blacklist Account
 - g. **Cancel:** Click to discard the rule setting.
 - h. **Apply:** Click to keep the rule setting. Apply Button will be enabled once the expression is successfully validated and action to be taken is selected.
7. Click **Apply**, Rule is shown in the list.
 8. To delete rule, Click on delete icon. Delete icon is displayed when there are 2 or more rules in the list.
 9. Output from the Record Processor should be connected to [RDBMS Sink](#).

Use Case Scenario

Let us consider the below dataset that has 3 records and 5 columns as shown below:

| subscriber_id | Subscriber_number | account_name | account_id | imsi_incoming |
|---------------|-------------------|--------------|------------|---------------|
| 1 | 9939483746 | account2 | 1 | 9876543211123 |
| 2 | 8893847453 | account1 | 2 | 9876543212345 |
| 3 | 7738495473 | account3 | 3 | 9876543211234 |

Case 1: When Action to be taken is "Close Case as Fraud" & "Blacklist Subscriber"

In this case, SubscriberID = 1 is identified as fraud and should be blacklisted. Let us consider, you have new case in Process Automation in which Grouped Entity used is IMSI that has value 9876543211123. The case generated in the process automation will be closed as fraud and subscriber will be marked blacklisted by defining the rule in the record processor. Follow below steps to perform the action:

1. In Process Automation, you have new case in which Group Entity has value **IMSI: 9876543211123**.
2. In Record Processor, create the rule to identify subscriberid = 1, and select the actions as **Blacklist Subscriber** and **Close Case as Fraud**.

3. In Process Automation > Grouped Cases, Closure Category for grouped entity IM-SI: 9876543211123 is marked as **Fraud**. Case Status is changed to Closed.
4. In Subscriber Table, status of subscriber with subscriberid=1 is marked as **Blacklisted**.

Case 2: When Action to be taken is "Close Case as Non-Fraud"

In this case, AccountID = 2 is identified as non-fraud. Let us consider, you have new case in Process Automation in which Grouped Entity used is Accountid that has value 02. The case generated in the process automation will be closed as non-fraud by defining the rule in the record processor. Follow below steps to perform the action:

1. In Process Automation, you have the new case in which Group Entity has value Accountid: 02.
2. In Record Processor, create the rule to identify account_id = 2, and select the actions as **Close Case as Non-Fraud**.
3. In Process Automation > Grouped Cases, Closure Category for grouped entity Accountid:02 is marked as **Non Fraud**, Case Status is changed to Closed.

Case 3: When Action to be taken is "Reject Record"

In this case, record is rejected when rule is satisfied. Record is not processed further. Let us consider, you want to reject the record that has IMSI value "9876543211234" and remaining records should be updated in the database. Follow below steps to perform the action:

1. In Record Processor, we are defining rule for IMSI = 9876543211234.
2. When the above data set is parsed through record processor, record that has IMSI value "9876543211234" is rejected.
3. Database is updated with remaining 2 records.

Sampling


Last updated on [September 6, 2024](#)

Sampling is a Transformation Operator that is used to filter the data while processing the incoming CDR based on the sampling method and selected column. If the match is found as per the selected column, matched record is sent to next operator.

Note:

- At present, MSISDN based sampling is supported by Sampling Operator.
- Input to Sampling is either from Parser or Mapping.

- Sampling can be used in ARC pipeline.

 This page provides step-wise configuration about the Transformation Operator. After you configure the Transformation Operator, go back to the *Pipeline Configuration Page* to know how to add and configure **Data Sink** to your pipeline.

To configure Sampling Operator, you must configure the following:

1. In Canvas, expand **Transformation Operator**. Drag and drop **Sampling** in canvas.
2. Provide the input link to the **Sampling** Operator.
3. Click the vertical ellipsis, and select **Edit**.
4. A **Configure: Sampling** window is displayed.
5. Complete the below details:
 - a. **Choose Method**: It displays the sampling method to perform the sampling. By default, method is **MSISDN based sampling**.
 - b. **Column Selection**: Select the column to perform the sampling.
 - c. **MSISDN Fields Configuration**: There are three ways to perform the sampling using MSISDN fields. Atleast one of the three MSISDN fields should be configured. MSISDN configuration ways listed as below:
 - i. **Upload MSISDN**: Allows you to Upload only CSV file for sampling. For more information, refer to [Upload MSISDN](#).
 - ii. **Configure MSISDN**: Allows you to add MSISDN number for sampling. You can add multiple MSISDN numbers. For more information, refer to [Configure MSISDN](#).
 - iii. **Configure MSISDN Ranges**: Allows you to configure the MSISDN by defining range through expression. For more information, refer to [Configure MSISDN Ranges](#).
6. Click **Submit** button to save the configuration.

Upload MSISDN

For sampling you can upload the CSV file. Below are the steps to upload MSISDN:

1. Expand **Upload MSISDN** tab.
2. You can either drag and drop the CSV file or upload it from system. Click upload icon to select file from system.
3. Select the file from the location. Click **Open**.
4. Below details are displayed:
 - a. **File Name**: It displays the CSV filename that is uploaded.

- b. **Delete:** This icon allows you to delete the uploaded file. Click Delete icon to delete the uploaded file.
 - c. **View:** This icon allows you to view the csv file details. Click View icon to view the details.
5. Once file is uploaded, Submit button is enabled. Click **Submit** button to save the configuration.
 6. If you try to upload file other than CSV file, snackbar message is displayed - "**Please upload a .csv file**".

Configure MSISDN

For sampling you can configure MSISDN number, either single or multiple. Below are the steps to configure MSISDN:

1. Expand **Configure MSISDN** tab.
2. Click **Add New Row** to add the MSISDN number.
3. Enter the **MSISDN numbers**. **Submit** button is enabled.
4. Click **Submit** button to save the configuration.
5. If you want to remove any MSISDN number, Click delete icon.

Configure MSISDN Ranges

For sampling you can configure the MSISDN number for wide range. Below are the steps to configure MSISDN ranges:

1. Expand the **Configure MSISDN Ranges** tab.
2. Enter the regular expression.
3. Click **Validate** icon to validate the expression. If expression is valid, message is displayed - "**Valid Expression**". Else "**Invalid expression**" is displayed.
4. Once expression is validated, **Submit** button is enabled. Click **Submit button to save the configuration**.

Subscriber Mapper

Last updated on February 5, 2024


Subscriber Mapper is one of the transformation operators that segregates the incoming schema into **subscriber and account record references**. Input to the subscriber mapper is from any of the transformation operators. The output from Subscriber Mapper is connected to [Record Processor](#). While connecting Subscriber Mapper to Record Processor, you should select one of the schemas.

By default, there are two schema:

1. [Subscriber Record Reference](#)

2. [Account Record Reference](#)

To know more about how to configure Subscriber Mapper, refer to [Configuration](#).

 This page provides step-wise configuration about the Transformation Operator. After you configure the Transformation Operator, go back to the [Pipeline Configuration Page](#) to know how to add and configure [Data Sink](#) to your pipeline.

Subscriber Record Reference

This schema holds the information about the subscriber. Individual subscriber is linked to specific account. By default below 9 columns should be configured which cannot be deleted:

1. **Subscriberid:** This column returns the subscriber id.
2. **Subscribernumber:** This column returns the subscriber number.
3. **Producttype:** This column returns the product type.
4. **Subscriberactivationdate:** This column returns the activation date of subscriber.
5. **Status:** This column returns the status of the subscriber. It could be Active, Deactivated, Disconnected, Suspended etc.
6. **IMSI:** This column returns the International Mobile Subscriber Identity number. It is 15 digit unique number to identify individual users to cellular network.
7. **SubscriberAccountname:** This column returns the account name of the subscriber. This should match with account name from account record reference.
8. **SubscriberSystemFileName:** This column returns the file name of incoming file. It has default apply function as `$toString($$fileName)`.
9. **SubscriberLastUpdatedDateTime:** This column returns the last updated date time when incoming file was processed. It has default apply function as `$now()`.

Out of above 9 column, column names: `SubscriberSystemFileName` and **SubscriberLastUpdated-DateTime** are auto-populated; they should not be modified.

Account Record Reference

This schema holds the information about the account. By default below 6 columns should be configured which cannot be deleted:

1. **Accountname:** This column returns the name of account.
2. **Accountid:** This column returns the ID of account.
3. **Accountactivationdate:** This column returns the activation date of account.
4. **Accountstatus:** This column returns the status of the account. It could be Active, Deactivated, Disconnected, Suspended etc.

5. **AccountSystemFileName:** This column returns the file name of incoming file. It has default apply function as `$toString($$fileName)`.
6. **AccountLastUpdatedDateTime:** This column returns the last updated date time when incoming file was processed. It has default apply function as `$now()`.

Out of above 6 column, column names: **AccountSystemFileName** and **AccountLastUpdatedDateTime** are auto-populated; they should not be modified.

Configuration

Steps to configure subscriber mapper as follows:

1. In Canvas, expand **Transformation Operator**. Drag and drop **Subscriber Mapper** in canvas.
2. Provide the input link to the **Subscriber Mapper**.
3. Click on **Vertical Ellipsis > Edit**.
4. **Configure: Subscriber Mapper** window is displayed.
5. Click on the schema to configure.
6. To configure **Subscriber Record Reference** schema, steps as follows:
 - a. Click on expand icon to expand schema.
 - b. **Subscriber Record Reference** is displayed in expanded view.
 - c. Complete the [schema details](#).
 - d. Click **Apply** to keep schema configuration.
7. To configure **Account Record Reference** schema, steps as follows:
 - a. Click on expand icon to expand schema.
 - b. **Account Record Reference** is displayed in expanded view.
 - c. Complete the [schema details](#).
 - d. Click **Apply** to keep schema configuration.
8. Complete the below schema details for available schemas:
 - a. **Incoming Column Name:** Displays name of column to be mapped. Column name is auto populated as per previous component.
 - b. **Incoming Data Type:** Displays data type of incoming column. Data Type auto populate as per previous component.
 - c. **New Column Name:** Displays new name for the incoming column. It is mandatory field. By default, new column name is same as incoming column name. This field can be modified, if required.

- d. **New Column Data Type:** Displays data type of new column name. It is mandatory field. By default, new column data type is same as incoming column data type. This field can be modified, if required.
 - e. **Apply Function:** Displays the configured apply function. As per configured Apply Function, specific operation is performed and the value outcome is assigned to the **New Column Name**. To know more about different Apply Functions, refer [Apply Functions](#).
 - f. **Add Field:** Allows you to add new field i.e. new column. You can provide column name, select data type, set apply function. You can delete newly added fields, if not required. To delete, Click Delete Icon.
 - g. **Add All Columns:** Select all the Incoming Column at single click.
 - h. **Remove All Columns:** Remove all the Incoming Column at single click.
 - i. **Cancel:** Click **Cancel** to discard schema configuration.
 - j. **Apply:** Click **Apply** to keep schema configuration.
9. Click **Save** to keep the Subscriber Mapper Configuration.
10. Subscriber Mapper should be connected to [Record Processor](#).

Data Sink

Data Sink

Last updated on May 12, 2023

The third step involved in Data Modelling Studio is to have Data Sink. It helps you to load the data into the selected database. DMS supports following types of data sink:

- [Azure Blob Storage](#)
- [Azure Blob Storage Trino](#)
- [Cache](#)
- [Custom Sink](#)
- [GCS](#)
- [GCS Trino](#)
- [HDFS](#)
- [Hive](#)
- [Kafka](#)
- [Local File](#)
- [RDBMS](#)
- [S3](#)
- [S3Presto](#)

Azure Blob Storage Data Sink

Last updated on January 19, 2024

Azure Blob Storage Sink is used to load the structured data to Azure Blob Storage server. Azure Blob Storage Sink stores the records in file level.

 This page provides step-wise configuration about the Data Sink. After you configure the Data Sink, go back to the *Pipeline Configuration Page* to know how to **schedule your pipeline**.

Note:

- Before you configure, provide the connection link from any Transformation Operator to the Data Sink.

To configure Azure Blob Storage as Data Sink, you can [Create New Connection](#) or use [Existing Connections](#) from the list.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Sink**. Drag and drop **Azure Blob Storage** sink in canvas.
2. Provide an input link from the transformation operator.
3. Click vertical ellipsis, select **Edit** option.
4. **Configure Azure Blob Storage** window is displayed. Click on **Create New Connection**.
5. **Connection Details** tab is displayed. Configure the below detail:
 - a. **General Details:**
 - i. **Connection Name:** Provide the connection name for the **Azure Blob Storage** sink. Specify unique name for name field. If you specify the same name which is specified for another sink name, an error message is displayed.
 - b. **Azure Connection Details:**
 - i. **Account Name** : Enter the account name for azure blob storage.
 - ii. **Account Key:** Enter the account key from azure portal.
 - c. **Test Connection:** Click Test Connection option to test the connection status. The connection status is displayed if the connection is Successful or failed.
 - d. **Click Next.**
6. **Column Details** tab is displayed.

Complete the below details:

- a. **Container Name:** Enter the container name available in the azure portal.
 - b. **Prefix:** Enter the path where the output file will be stored. If not specified, then it is saved at root container folder.
 - c. **File Format:** Select the output file format from the drop-down. Available file formats are: **CSV, JSON, ORC, and Avro.**
 - d. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like **Reference Tag, usage Tag and system internal tag** which comes under default tags category and populated from backend.
 - i. **To add or Specify a tag, From Configure: RDBMS window,** select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
or
Place the cursor at Type to add new, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.
 - ii. **Usage and Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.
 - e. **Incoming Columns:** Displays the incoming columns from the Transformation Operator.
 - f. **Partition Columns:** Click **Add Field**, a row is added. All the **incoming columns** are available as drop down in the **Partition Name** field. Select the required column based on which the data must be partitioned while storing to Azure Blob Storage.
7. Click **Save**.

Configured Example:

Existing Connections Re-usability

Steps to use existing connection as follows:

1. To use the existing connection, select any of the them from **Recently Created or All Connections**.
2. If you want to edit the existing connections and use, then click edit icon and perform the modifications.

Note:

- If the selected connection is already used for any other sinks, then the modifications will impact those sinks connection as well.

3. Click **Next**.
4. **Configure Azure Blob Storage** Column Details tab is displayed.

Complete the below column details:

- a. **Container Name:** Enter the bucket name.
 - b. **Prefix:** Enter the path where the output file will be stored. If not specified, then it is saved at root container folder.
 - c. **File Format:** Select the file format from the drop-down. Available file formats are: CSV, JSON, ORC, and Avro.
 - d. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like Reference Tag, usage Tag and system internal tag which comes under default tags category and populated from backend.
 - i. **To add or Specify a tag, From Configure: RDBMS window,** select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
or
Place the cursor at Type to add new, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.
 - ii. **Usage and Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.
 - e. **Incoming Columns:** Displays the incoming columns from the Transformation Operator.
 - f. **Partition Columns:** Click **Add Field**, a row is added. All the **incoming columns** are available as drop down in the **Partition Name** field. Select the required column based on which the data must be partitioned while storing to Azure storage.
5. Click **Save**.
6. Go back to the [Pipeline Configuration Page](#) to know how to schedule the pipeline.

Note:

After saving the configuration, if you deploy and run the pipeline, the data is stored in the given container and a new dataset is created in Data Catalog.

Azure Blob Storage Trino Sink

Last updated on January 19, 2024

Azure Blob Storage Sink is used to load the structured data to Azure Blob Storage server. Azure Blob Storage Sink stores the records in table level.



This page provides step-wise configuration about the Data Sink. After you configure the Data Sink, go back to the [Pipeline Configuration Page](#) to know how to schedule your pipeline.

Note:

- Before you configure, provide the connection link from any Transformation Operator to the Data Sink.
- Below are the alter table actions that can be performed in Azure Blob Storage Trino Sink.
 - Adding a column
 - Deleting a column.Changing column name.

To configure Azure Blob Storage Trino as Data Sink, you can [Create New Connection](#) or use [Existing Connections](#) from the list.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Sink**. Drag and drop **Azure Blob Storage Trino** sink in canvas.
2. Provide an input link from the transformation operator.
3. Click vertical ellipsis, select **Edit** option.
4. **Configure Azure Blob Storage Trino** window is displayed. Click on **Create New Connection**.
5. **Connection Uniqueness Validation Feature for tabular Sinks**: In Azure Blob Storage Trino Data Sink, while creating new connection, if user gives already existing connection details, then it will shows connection uniqueness validation pop-up with existing connection name. It is implemented to avoid duplication of datasets created in data catalog. It is implemented for table based sinks such as RDBMS, S3Presto, GCSTrino, Azure Blob Storage Trino and Hive Sinks.
6. **Connection Details** tab is displayed.
Complete the below details:
 - a. **General Details**:
 - i. Provide the connection name for the **Azure Blob Storage** sink. Specify unique name for name field. If you specify the same name which is specified for another sink name, an error message is displayed.
 - b. **Azure Trino Connection Details**:
 - i. **Account Name** : Enter the account name for azure blob storage.
 - ii. **Account Key**: Enter the account key from azure portal.
 - c. **Trino Connection Details**:
 - i. **Hostname**: Enter the hostname for trino connection details.
 - ii. **Port**: Enter the port number for trino connection details.
 - iii. **Username**: Enter the user name for trino connection details.
 - iv. **Password**: Enter the password for given username.
 - d. **Hive Metastore Connection Details**: Enter the hive metastore connection URL.

- e. **Test Connection:** Click test connection option to test the connection status. The connection status is displayed if the connection is successful or failed.
 - f. **Click Next.**
7. **Table Details** tab is displayed.

Complete the below details:

- a. **Container Name:** Enter the container name available in the azure portal.
- b. **Schema Name:** Enter the schema name for table details.
- c. **Table Name:** Enter the table name.
- d. **Prefix:** Enter the path where the output file will be stored. If not specified, then it is saved at root container folder.
- e. **Warehouse Directory:** After entering containers name, the path for **Warehouse Directory** is auto generated.
- f. **File Format:** By default connection type is **Parquet**.
- g. **Merge File:** After selecting connection type, the **Merge File** drop down is displayed.
- h. **File Size (in MB):** After you enable the **Merge File** option, **File Size (in MB)** field box is displayed. Enter the size of the file to be merged.
- i. **File Merge Interval (in Seconds):** Enter the required duration in seconds. It will merge the file till it reaches to the provided size and then process. If the file size is not reached, it will wait until configured duration and will process. For example, if you provide file merge size as 10mb and file merge interval as 10 seconds. While merging incoming flow files if size reaches to 10mb it will start processing otherwise, it will wait until file merge interval provided 10 seconds and then it will start processing.
- j. **Primary Key:** Selected column act as primary key for the table. Select the column from the dropdown list.
- k. **Truncate and Load:** On selecting the **Truncate and Load** checkbox. This feature allows you to delete the existing records and load the new records with in the same table structure.
- l. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like **Reference Tag**, **usage Tag** and **system internal tag** which comes under default tags category and populated from backend.
 - i. **To add or Specify a tag, From Configure: RDBMS** window, select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
or
Place the cursor at Type to add new, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.
 - ii. **Usage and Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.
- m. **Incoming Columns:** Displays the incoming columns from the Transformation Operator.

- n. **Partition Columns:** Click **Add Field**, a row is added. All the **incoming columns** are available as drop down in the **Partition Name** field. Select the required column based on which the data must be partitioned while storing to Azure Blob Storage.

Note:

If partition column is configured, then truncate and load is disabled.

8. Click **Save**.

Here is a configured example:

Existing Connections Re-usability

Steps to use existing connection as follows:

1. To use the existing connection, select any of the them from **Recently Created** or **All Connections**.
2. If you want to edit the existing connections and use, then click edit icon and perform the modifications.

Note:

- If the selected connection is already used for any other sinks, then the modifications will impact those sinks connection as well.
- In Azure Blob Storage Trino Data Sink, while reusing existing connection, click the **Table Details** button to navigate to the next tab.

3. Click **Next**.
4. **Configure Azure Blob Storage Trino Table Details** tab is displayed.
Complete the below details:
 - a. **Container Name:** Enter the container name available in the azure portal.
 - b. **Schema Name:** Enter the schema name for table details.
 - c. **Table Name:** Enter the table name.
 - d. **Prefix:** Enter the path where the output file will be stored. If not specified, then it is saved at root container folder.
 - e. **Warehouse Directory:** After entering containers name, the path for Warehouse Directory is auto generated.
 - f. **File Format:** By default connection type is **Parquet**.
 - g. **Merge File:** After selecting connection type as Parquet, the **Merge File** drop down is displayed.

- h. **File Size (in MB):** After you enable the Merge File option, File Size (in MB) field box is displayed. Enter the size of the file to be merged.
- i. **File Merge Interval (in Seconds):** Enter the required duration in seconds. It will merge the file till it reaches to the provided size and then process. If the file size is not reached, it will wait until configured duration and will process. For example, if you provide file merge size as 10mb and file merge interval as 10 seconds. While merging incoming flow files if size reaches to 10mb it will start processing otherwise, it will wait until file merge interval provided 10 seconds and then it will start processing.
- j. **Primary Key:** Selected column act as primary key for the table. Select the column from the dropdown list.
- k. **Truncate and Load:** On selecting the **Truncate and Load** checkbox. This feature allows you to delete the existing records and load the new records with in the same table structure.
- l. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like Reference Tag, usage Tag and system internal tag which comes under default tags category and populated from backend.
 - i. **To add or Specify a tag, From Configure: RDBMS** window, select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
or
Place the cursor at Type to add new, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.
 - ii. **Usage and Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.
- m. **Incoming Columns:** Displays the incoming columns from the Transformation Operator.
- n. **Partition Columns:** Click **Add Field**, a row is added. All the **incoming columns** are available as drop down in the **Partition Name** field. Select the required column based on which the data must be partitioned while storing to Azure storage.

Note:

If partition column is configured, then truncate and load is disabled.

- 5. Click **Save**.
- 6. Go back to the [Pipeline Configuration Page](#) to know how to schedule the pipeline.

Note:

After saving the configuration, if you deploy and run the pipeline, the data is stored in the given container and a new dataset is created in Data Catalog.

Cache

Last updated on January 19, 2024

A cache is a high-speed data storage layer which stores a subset of data, typically transient in nature, so that future requests for that data are served up faster than is possible by accessing the data's primary storage location. It allows you to efficiently reuse previous retrieved or computed data. In a distributed cache, data is partitioned and distributed across multiple cache nodes or servers. Each node holds a portion of the cached data, and together, they form a distributed cache cluster. When an application needs to access a piece of data, it first checks if it exists in the local cache of the node it is running on. If the data is not present locally, the cache system retrieves it from a remote cache node within the cluster.



This page provides step-wise configuration about the Data Sink. After you configure the Data Sink, go back to the *Pipeline Configuration Page* to know how to schedule your pipeline.

Note:

Before you configure, provide the connection from any Transformation Operator to the Data Sink.

To configure Data Sink as Cache follow the steps below:

1. In Canvas, expand **Data Sink**. Drag and drop **Cache sink in canvas**.
2. Provide an input connection link.
3. Click vertical ellipsis, select **Edit** option.
4. A **Select Cache Type** window is displayed. Select the type of cache required and click **Next**.
5. Configure the following connection details for Distributed Cache type:
 - **Connection URL**: Enter the URL of the machine where the Ignite service is running.
 - **Cache Name**: Enter a name for this sink component.
 - **Expiry Duration (in seconds)**: Allows you to enter seconds after which the output files will be deleted.
 - **Cache Value Update Strategy**:
 - **Override**: If it finds a duplicate key, it will over write and keep only one value. Suppose, Subex column contains two records in country value, first is India and second is America. then system will keep the latest one that is America.
 - **Append**: If it finds a duplicate key, it will not over write, but keeps both the value. Suppose, Subex column contains two records in country value, first is India and second is America. then system will keep both India and America.

- **Truncate and Load:** It will help to Truncate the existing cache and load the required data. It helps to improve the sink performance.
 - **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like **Reference Tag**, **usage Tag** and **system internal tag** which comes under default tags category and populated from backend.
 - **To add or Specify a tag, From Configure: Distributed Cache window,** select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
 - or
 - Place the cursor at **Type to add new**, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.
 - **Usage** and **Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.
 - **Cache Key Value Configuration:** Allow you to configure the column and the cache value.
 - **Configuration Preview:** Displays the key and the values.
 - **Test Connection:** Click the **Test Connection** option to test the connection status. The connection status is displayed if the connection is Successful or Failed.
 - **Distributed Cache Configuration Example:**
 - **Local Cache Configuration Example:**
6. Click **Save**.
7. Go back to the [Pipeline Configuration Page](#) to know how to schedule the pipeline.

Custom Sink

Last updated on January 19, 2024

This is a **Data Sink** component where we select Specific Jar Configured in Custom JAR Registration to operate on the incoming data along with properties file. The output is validated with the Output Schema file which is uploaded in same Jar Registration and gives output only when schema is matched. In the output, the incoming data can be pushed into sink as per the configuration made in the registered Jar.

To configure Custom Sink:

1. In Canvas, expand **Data Sink**. Drag and drop **Custom Sink** in canvas.
2. Provide an input link from the transformation operator.
3. Click vertical ellipsis, select **Edit** option.

4. **Configure: Custom Sink** window is displayed. From **Select Jar** tab, select the required Jar from available list.
5. If it is required to make changes to existing configuration, expand the JAR and Click **Override**.
6. **Upload Properties File** and **Upload Output Schema File**. This change is local to the pipeline and will not update the JAR registered in Registration screen. Click **Save**.
7. Click **Next**, **Table Details** tab is displayed.
8. In **Table Name** field, enter the table name to store data and Click **Submit**. The configuration will be saved.
9. As per the configuration in the above screenshot, the incoming data is split and displayed in two tables: **orders** and **customers**.
10. Go back to the [Pipeline Configuration Page](#) to know how to deploy and run the pipeline.

GCS

Last updated on January 19, 2024

GCS Sink is used to load the structured data to GCS server.

 This page provides step-wise configuration about the Data Sink. After you configure the Data Sink, go back to the [Pipeline Configuration Page](#) to know how to **schedule your pipeline**.

Note:

Before you configure, provide the connection link from any Transformation Operator to the Data Sink.

To configure GCS as Data Sink, you can [Create New Connection](#) or use [Existing Connections](#) from the list.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Sink**. Drag and drop **GCS** sink in canvas.
2. Provide an input link from the transformation operator.
3. Click vertical ellipsis, select **Edit** option.
4. **Configure GCS** window is displayed. Click on **Create New Connection**.
5. **Connection Details** tab is displayed. Complete the below details:
 - a. **General Details**:

- i. **Connection Name:** Provide the connection name for the GCS sink. Specify unique name for name field. If you specify the same name which is specified for another sink name, an error message is displayed.
 - b. **GCS Connection Details**
 - i. **Project ID:** Enter the Project ID where the GCS is available.
 - ii. **Auth JSON file:** Displays the preview of the JSON file.
 - iii. **Upload:** Allows you to upload a json file to authenticate the GCS connection.
 - c. **Test Connection:** Click Test Connection option to test the connection status. The connection status is displayed if the connection is successful or failed.
 - d. **Click Next.**
6. **Column Details** tab is displayed.

Complete the following details:

- a. **Bucket Name:** Enter the bucket name.
 - b. **Prefix:** Enter the path where the output file will be stored. If not specified, then it is saved at root bucket folder.
 - c. **File Format:** Select the file format from the drop-down. Available Options are CSV, JSON, ORC, and Avro.
 - d. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like Reference Tag, usage Tag and system internal tag which comes under default tags category and populated from backend.
 - i. **To add or Specify a tag, From Configure: RDBMS** window, select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
or
Place the cursor at Type to add new, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.
 - ii. **Usage** and **Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.
 - e. **Incoming Columns:** Displays the incoming columns from the Transformation Operator.
 - f. **Partition Columns:** Click **Add Field**, a row is added. All the **incoming columns** are available as drop down in the **Partition Name** field. Select the required column based on which the data must be partitioned while storing to GCS.
7. Click **Save**.

Here is a configured example:

Existing Connections Re-usability

Steps to use existing connection as follows:

1. To use the existing connection, select any of the them from **Recently Used** or **All Connections**.
2. If you want to edit the existing connections and use, then click edit icon and perform the modifications.

Note:

- If the selected connection is already used for any other sinks, then the modifications will impact those sinks connection as well.

3. Click **Next**.
4. **Configure GCS** column details tab is displayed.

Configure the table details:

- a. **Bucket Name:** Enter the bucket name.
 - b. **Prefix:** Enter the path where the output file will be stored. If not specified, then it is saved at root bucket folder.
 - c. **File Format:** Select the file format from the drop-down. Available Options are CSV, JSON, ORC, and Avro.
 - d. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like Reference Tag, usage Tag and system internal tag which comes under default tags category and populated from backend.
 - i. To add or Specify a tag, From Configure: RDBMS window, select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
or
Place the cursor at **Type to add new**, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.
 - ii. **Usage** and **Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.
 - e. **Incoming Columns:** Displays the incoming columns from the Transformation Operator.
 - f. **Partition Columns:** Click **Add Field**, a row is added. All the **incoming columns** are available as drop down in the **Partition Name** field. Select the required column based on which the data must be partitioned while storing to GCS.
5. Click **Save**.
 6. Go back to the [Pipeline Configuration Page](#) to know how to schedule the pipeline.

GCSTrino

Last updated on May 19, 2025

GCS Sink is used to load the structured data to GCP server and Trino data base.

 This page provides step-wise configuration about the **Data Sink**. After you configure the Data Sink, go back to the *Pipeline Configuration Page* to know how to **schedule your pipeline**.

Note:

- Before you configure, provide the connection link from any Transformation Operator to the Data Sink.
- Below are the alter table actions that can be performed in **GCS Trino Sink**.
 - Adding Column
 - Deleting Column.
 - Changing Column Name.

To configure GCSTrino as Data Sink, you can [Create New Connection](#) or use [Existing Connections](#) from the list.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Sink**. Drag and drop **GCS Trino sink in canvas**.
2. Provide an input link from the transformation operator.
3. Click vertical ellipsis, select **Edit** option.
4. **Configure GCSTrino window is displayed. Click on Create New Connection.**
5. In GCS Trino Data Sink, while creating new connection, if user enter already existing connection details, then it will shows connection uniqueness validation pop-up with existing connection name. It is implemented to avoid the duplication of datasets created in data catalog.
6. Configure: GCSTrino **Connection Details** tab is displayed.

Complete the below details:

a. General Details

- i. **Provide the connection name for the GCSTrino sink. Specify unique name for name field.**
If you specify the same name which is specified for another sink name, an error message is displayed.

b. GCS Connection Details

- i. **Project ID:** Enter the Project ID where the GCS is available.

- ii. **Region:** Enter the Region where you want to push data.
 - iii. **Bucket Name:** Enter the bucket name.
 - iv. **Warehouse Base Path:** Enter the warehouse directory.
 - v. **Auth JSON file:** Displays the preview of the JSON file.
 - vi. **Upload:** Allows you to upload a json file to authenticate the GCS connection.
 - c. **Trino Connection Details:** Provide the trino database server connection details.
 - d. **Hive Metastore Connection Details:** Provide the Hive url. Warehouse directory is auto populated.
 - e. **Test Connection:** Click test connection option to test the connection status. The connection status is displayed if the connection is successful or failed.
 - f. Click **Next**.
7. **Table Details** tab is displayed.

Complete the below details:

- a. **Schema Name:** Provide the schema details for the RDBMS.
 - b. **Table Name:** It displays the Table Name which you are expecting to be created according to the structure, and the file format.
 - c. **File Format:** Select the file format from the drop down list.
 - d. **Primary Key:** Selected column act as primary key for the table. Select the column from the dropdown list.
 - e. **Truncate and Load:** This feature allows you to delete the records of an existing table and load the new records to the same table structure.
 - f. **Merge File:** After selecting connection type as Parquet, the **Merge File** drop down is displayed.
 - g. **File Size (in MB):** After you enable the Merge File option, File Size (in MB) field box is displayed. Enter the size of the file to be merged.
 - h. **File Merge Interval (in Seconds):** Enter the required duration in seconds.
It will merge the file till it reaches to the provided size and then process. If the file size is not reached, it will wait until configured duration and will process. For example, if you provide file merge size as 10mb and file merge interval as 10 seconds. While merging incoming flow files if size reaches to 10mb it will start processing otherwise, it will wait until file merge interval provided 10 seconds and then it will start processing.
 - i. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like Reference Tag, usage Tag and system internal tag which comes under default tags category and populated from backend.
 - j. **To add or Specify a tag, From Configure:** RDBMS window, select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
- or

Place the cursor at **Type to add new**, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.

- ii. **Usage** and **Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.
- j. **Incoming Columns**: Displays the incoming columns from the Transformation Operator.
- k. **Partition Columns**: Click **Add Field**, a row is added. All the **incoming columns** are available as drop down in the **Partition Name** field. Select the required column based on which the data must be partitioned while storing to Trino.

Note:

If partition column is configured, then truncate and load is disabled.

- 8. Click **Save**.

Here is a configured example:

Existing Connections Re-usability

Steps to use existing connection as follows:

- 1. To use the existing connection, select any of the them from **Recently Used** or **All Connections**.
- 2. If you want to edit the existing connections and use, then click edit icon and perform the modifications.

Note:

- If the selected connection is already used for any other sinks, then the modifications will impact those sinks connection as well.

- 3. Click **Next**.
- 4. **Configure GCSTrino** table details tab is displayed.

Configure the table details:

- a. **Schema Name**: Provide the schema details for the RDBMS.
- b. **Table Name**: It displays the Table Name which you are expecting to be created according to the structure, and the file format.
- c. **File Format**: Select the file format from the drop down list.
- d. **Primary Key**: Selected column act as primary key for the table. Select the column from the dropdown list.

- e. **Truncate and Load:** This feature allows you to delete the records of an existing table and load the new records to the same table structure.
- f. **Merge File:** After selecting connection type as Parquet, the **Merge File** drop down is displayed.
- g. **File Size (in MB):** After you enable the Merge File option, File Size (in MB) field box is displayed. Enter the size of the file to be merged.
- h. **File Merge Interval (in Seconds):** Enter the required duration in seconds.
It will merge the file till it reaches to the provided size and then process. If the file size is not reached, it will wait until configured duration and will process. For example, if you provide file merge size as 10mb and file merge interval as 10 seconds. While merging incoming flow files if size reaches to 10mb it will start processing otherwise, it will wait until file merge interval provided 10 seconds and then it will start processing.
- i. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like Reference Tag, usage Tag and system internal tag which comes under default tags category and populated from backend.
 - i. **To add or Specify a tag, From Configure: RDBMS** window, select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
or
Place the cursor at Type to add new, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.
 - ii. **Usage and Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.
- j. **Incoming Columns:** Displays the incoming columns from the Transformation Operator.
- k. **Partition Columns:** Click **Add Field**, a row is added. All the **incoming columns** are available as drop down in the **Partition Name** field. Select the required column based on which the data must be partitioned while storing to Trino.

Note:

If partition column is configured, then truncate and load is disabled.

- 5. Click **Save**.
- 6. Go back to the [Pipeline Configuration Page](#) to know how to schedule the pipeline.

Hive

Last updated on January 19, 2024

Hive Reader can read the data from the hive external tables created on top different object stores like AWS S3, Minio, GCS and Microsoft Azure. Hive reader connects to hive metastore and query all the databases and respective tables. Once user selects a table Hive readers looks for the metadata of the table and reads the data into a data frame from respective object store location.

 This page provides step-wise configuration about the Data Sink. After you configure the Data Sink, go back to the *Pipeline Configuration Page* to know how to **schedule your pipeline**.

Note:

Before you configure, provide the connection from any Transformation Operator to the Data Sink.

To configure Hive as Data Sink, you can [Create New Connection](#) or use [Existing Connections](#) from the list.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Sink**. Drag and drop **Hive sink in canvas**.
2. Provide a connection link from the transformation operator.
3. Click vertical ellipsis, select **Edit** option.
4. **Configure: Hive** window is displayed. Click on **Create New Connection**.
5. **Connection Uniqueness Validation Feature for tabular Sinks:** In Hive Data Sink, while creating new connection, if user gives already existing connection details, then it will shows connection uniqueness validation pop-up with existing connection name. It is implemented to **avoid duplication of datasets created in data catalog**. It is implemented for table based sinks such as RDBMS, S3Presto, GCSTrino, Azure Blob Storage Trino and Hive Sinks.
6. **Connection Detail** tab is displayed.

Configure the following details:

a. **General Details:**

- i. **Connection Name:** Provide the connection name for the hive sink. Specify unique name for name field. If you specify the same name which is specified for any other sink name, an error message is displayed.

b. **Connection Details**

- i. **Connection URL:** Enter the URL of the machine where the Hive DB is available.
- ii. **Username:** Enter the username.
- iii. **Password:** Enter the database password.
- iv. **Database Name:** Enter the database name where the data has to be stored.
- v. **Default File System Host:** Enter the default file system host address.
- vi. **Default File System Port:** Enter the default file system port number.
- vii. **Namenode Host:** Enter the namenode host address.

- viii. **Namenode Port:** Enter the namenode port for the host address.
 - ix. **Thrift URL:** Enter the thrift url.
 - x. **Test Connection:** Click test connection option to test the connection status. The connection status is displayed if the connection is successful or failed.
 - xi. Click **Next**. **Collection Details** Tab is displayed.
7. Configure the following **Collection details**:
- a. **Directory:** Enter the path where the Hive DB resides.
 - b. **Table Name:** Enter the table name where the data has to be stored.
 - c. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like **Reference Tag**, **usage Tag** and **system internal tag** which comes under default tags category and populated from back end.
 - i. **To add or Specify a tag, From Configure: S3 window,** select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
 - or
 - Place the cursor at **Type to add new**, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.**
 - ii. **Usage** and **Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.
8. Click **Save**.

Here is a configured Example:

Existing Connections Re-usability

Steps to use existing connection as follows:

1. To use the existing connection, select any of the them from **Recently Used** or **All Connections**.
2. If you want to edit the existing connections and use, then click edit icon and perform the modifications.

Note:

If the selected connection is already used for any other sinks, then the modifications will impact those sinks connection as well.

3. Click **Next**.
4. **Configure Hive Collection Details** tab is displayed.

- a. **Directory:** Enter the path where the Hive DB resides.
- b. **Table Name:** Enter the table name where the data has to be stored.
- c. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like **Reference Tag**, **usage Tag** and **system internal tag** which comes under default tags category and populated from back end.
 - i. **To add or Specify a tag, From Configure: S3 window,** select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.

or

Place the cursor at **Type to add new**, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.

- ii. **Usage** and **Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.

5. Click **Save to save the connection.**

6. Go back to the [Pipeline Configuration Page](#) to know how to schedule the pipeline.

HDFS

Last updated on January 19, 2024

HDFS is a distributed file system that handles large data sets running on commodity hardware. It is used to scale a single Apache Hadoop cluster to hundreds of nodes.

 **This page provides step-wise configuration about the Data Sink.** After you configure the Data Sink, go back to the [Pipeline Configuration Page](#) to know how to **schedule your pipeline**.

Note:

Before you configure, provide the connection from any Transformation Operator to the Data Sink.

To configure HDFS as Data Sink, you can [Create New Connection](#) or use [Existing Connections](#) from the list.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Sink**. Drag and drop **HDFS sink** in canvas.
2. Provide a connection link from the transformation operator.
3. Click vertical ellipsis, select **Edit** option.

4. **Configure:** HDFS window is displayed. Click on **Create New Connection**.
5. **Connection Details** tab is displayed.

Configure the following details:

- a. **General Details:**
 - i. **Connection Name:** Provide the connection name for the HDFS sink. Specify unique name for name field. If you specify the same name which is specified for another sink name, an error message is displayed.
- b. **Connection Details:**
 - i. **Default File System Host:** Enter the default file system host address. Example: 10.116.34.34
 - ii. **Default File System Port:** Enter the default file system port number. Example: 8030
 - iii. **Namenode Host:** Enter the namenode host address. Example: 10.116.34.34
 - iv. **Namenode Port:** Enter the namenode port for the host address. 50070
 - v. **Test Connection:** Click test connection option to test the connection status. The connection status is displayed if the connection is successful or failed.
 - vi. Click **Next**. **Column Details** tab is displayed.
6. In **Column Details** tab complete below details:
 - a. **Collection Details:**
 - i. **Directory:** Enter the path url. Example: /home/newp5/msdin_hdfs.
 - ii. **Format Details:** Select the file format in which it has to be stored.
 - iii. **Save Mode:** Only for CSV file format. It auto-populates after selecting CSV file format.
 - iv. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like **Reference Tag**, **usage Tag** and **system internal tag** which comes under default tags category and populated from back end.
 - **To add or Specify a tag, From Configure: S3** window, select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
 - or
 - Place the cursor at **Type to add new**, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.
 - **Usage** and **Reference Specify Tag** can not be tagged same time. System will display error, if both tags are used same time.
 - b. **Incoming Columns:** Displays the incoming columns from the Transformation Operator.

- c. **Partition Columns:** Click **Add Field**, a row is added. All the **incoming columns** are available as drop down in the **Partition Name** field. Select the required column based on which the data must be partitioned while storing to HDFS. Here is an example of configured partition column **date_of_birth**. The data will be categorized based on the date of birth and will be stored in HDFS.

7. Click **Save**.

Here is a configured example:

Existing Connections Re-usability

Steps to use existing connection as follows:

1. To use the existing connection, select any of the them from **Recently Used** or **All Connections**.
2. If you want to edit the existing connections and use, then click edit icon and perform the modifications.

Note:

If the selected connection is already used for any other sinks, then the modifications will impact those sinks connection as well.

3. Click **Next**.
4. **Configure HDFS Column Details tab is displayed.**
Complete the below details:
 - a. **Collection Details:**
 - i. **Directory:** Enter the path url. Example: /home/newp5/msdin_hdfs.
 - ii. **Format Details:** Select the file format in which it has to be stored.
 - iii. **Save Mode:** Only for CSV file format. It auto-populates after selecting CSV file format.
 - iv. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like **Reference Tag**, **usage Tag** and **system internal tag** which comes under default tags category and populated from back end.
 - To add or Specify a tag, From **Configure: S3** window, select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
 - or
 - Place the cursor at **Type to add new**, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.
 - **Usage** and **Reference Specify Tag** can not be tagged same time. System will display error, if both tags are used same time.

- b. **Incoming Columns:** Displays the incoming columns from the Transformation Operator.
 - c. **Partition Columns:** Click **Add Field**, a row is added. All the **incoming columns** are available as drop down in the **Partition Name** field. Select the required column based on which the data must be partitioned while storing to HDFS. Here is an example of configured partition column **date_of_birth**. The data will be categorized based on the date of birth and will be stored in HDFS.
5. Click **Save** to save the connection.
 6. Go back to the [Pipeline Configuration Page](#) to know how to schedule the pipeline.

Kafka

Last updated on January 19, 2024

Kafka Sink is a data sink - file system output, which allows you to store the data into the configured Kafka file system database tables.



This page provides step-wise configuration about the **Data Sink**. After you configure the Data Sink, go back to the [Pipeline Configuration Page](#) to know how to **schedule your pipeline**.

Note:

Before you configure, provide the connection link from any Transformation Operator to the Data Sink.

To configure Kafka as Data Sink, you can [Create New Connection](#) or use [Existing Connections](#) from the list.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Sink**. Drag and drop **Kafka sink** in canvas.
2. Provide an input link from the Transformation operator.
3. Click vertical ellipsis, select **Edit** option.
4. **Configure Kafka** window is displayed. Click on **Create New Connection**.
5. **Connection Details** tab is displayed.
Complete the below details:
 - a. **General Details:**
 - i. **Broker Name:** Enter the name for the connection. Specify unique name for name field. If you specify the same name which is specified for another sink name, an error message is displayed.

b. **Broker Details:**

- i. **BrokerUrls:** Enter the url of the broker and press Enter key. You can add any number of broker URLs. Broker Urls should be in proper format. Example: 11.110.34.98:9092 or kafka:6080.
- ii. **Test Connection:** Click the Test Connection option to test the connection status. The connection status is displayed if the connection is Successful or Failed.

c. Click **Next**. **Topic Details** tab is displayed.

6. Configure **Topic Details:**

- a. **Topic Name:** Enter the topic(s) name.
- b. **File Format:** Select the file format to be stored in Kafka Sink. CSV, JSON, ORC, and Avro are supported.
- c. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like Reference Tag, usage Tag and system internal tag which comes under default tags category and populated from backend.
 - i. **To add or Specify a tag, From Configure: Kafka window,** select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
or
Place the cursor at Type to add new, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.
 - ii. **Usage** and **Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.

7. Click **Save**.

Here is a configured example:

Existing Connections Re-usability

Steps to use existing connection as follows:

1. To use the existing connection, select any of the them from **Recently Used** or **All Connections**.
2. If you want to edit the existing connections and use, then click edit icon and perform the modifications.

Note:

If the selected connection is already used for any other sinks, then the modifications will impact those sinks connection as well.

3. Click **Next**.
4. Configure Kafka **Topic Details** tab is displayed.

Complete the below details:

- a. **Topic Name:** Enter the topic(s) name.
 - b. **File Format:** Select the file format to be stored in Kafka Sink. CSV, JSON, ORC, and Avro are supported.
 - c. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like Reference Tag, usage Tag and system internal tag which comes under default tags category and populated from backend.
 - i. **To add or Specify a tag, From Configure: Kafka window,** select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
or
Place the cursor at **Type to add new**, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.
 - ii. **Usage and Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.
5. Click **Save**.
 6. Go back to the [Pipeline Configuration Page](#) to know how to schedule the pipeline.

Local File Sink

Last updated on January 19, 2024

Local File refers to Simple Local File Storage Service, which is an object storage service that offers storing large files into local file system i.e storing files into the PVC (persistent volume claim) mounts.

To configure Local File Sink:

1. In Canvas, expand **Data Sink**. Drag and drop **Local File** sink in canvas.
2. Provide an input link from the transformation operator.
3. Click vertical ellipsis, select **Edit** option.
4. **Configure: Local File** window is displayed.
5. Configure the Details:
 - a. **In Collection details**, configure below:
 - i. **Directory:** Enter the path/directory to store the output data. By default, local file system location is **/datastore/**. Maximum allowed characters is 40.

- For example: If you want to store the data in folder "sample", enter the directory as / **datastore/sample/**. Data will be stored in the sample folder.
 - ii. **File Format:** Select the file format from drop-down to store the output data. Available file format is **CSV**.
 - b. **Incoming Column:** Displays the incoming column from the transformation operator. Incoming Column has below details:
 - i. **SI. No:** Displays number of Column present in file.
 - ii. **Name:** Displays Column Name for specific type of data category e.g name, age, phone number etc.
 - iii. **Column Type:** Displays data type that specific column contains e.g Integer, String, Date Time etc.
6. Click **Save** to save the sink configuration.
7. Configured Example:
8. Go back to the [Pipeline Configuration Page](#) to know how to schedule the pipeline.

RDBMS

Last updated on July 29, 2024

The data available in table format can be stored in RDBMS Sink. Before you store data, you must connect to the respective database, fetch the required data tables. All the connection configurations are based on the selected database type. **Example: If you have selected MySQL, then you must configure the connections for MySQL database, where it resides.**

 **This page provides step-wise configuration about the Data Sink. After you configure the Data Sink, go back to the [Pipeline Configuration Page](#) to know how to schedule your pipeline.**

Note:

- You can connect the multiple nodes from transformation operator to a single RDBMS Sink at a time. Before you configure, provide the connection link from any Transformation Operator to the Data Sink.
- Below are the alter table actions that can be performed in **RDBMS** Sink.
 - **Alter table by removing primary key**
 - **Alter table by removing not null**
 - **Alter table by removing columns**
 - **Alter table by adding columns**
 - **Alter table by adding not null**
 - **Alter table by adding primary keys**
 - **Alter table by changing column name or data type.**

To configure RDBMS as Data Sink, you can [Create New Connection](#) or use [Existing Connections](#) from the list.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Sink**. Drag and drop **RDBMS sink** in canvas.
2. Provide an input link from the Transformation operator.
3. Click vertical ellipsis and select **Edit** option.
4. **Configure RDBMS** window is displayed. Click on **Create New Connection**.
5. **Connection Uniqueness Validation Feature for tabular Sinks:** In RDBMS Data Sink, while creating new connection, if user gives already existing connection details, then it will shows connection uniqueness validation pop-up with existing connection name. It is implemented to **avoid duplication of datasets created in data catalog**. It is implemented for table based sinks such as RDBMS, S3Presto, GCSTrino, Azure Blob Storage Trino and Hive Sinks.
6. **Connection Details** tab is displayed.

Complete the following details:

- a. **General Details:** Provide the connection name for the RDBMS sink. Specify unique name for name field. If you specify the same name which is specified for another sink name, an error message is displayed.
 - b. **Connection Details:** Provide the RDBMS connection details; Host, User Name, Password, Port, Database Name, and Database Type (Oracle, PostgreSQL, MySQL) where you want to push data.
 - c. **Test Connection:** Click the **Test Connection** option to test the connection status. The connection status is displayed if the connection is Successful or Failed.
 - d. Click **Next**.
7. **Table Details** tab is displayed.

Complete the following details:

- a. **Schema Name:** Provide the schema details for the RDBMS.
- b. **Table Name:** It displays the Table Name which you are expecting to be created according to the structure, and the file format.
- c. **Truncate and Load:** This feature allows you to delete the records of an existing table and load the new records to the same table structure.
- d. **Use UPSERT:** This feature allows you to insert a new record or update columns of an existing record by selecting the Use UPSERT checkbox. This functionality requires Primary Key, which is mandatory.

- e. **Bulk Copy:** This feature allows transferring or loading a large amount of data from source to sink in single operation by selecting checkbox. It allows to maximize performance and minimize the time required to transfer data.
- f. **File Size (In MB):** It is displayed when Bulk Copy is selected. System will compress large size files to the mentioned file size. Enter the file size.
- g. **File Merge Interval (in Seconds):** It is displayed when Bulk Copy is selected. Enter the required duration in seconds. It will merge the file till it reaches to the provided size and then process. If the file size is not reached, it will wait until configured duration and will process. For example, if you provide file merge size as 10mb and file merge interval as 10 seconds. While merging incoming flow files if size reaches to 10mb it will start processing otherwise, it will wait until file merge interval provided 10 seconds and then it will start processing.
- h. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like Reference Tag, usage Tag and system internal tag which comes under default tags category and populated from backend.
 - i. **To add or Specify a tag, From Configure: RDBMS window,** select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
or
Place the cursor at Type to add new, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.
 - ii. **Usage and Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.
- i. **Incoming Column:** Displays the incoming column from the transformation operator. Below are the column properties:
 - i. **SI. No:** Displays the sequence of the column.
 - ii. **Name:** Displays the name of the column.
 - iii. **Column Type:** Displays the type of data that column stores such as integer, decimal, date-time, string.
 - iv. **Primary Key:** Select if column to be identified uniquely.
 - v. **Not Null:** Select if column should not have empty field.
 - vi. **Index:** This feature helps to improve the speed of data retrieval operation for the selected index column, by reducing the amount of data needs to be searched.

Note:

- Any column selected as **Primary Key**, can not be selected as **Index** at same time; and vice-versa.
- If any column is selected as **Primary Key**, **Index** check-box will get disable. If you want to select **Index** of any column, first clear the **Primary Key** check-box then select **Index** check-box.
- **Indexing check box is disabled for Column Type Decimal.**

- Multiple column can be selected as index. For better practice, it is recommended to select 2-3 columns.
- For existing deployed pipelines, index column selection is enabled.

- j. **Apply Subscriber Grouping:** Select the checkbox if you want to apply subscriber grouping to the output table. For more information, refer to [Subscriber Grouping](#).
- k. **Partition Column:** This feature allows to divide table in smaller, manageable partitions. When table is partitioned, each partition is stored in a separate table. This helps to improve the query operation performance.

Note:

- In each table only **one partition is allowed**. If partition is already exist, **Add Field** will be disabled.
- **Partition is not allowed for DateTime column which is selected as Primary key**. In case, you select DateTime column as Primary Key for partition, primary key check box will get unchecked and disabled.
- **Partition column is supported for Column Type DateTime**. DateTime should not include hh:mm:ss format.
- Partition Column will get disabled, if **Truncate and Load, Use UPSERT check-box is selected**; and vice-versa.
- **In case partition column is created use Usage specify tag, else use Reference specify tag**.
- The maximum number of characters allowed in the table name of a partitioned table is 36, and for a non-partitioned table it is 63.
- For postgres usage tables, at any point partitioned data is processed for the range of +/- 60 days.
- **Pipeline scenarios with partition column:**
 - For non-deployed pipelines, you can add partition column and deploy.
 - Once pipeline is deployed, partition column can not be edited.
 - **For existing deployed pipelines, partition column can not be added**. In this case delete the existing sink and create with new Postgres Sink with new table name.
 - For existing deployed pipelines, if pipeline is redeploy without editing sink it will run with older configuration without partition column.
 - **After pipeline deployment with partition table, Alter table actions by adding or removing primary key functionality will be disabled**. To proceed with this scenario, you needs to delete the existing sink and redeploy the pipeline.

8. Click **Save**.

Here is a configured example:

Existing Connections Re-usability

Steps to use existing connection as follows:

1. To use the existing connection, select any of the them from **Recently Used** or **All Connections**.
2. If you want to edit the existing connections and use, then click edit icon and perform the modifications.

Note:

- If the selected connection is already used for any other sinks, then the modifications will impact those sinks connection as well.
- In RDBMS Data Sink, while reusing existing connection, click the **Table Details** button to navigate to the next tab.

3. Click **Next**.
4. Configure RDBMS **Table Details** Tab is displayed.

Complete the following details:

- a. **Schema Name:** Provide the schema details for the RDBMS.
- b. **Table Name:** It displays the Table Name which you are expecting to be created according to the structure, and the file format.
- c. **Truncate and Load:** This feature allows you to delete the records of an existing table and load the new records to the same table structure.
- d. **Use UPSERT:** This feature allows you to insert a new record or update columns of an existing record by selecting the Use UPSERT checkbox. This functionality requires Primary Key, which is mandatory.
- e. **Bulk Copy:** This feature allows transferring or loading a large amount of data from source to sink in single operation by selecting checkbox. It allows to maximize performance and minimize the time required to transfer data.
- f. **File Size (in MB):** It is displayed when Bulk Copy is selected. System will compress large size files to the mentioned file size. Enter the file size.
- g. **File Merge Interval (in Seconds):** It is displayed when Bulk Copy is selected. Enter the required duration in seconds. It will merge the file till it reaches to the provided size and then process. If the file size is not reached, it will wait until configured duration and will process. For example, if you provide file merge size as 10mb and file merge interval as 10 seconds. While merging incoming flow files if size reaches to 10mb it will start processing otherwise, it will wait until file merge interval provided 10 seconds and then it will start processing.
- h. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like Reference Tag, usage Tag and system internal tag which comes under default tags category and populated from backend.

- i. To add or Specify a tag, From **Configure: RDBMS** window, select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.

or

Place the cursor at **Type to add new**, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.

- ii. **Usage and Reference** Tag can not be tagged same time. System will display error, if both tags are used same time.
- i. **Incoming Column:** Displays the incoming column from the transformation operator. Below are the column properties:
 - i. **Sl. No:** Displays the sequence of the column.
 - ii. **Name:** Displays the name of the column.
 - iii. **Column Type:** Displays the type of data that column stores such as integer, decimal, date-time, string.
 - iv. **Primary Key:** Select if column to be identified uniquely.
 - v. **Not Null:** Select if column should not have empty field.
 - vi. **Index:** This feature helps to improve the speed of data retrieval operation for the selected index column, by reducing the amount of data needs to be searched.

Note:

- Any column selected as **Primary Key**, can not be selected as **Index** at same time; and vice-versa.
- If any column is selected as **Primary Key**, **Index** check-box will get disable. If you want to select **Index** of any column, first clear the **Primary Key** check-box then select **Index** check-box.
- Indexing check box is disabled for **Column Type Decimal**.
- Multiple column can be selected as index. For better practice, it is recommended to select 2-3 columns.
- For existing deployed pipelines, index column selection is enabled.

- j. **Apply Subscriber Grouping:** Select the checkbox if you want to apply subscriber grouping to the output table. For more information, refer to **Subscriber Grouping**.
- k. **Partition Column:** This feature allows to divide table in smaller, manageable partitions. When table is partitioned, each partition is stored in a separate table. This helps to improve the query operation performance.

Note:

- In each table only **one partition** is allowed. If partition is already exist, **Add Field** will be disabled.
- **Partition is not allowed for DateTime column which is selected as Primary key.** In case, you select DateTime column as Primary Key for partition, primary key check box will get unchecked and disabled.
- **Partition column is supported for Column Type DateTime.** DateTime should not include hh:mm:ss format.
- Partition Column will get disabled, if **Truncate and Load, Use UPSERT check-box** is selected; and vice-versa.
- In case partition column is created use **Usage** specify tag, else use **Reference** specify tag.
- The maximum number of characters allowed in the table name of a partitioned table is 36, and for a non-partitioned table it is 63.
- For postgres usage tables, at any point partitioned data is processed for the range of +/- 60 days.
- **Pipeline scenarios with partition column:**
 - For non-deployed pipelines, you can add partition column and deploy.
 - Once pipeline is deployed, partition column can not be edited.
 - **For existing deployed pipelines, partition column can not be added.** In this case delete the existing sink and create with new Postgres Sink with new table name.
 - For existing deployed pipelines, if pipeline is redeploy without editing sink it will run with older configuration without partition column.
 - **After pipeline deployment with partition table, Alter table actions by adding or removing primary key functionality will be disabled.** To proceed with this scenario, you needs to delete the existing sink and redeploy the pipeline.

5. Click **Save** to save the connection.

6. Go back to the *Pipeline Configuration Page* to know how to schedule the pipeline.

S3

Last updated on January 19, 2024

S3 stands for Amazon Simple Storage Service, which is an object storage service that offers industry-leading scalability, data availability, security, and performance. Amazon S3 provides management features so that user can optimize, organize, and configure data to meet specific business, organizational, and compliance requirements.



This page provides step-wise configuration about the Data Sink. After you configure the Data Sink, go back to the *Pipeline Configuration Page* to know how to schedule your pipeline.

Note:

Before you configure, provide the connection link from any Transformation Operator to the Data Sink.

To configure S3 as Data Sink, you can [Create New Connection](#) or use [Existing Connections](#) from the list.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Sink**. Drag and drop **S3 sink** in canvas.
2. Provide a connection link from the transformation operator.
3. Click vertical ellipsis and select **Edit** option.
4. **Configure: S3** window is displayed. Click on **Create New Connection**.
5. **Connection Details** tab is displayed.
Complete the following details:
 - a. **General Details:**
 - i. **Connection Name:** Provide the connection name for the S3 sink. Specify unique name for name field. If you specify the same name which is specified for another sink name, an error message is displayed.
 - b. **Connection Details:** Enter **Bucket Name**, **EndPointUrl**, **AccessKeyID**, **SecretKey**, and select **Region** where you want to push data.
 - c. **Test Connection:** Click test connection option to test the connection status. The connection status is displayed if the connection is successful or failed.
 - d. Click **Next**. **Column Details** is displayed.
6. Configure the following in **Column Details** tab:
 - a. **Collection Details:** Enter **Prefix** and select the file format in which data has to be stored.
 - b. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like **Reference Tag**, **usage Tag** and **system internal tag** which comes under default tags category and populated from back end.
 - i. **To add or Specify a tag, From Configure: S3 window**, select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
or
Place the cursor at **Type to add new**, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.
 - ii. **Usage and Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.

- c. **Incoming Columns:** Displays the incoming columns from the Transformation Operator.
- d. **Partition Columns:** Click **Add Field**, a row is added. All the **Incoming columns** are available as drop down in the **Partition Name** field. Select the required column based on which the data must be partitioned while storing to S3.

7. Click **Save**.

Here is a configured example:

Existing Connections Re-usability

Steps to use existing connection as follows:

1. To use the existing connection, select any of the them from **Recently Used** or **All Connections**.
2. If you want to edit the existing connections and use, then click edit icon and perform the modifications.

Note:

If the selected connection is already used for any other sinks, then the modifications will impact those sinks connection as well.

3. Click **Next**.
4. **Configure S3 Column Details** tab is displayed.

Complete the details:

- a. **Collection Details:** Enter Prefix and select the file format in which data has to be stored.
- b. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like Reference Tag, usage Tag and system internal tag which comes under default tags category and populated from backend.
 - i. **To add or Specify a tag, From Configure: S3 window,** select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
or
Place the cursor at **Type to add new**, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.
 - ii. **Usage and Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.
- c. **Incoming Columns:** Displays the incoming columns from the Transformation Operator.

- d. **Partition Columns:** Click **Add Field**, a row is added. All the **incoming columns** are available as drop down in the **Partition Name** field. Select the required column based on which the data must be partitioned while storing to S3.
5. Click **Save**.
6. Go back to the [Pipeline Configuration Page](#) to know how to schedule the pipeline.

S3Presto

Last updated on January 19, 2024

Presto is an open source, distributed SQL query engine designed from the ground up for fast analytic queries against data of any size.

 This page provides step-wise configuration about the Data Sink. After you configure the Data Sink, go back to the [Pipeline Configuration Page](#) to know how to **schedule your pipeline**.

Note:

- You can connect the multiple nodes from transformation operator to a single **S3Presto Sink** at a time. Before you configure, provide the connection link from any Transformation Operator to the Data Sink.
- Below are the alter table actions that can be performed in **S3Presto Sink**.
 - Adding a column
 - Deleting a column
 - Changing column name

To configure S3Presto as Data Sink, you can [Create New Connection](#) or use [Existing Connections](#) from the list.

Create New Connection

Steps to create new connection as follows:

1. In Canvas, expand **Data Sink**. Drag and drop **S3Presto sink** in canvas.
2. Provide an input link from the transformation operator.
3. Click vertical ellipsis, select **Edit** option.
4. Configure S3Presto window is displayed. Click on Create New Connection.
5. **Connection Uniqueness Validation Feature for tabular Sinks:** In S3 Presto Data Sink, while creating new connection, if user gives already existing connection details, then it will shows connection uniqueness validation pop-up with existing connection name. It is implemented to avoid duplication of datasets created in data catalog. It is implemented for table based sinks such as RDBMS, S3Presto, GCSTrino, Azure Blob Storage Trino and Hive Sinks.

6. **Connection Details** tab is displayed.

Complete the following details:

a. **General Details:**

- i. **Connection Name:** Provide the connection name for the S3Presto sink. Specify unique name for name field. If you specify the same name which is specified for another sink name, an error message is displayed.

- b. **S3 Connection Details:** Provide the S3 connection detail like Bucket Name, End Point URL, Access Key ID, Secret Key, Warehouse Base Path, Prefix, and select Region where you want to push data.

- c. **Presto Connection Details:** Provide the trino database server connection details.

- d. **Hive Metastore Connection Details:** Provide the Hive url. Warehouse directory is auto populated.

- e. **Test Connection:** Click test connection option to test the connection status. The connection status is displayed if the connection is successful or failed.

- f. **Click Next. Table Details** is displayed.

7. Configure the following in **Table Details** tab:

- a. **Schema Name:** Provide the schema details for the RDBMS.

- b. **Table Name:** It displays the Table Name which you are expecting to be created according to the structure, and the file format.

- c. **File Format:** Select the file format from the drop down list.

- d. **Primary Key:** Selected column act as primary key for the table. Select the column from the dropdown list.

- e. **Truncate and Load:** This feature allows you to delete the existing records and load the new records with in the same table structure.

- f. **Merge File:** After selecting the file format type, this option is displayed. If enabled, one more field **File Size** will be displayed.

- g. **File Size (in MB):** After you enable the Merge File option, **File Size (in MB)** field box is displayed. Enter the size of the file to be merged.

- h. **File Merge Interval (in Seconds):** Enter the required duration in seconds. It will merge the file till it reaches to the provided size and then process. If the file size is not reached, it will wait until configured duration and will process. For example, if you provide file merge size as 10mb and file merge interval as 10 seconds. While merging incoming flow files if size reaches to 10mb it will start processing otherwise, it will wait until file merge interval provided 10 seconds and then it will start processing.

- i. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like Reference Tag, usage Tag and system internal tag which comes under default tags category and populated from backend.

- j. **To add or Specify a tag, From Configure: S3Presto** window, select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The

matching tags are populated as dropdown. Select required tag.

or

Place the cursor at **Type to add new**, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.

- ii. **Usage** and **Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.
- j. **Incoming Columns**: Displays the incoming columns from the Transformation Operator.
- k. **Partition Columns**: Click **Add Field**, a row is added. All the **incoming columns** are available as drop down in the **Partition Name** field. Select the required column based on which the data must be partitioned while storing to S3Presto.

Note:

If partition column is configured in S3 presto, then truncate and load is disabled.

- 8. Click **Save**.

Here is a configured example:

Existing Connections Re-usability

Steps to use existing connection as follows:

- 1. To use the existing connection, select any of the them from **Recently Used** or **All Connections**.
- 2. If you want to edit the existing connections and use, then click edit icon and perform the modifications.

Note:

If the selected connection is already used for any other sinks, then the modifications will impact those sinks connection as well.

- 3. Click **Next**.
- 4. **Configure S3Presto Table Details** tab is displayed.

Complete the below details:

- a. **Schema Name**: Provide the schema details for the RDBMS.
- b. **Table Name**: It displays the Table Name which you are expecting to be created according to the structure, and the file format.
- c. **File Format**: Select the file format from the drop down list.

- d. **Primary Key:** Selected column act as primary key for the table. Select the column from the dropdown list.
- e. **Truncate and Load:** This feature allows you to delete the existing records and load the new records with in the same table structure.
- f. **Merge File:** After selecting the file format type, this option is displayed. If enabled, one more field **File Size** will be displayed.
- g. **File Size (in MB):** After you enable the **Merge File** option, **File Size (in MB)** field box is displayed. Enter the size of the file to be merged.
- h. **File Merge Interval (in Seconds):** Enter the required duration in seconds. It will merge the file till it reaches to the provided size and then process. If the file size is not reached, it will wait until configured duration and will process. For example, if you provide file merge size as 10mb and file merge interval as 10 seconds. While merging incoming flow files if size reaches to 10mb it will start processing otherwise, it will wait until file merge interval provided 10 seconds and then it will start processing.
- i. **Specify Tags:** Specifying or adding tags differentiates the tables based on tags. There are different tags like **Reference Tag**, **usage Tag** and **system internal tag** which comes under default tags category and populated from backend.
 - i. **To add or Specify a tag, From Configure: S3Presto** window, select **Column Details** tab and place the cursor at **Type to add new** under **Specify Tags** pane and enter a tag name. The matching tags are populated as dropdown. Select required tag.
or
Place the cursor at **Type to add new**, enter the required tag name and hit enter. The required tag will get selected. You can only use alpha numeric characters and space when adding a tag.
 - ii. **Usage** and **Reference** Specify Tag can not be tagged same time. System will display error, if both tags are used same time.
- j. **Incoming Columns:** Displays the incoming columns from the Transformation Operator.
- k. **Partition Columns:** Click **Add Field**, a row is added. All the **incoming columns** are available as drop down in the **Partition Name** field. Select the required column based on which the data must be partitioned while storing to S3Presto.

Note:

If partition column is configured, then truncate and load is disabled.

- 5. Click **Save**.
- 6. Go back to the [Pipeline Configuration Page](#) to know how to schedule the pipeline.

BMS

Overview

Overview

Last updated on May 12, 2023

Business Modelling Studio (BMS) enables batch and streaming business rule processing engines, **which helps you to seamlessly build batch and data pipelines.**

It is a simple graphical user interface, which is used for creating and maintaining pipelines. You will **be able to configure new source/sink connections, new streams, pipelines and will be able to manage tasks from the GUI** (Provided they have necessary rights).

The existing data sources or pipelines can be edited or updated and be reused based on requirements.

Data from any source is added into the system and enriched or transformed into the required format, which can be used by other products/studios.

A pipeline can have the following operators:

1. **Data Source:** Allows you to fetch the data from any database or upload the data, based on operator capabilities. Following are the operators supported: DSV Source, Database, Kafka Source, and S3 Source.
2. **Transform Operators:** Allows you to transform the data based on the operators configured. Following are the operators supported: Filter, Reducer, Mapper, Formula, Summarizer, Join, and Query Measure.
3. **Data Sink:** Allows you to store the transformed data into any database. Following are the operators supported: DSV Sink, Database, Kafka Sink, and S3 Sink.

Note:

A pipeline with query measure cannot have any other operators, as it fetches data from Data Catalog.

Data is processed in the following modes:

1. **Streaming:** Data is fetched from the configured data source.
 - Batch processing mode (DSV Source, Database)
 - Streaming processing (Kafka Source, S3 Source)
2. **Query Measure :** It takes the data from the Data Catalog and helps the users to write join scenarios (inner, outer, left and right).

Note:

Workspace

Workspace Basics

Last updated on May 13, 2025

The landing page of Business Modelling Studio is a Pipeline Repository and has the following panes as listed below:

1. **Left Menu Bar:** It includes the list of Studios that you have access to, user account information, access to Home Page, and access to Help document.
2. **Operational Summary:** It displays the pipelines status with differentiated category for the particular user login. The categories are Completed Data Pipelines, Finished Data Pipelines, Stopped Data Pipelines, Draft Data Pipelines, Failed Data Pipelines and Running Data Pipelines.
3. Action items:
 - a. **Setting:** This icon contains the list of other features. Click to view the dropdown list. Below features are available:
 - i. **Flooding Control:** This feature allows you to set the alarm and notify the user whenever cases generated from the rule engine breach the set threshold limit. It helps the user to be more vigilant against fraud. For more information regarding how to set the alarm, see [Flooding Control](#).
 - ii. **API Logs:** This feature allows you to view the logs related to API registered with Centralized API framework. When you click **API Logs**, data table for various API log is displayed

Select the serial number of API log to perform below actions:

- **Copy:** It allows you to copy the log details.
- **Retry:** It allows you to retry the API request.
- You can use toolbar to perform actions such as Share, Basic Filter, Advance Filter, Fx Operation and so on. For more information, see [Toolbar Options](#).
- iii. **Global Exception:** This feature allows you to create the global exception list upon which rules are not applicable. For more information, see [Global Exception](#).
- b. **Refresh Pipeline Status Icon:** This icon allows you to update the pipeline(s) status. Click the refresh icon to update the status of the list of pipeline(s), it refreshes and displays the updated status of pipelines.
- c. **Rule Export Icon:** This icon helps you to download the configured rules. When you click the icon, excel file is downloaded which includes details of all the configured rules. You can open the excel and view the Pipeline, Data Reader, Transform Operator, and Data Writer configured details.

- d. **Triggered Alert Icon:** This icon helps you to view the triggered alerts. For more information, see [Triggered Alerts](#).
 - e. **Context-Sensitive Help Icon:** This is an online help that is obtained from a specific point in the state of the software, providing help for the situation that is associated with the particular process step.
 - f. **Create New Pipeline icon:** Add Icons to create a new pipeline. For more information, see [Create New Pipeline](#).
4. **Pipeline Repository:** List of pipelines created in the system. This feature allows the user to view and access the respective pipelines that a user is configured to view and access.
 5. **Search Bar:** To search the pipeline information.

Pipeline Repository

The Pipeline Repository has all the configured pipelines listed. To process any data, a [pipeline must be created](#).

Viewing Pipelines

It allows you to view the Batch pipeline(s) in the different display format. You can change the pipeline(s) as list view and grid view.

- : Click to view the pipelines in the list view. By default, pipelines are listed in list view.
- : Click to view the pipelines in the grid view.

Grid View

Pipeline Actions on Landing Page

Following are the functions which are associated with the pipeline in the landing page:

- : It allows you to schedule the pipelines. The pipelines are executed in the scheduled time and the data on data sources is processed dynamically based on the schedules. For more information, refer to [Schedule Pipeline](#).
- **Start/Stop:** After any pipeline is created, it will be in the draft status, you must have to run and stop the pipeline to generate the results of any configurations performed.

To stop and run the pipeline:

- : It allows you to Start or run the pipeline.
- : It allows you to Stop the running pipeline.
- : It allows you to Resume the configured pipeline from the stopped or failed point.

Note :

The resume function is applicable only for batch pipelines. It is not applicable for stream pipelines.

- Click Vertical Ellipses from the particular row of the pipeline to perform below actions:

- **View:** Allows you to view the active version of the pipeline.

Note :

- You cannot edit and save any data information in the view mode.
- You can view the KPI pipeline configuration details by selecting the **View KPI**. This is applicable for the **Query Measure** and **SQL Measure** only.

To view KPI configuration, follow below steps:

- In the pipeline, click vertical ellipses.
- Select the **View KPI**. The application displays the data inside the pipeline.

- **Edit:** Allows you to edit the pipeline information.

Note:

- When you click edit, you are navigated to latest version of the pipeline.

- **Version History:** Allows you to track and manage the different versions of same pipeline. For more information, see [Version History](#).
- **Schedule:** Allows you to schedule the pipeline for run. Scheduler configuration is applied across all the versions.
- **Rename:** Allows you to rename or update the new name of the pipeline. It is applied across all the versions. For more information, refer to [Rename Pipeline](#).
- **Delete:** Allows you to delete the pipeline along with all the versions from the Pipeline Repository.
- **Share:** It allows you to share the pipeline along with all the versions with other user or user group within an organization. For more information go to [Share Pipeline](#).
- **Migrate:** It allows you to migrate the data available in the usage table (reporting and run-history table) with partition columns. It improves the rerun performance at the pipeline or measure level.

Note:

- After successfully migrating the data, the Migrate option becomes disabled.

- The migrate option is available for pipelines that have failed to migrate.
- When you click Migrate, the following pipeline statuses appear:
 - When you click Migrate, the pipeline status is displayed as "**Migrating**".
 - When migration is successfully completed, the pipeline status changes to "**Finished**".
 - When migration fails, the pipeline status is displayed as "**Failed**".
- During migration, if the partition column is available, the system partitions the record using the combination of run ID and partition column. If the partition column is not available, system create a partition with the run ID to complete the migration.

Pipeline - Expanded View

1. Click or click on the pipeline in the list to expand it to view in detail. The expanded pipeline is viewed as below:
2. To learn more, go back to the [Configure a Pipeline](#) Page.

Configure a pipeline

Last updated on May 12, 2023

Business modeling refers to the process of creating, analyzing, and communicating models or representations of various aspects of a business. It involves using different tools, techniques, and frameworks to understand, visualize, and design how a business operates.

The purpose of business modeling is to gain a deeper understanding of the organization's structure, processes, strategies, and interactions, and to identify opportunities for improvement or innovation. Below are the sequence of operation in which Business Modelling is achieved using HyperSense.

1. Create a new Workflow

Creating a workflow refers to designing and establishing a systematic flow of activities or processes that align with the business model. A workflow of the pipeline represents the sequential steps or stages through which a product, service, or information moves from one point to another within the **business**. It helps ensure that tasks or activities are executed in a structured and organized manner, minimizing delays, bottlenecks, and errors.

The first step is to navigate to the **Business Modeling Home** page. For more information, refer to the [Create New Pipeline](#).

2. Online Pipeline(Streaming)

Tip - If you choose the **Online (streaming pipeline)** option in the Choose Processing Mode while configuring the Create New Workflow, then this **step** applies to you.

a. Configure a Data Reader

The second step is to configure the **Data Reader**. A data reader refers to a component or module that **is responsible for reading and extracting data from various types of sources to be used in the modeling process**. It is a mechanism or tool used to gather relevant data from internal and external sources and convert it into a format suitable for analysis and modeling.

In HyperSense's a "Data Reader" gathers and extracts data from a variety of sources, including databases, spreadsheets, files, and other data repositories. **It enables the business modeler to access the necessary data required for their modeling activities**. There are different types of Data reader **for more information, refer to the [Data Reader Overview](#)**.

HyperSense integrates with a variety of readers including databases, files, message queues, or other data repositories which are explained below:

- **Relational databases** : Data can be extracted from database systems such as Postgres. The pipeline process connects to the database, executes queries, and retrieves the required data. For more information about data base configuration, refer to [Database](#).
- **File System** : Data can be sourced from files in formats like DSV (De-limiter Separated Values), CSV (comma-separated values), Excel spreadsheets, JSON (JavaScript Object Notation), or other format files. The pipeline reads the files and extracts the relevant information. For more information about files configuration, refer to [DSV](#).
- **Cloud storage** : Data can be stored in cloud-based storage systems like Amazon S3, Google Cloud Storage, Azure Blob Storage or Hive. The audit pipeline connects to these storage systems, accesses the data files, and performs the extraction. For more information about cloud storage configuration, refer to [Amazon S3](#), [Google Cloud Storage](#) or [Azure Blob Storage](#).
- **Message queues** : Some pipelines may source data from message queues such as Apache Kafka. The pipeline consumes messages from the queue and processes the data they contain. In real-time or near-real-time pipeline scenarios, data sources can be streaming platforms like Apache Kafka. The pipeline continuously processes data as it arrives. For more information about configuration, refer to [Kafka](#).

b. Transformation Operator

In the third step configure a Transformation Operator. A Transformation Operator refers to a component that performs data cleansing, validation, normalization, aggregation, enrichment on the data as it moves through the pipeline. These operators are responsible for transforming the data from its source format to a desired format, applying various operations or calculations, and preparing it for further processing or loading into the target system.

In HyperSense, "Transform Operators" carry out actions or operations that are applied to data or inputs to produce desired results. In **business modeling, this could involve applying various mathematical, logical, or analytical transformations to input data to generate insights, forecasts, or decision-making outputs**.

Data transformation operators in HyperSense can perform a wide range of operations, including:

- **Filtering:** The operator selects or filters specific data records or attributes based on defined criteria. This allows for the extraction of relevant data while discarding unnecessary information. For more information detailed configuration, see [Filter](#).
- **Reducer :** What ever user selects, it keeps the the selected columns and reduce the column which are unselected. For more information on detailed configuration, see [Reducer](#).
- **Mapper :** Schema Mapper helps you map your Source schemas to tables in your Destination (Sink). Mapping operators transform data values from one representation or format to another. They can be used to standardize data formats, convert data types, or map values from different source systems into a unified format. For more information on detailed configuration, see [Mapper](#).
- **Formula :** A "Formula" typically represents a mathematical or logical expression that defines a relationship or calculation between variables. In the context of a transform operator within a business module, a formula can be used to define the specific transformation or operation to be applied to the input data. For more information on detailed configuration, see [Formula](#).
- **Join :** Join operator combine the data from multiple sources based on common keys or attributes. They enable the consolidation of data from different tables or datasets into a single dataset for further processing. For more information on detailed configuration, see [Join](#).
- **Summarizer :** In this interpretation, a "Summarizer" would refer to a module or component that performs automatic text summarization within the business module. It would involve the application of summarization techniques to extract key information or generate concise summaries from textual data related to the business domain. For more information on detailed configuration, see [Summarizer](#).

c. Data Writer

In the third step, you need to configure **Data Writer**. A data writer refers to a component or module that is responsible for storing, recording, or outputting data generated from the modeling process. It facilitates the transfer of data from the modeling environment to a designated destination or output format.

In **HyperSense**, a "Data Writer" is responsible for saving or presenting the results, insights, or outputs obtained from modeling activities in a structured and usable format. It allows for the persistence of data, making it available for further analysis, reporting, or decision-making.

Data writers in HyperSense can perform a variety of outputs, which includes:

- **Relational Database:** Relational databases, such as Postgres are commonly used as Data Sinks. Data can be stored in tables with predefined schemas, allowing for efficient querying and structured data storage. For more information about data base configuration, refer to [Database Sink](#).
- **File System :** A File system refers to a destination or target where data is written or stored in a file system. It involves saving or persisting data into files within a file storage system. For more information about File System configuration, refer to [DSV Sink](#).
- **Message Queuing Systems:** Message queuing systems like Apache Kafka can act as Data Sinks when the focus is on real-time streaming or event-based data processing. Data is written to top-

ics or queues for consumption by downstream systems or applications. For more information about Kafka configuration, refer to [Kafka Sink](#).

- **Cloud Data Lakes:** Cloud-based data lakes, such as Amazon S3, Azure Blob Storage, S3Presto, Azure Blob Storage Trino, offer scalable and cost-efficient storage for both structured and unstructured data. Data Sinks like data lakes provide a flexible and schema-on-read approach, enabling data exploration and analysis. For more information about cloud storage configuration, refer to [Amazon S3](#), [Azure Blob Sink](#), [S3 External - Table Sink](#), [Azure Blob External - Table Sink](#), [Google Cloud Storage \(GCS\) Sink](#), [GCS External - Table Sink](#).
- **Case Operator :** A "Case Operator Sink" could potentially refer to a component or functionality that handles different cases or scenarios within a business module. It could involve conditional logic or decision-making based on various conditions or inputs. **For more information on configuration of Case Operator Sink, see [Case Operator Sink](#).**
- **Notification :** A "Notification Sink" refers to a destination or target where notifications or alerts are sent or stored. A notification sink is typically used to receive and handle notifications generated by a system or application. It can be an email server, a messaging service, a database, or any other mechanism that accepts and stores notifications for further processing or delivery to the intended recipients. For more information on configuration of Notification Sink, see [Notification Sink](#).

Once the sink has **successfully** set up, move on to the schedule and run. Refer to [step 6](#) to discover how to Schedule and Run.

3. Offline Pipeline (Batch Measures)

Tip - If you choose the **Offline(Batch Measures)** option in the Choose Processing Mode while configuring the Create New Workflow, then this step applies to you.

The second step is to configure the Offline(Batch measures) pipeline(SQL, QM, DMM). The Batch measures in business modeling involve aggregating and analyzing data over a specific time period or a defined set of records. Instead of processing data in real-time or continuously, batch measures are calculated periodically, such as hourly, daily, weekly, or monthly.

Below are the defined offline batch measure transform Operators:

- **Query Measure :** Query measure refers to the process of retrieving data from a database or data source and calculating quantitative values or metrics to assess various aspects of the data. It involves querying the data based on specific criteria or conditions and deriving measures or metrics from the queried data. For more information on configuration of QM, see [Query Measure](#).

Info - Query Measures and SQL Measure provide data to Key Performance Indicators (KPIs). To configure KPI, refer to [KPI](#).

- **SQL Measure :** SQL (Structured Query Language) typically refers to a calculated value or metric derived from data within a database. SQL provides various functions and operators that allow you to perform calculations and derive measures from the data stored in tables. For more information on configuration of SQL Measure, see [SQL Measure](#).

Info - Query Measures and SQL Measure provide data to Key Performance Indicators (KPIs). To configure KPI, refer to [KPI](#).

- **Data Match Measure** : Data match measure refers to a metric or evaluation criterion used to assess the quality and effectiveness of the data matching process. It provides a quantitative measure of the accuracy, completeness, or reliability of the matched data. For more information on configuration of Data Match Measure, see [Data Match Measure](#).
- **Custom Operator** : A user-defined or custom-built operation or function within a data processing or business module that is tailored to meet specific requirements or perform specialized tasks. The purpose of using custom operators is to enhance the capabilities of the data processing pipeline and accommodate specific business logic, algorithms, or transformations that are specific to the domain or application. Custom operators allow businesses to tailor the data processing flow to their specific requirements and unlock the full potential of their data. For more information on configuration of Custom Operator, see [Custom Operator](#).

In the Offline, the Query Measures and SQL Measure provide data to Key Performance Indicators (KPIs).

KPI : KPI stands for Key Performance Indicator. A KPI is a measurable metric or indicator that helps businesses assess their performance and progress towards specific goals or objectives. KPIs provide quantifiable measurements that enable organizations to track, evaluate, and manage various aspects of their operations. For more information on detailed configuration, see [Navigation of BMS, KPI Page, KPI](#).

Connect the Operators as per the required output and save the pipeline. to learn Schedule and run, refer to [Schedule and Run](#) pipeline.

4. Schedule, Run and Report

Run: The running phase involves the execution and monitoring of the deployed data pipeline. Once the infrastructure is set up and the pipeline is deployed, it starts processing data according to the defined workflow. During the run phase, data is ingested, transformed, and analyzed based on the predefined tasks and processing logic.

Tip - If the pipeline is copied, then by default pipeline will be saved in **Drafted** status. In the Draft status, pipeline Run and schedule function will be disabled status. So make sure the pipeline must be in **Completed** status.

1. The newly created pipeline is listed in the [Pipeline Repository](#) page in **Completed** state. You need to [schedule the pipeline](#) in order to Run the pipeline.
2. Once the pipeline is scheduled, the pipeline can be [Run](#) in order to generate the result. A confirmation prompt is displayed after the successful deployment of the pipeline.
3. HyperSense provide features and connectors specifically designed for [importing and exporting](#) data, facilitating seamless data movement and integration across different systems within the pipeline.

5. Preview and Report

Preview: A preview is a glimpse or sneak peek of data that allows you to see or experience a part of it before the full version is available.

Report: A report is the outcome or consequence of a specific action, process, or event. It is what happens as a consequence of a particular cause.

1. The generated outputs of the Online processing mode are stored or reflected in various forms, as shown below:
 - The Database, S3 External Sink, Azure External Sink and GCS External Sink table outputs are loaded in *Data Catalog* module of HyperSense.
 - DSV stores its output in the system.
 - S3 sink **generates the output** and store them in the S3 cloud.
 - Azure sink **generates the output** and stores it in the Azure Cloud.
 - GCS sink **generates output and stores in GCS Cloud.**
 - Notification sink **generates output in the form of E-mails.**
 - Kafka sink output generates and stores. It can be viewed in System storage.
 - The case operator generates cases as output in the PAS.
2. The generated outputs of the **Offline** processing mode are stored or reflected in various forms, as shown below:
 - The Batch measure result can be viewed in preview option in the Pipeline configuration screen. Form more refer to *Result Preview Features*.
 - Batch measure outputs (QM, SQLM and DMM) generates table outputs are loaded in *Data Catalog* module of HyperSense.
 - For Custom Operator, the output is generated based on the customer configuration.
 - HyperSense provide the feature of **preview**, which displays the preliminary or early version of something that provides a glimpse or insight into what the final version will look like or include.

BMS Navigation

Last updated on May 12, 2023

From the Hypersense Home Page, you can either navigate to the Pipeline Repository or KPI listing page.

1. From any page on the HyperSense, move the cursor towards the left side of the screen. The left navigation menu is displayed, go to **BMS** menu.

2. Click Pipelines, to navigate to Pipeline Repository page.
OR
Click KPI List, to navigate to KPI Listing page.

3. To learn more refer to below topics:

- To learn more about KPI Configuration and basic functions, refer to [Configuration KPI](#) and [KPI Home page Features](#).
- To learn more about basic functions in Pipeline Repository, refer to [Pipeline Repository](#).
- To learn more about how to configure pipeline, refer to [Configure a Pipeline](#).

KPI Page

Last updated on May 12, 2023

The KPI Page of BMS has the list of KPIs generated in the system and their current status.

You can view the following information on KPI page:

1. **Violation:** Displays the count of records violated, for the respective KPI.
2. **Leakage:** Displays the computed leakage based on expression configured in the KPI window.
3. **Min Threshold:** Displays the minimum KPI threshold, on which a violation has occurred.
4. **Max Threshold:** Displays the maximum threshold, on which a violation has occurred.
5. **Run Date Selector:** Allows you to select the dates to view KPIs.
 - a. **View Only violated:** If enabled, it displays only the violated KPIs.
 - b. **Run Start Date:** Indicates the start date of KPI run.
 - c. **Run End Date:** Indicates the end date of KPI run.
 - d. To view the KPIs in the selected dates:
 - Select the start date and end date.
 - Click **Apply**.
6. **Total Cases:** Displays the number of cases generated for a KPI violated.
7. **Last Run:** Displays the date, when the KPI was last run.
8. To know how to View and Filter, refer to [Viewing and Filter KPIs](#).

Viewing and Filter KPIs

- : Click to view the KPIs in the list view. By default, KPIs are listed in list view.
- : Click to view the KPIs in the grid view.
- **Filter:** Filter feature allows you to refine and filter the data of the asset by search. Assets are searched on the basis of criteria given in the fields. For more details, refer [Filter](#).
- To know KPI expand feature, refer to [KPI Expanded View](#).

KPI Expanded View

Click the KPI to expand and view the details.

The expanded view displays the following details in table format:

- **Start Date:** Displays the start date of KPI run. You can configure the required date format through the [Common Settings](#) available in Admin settings of HyperSense.
- **End Date:** Displays the end date of KPI run.
- **Run Date:** Displays the actual date of KPI run.
- **Dimensions:** Displays the dimensions involved for KPI violation.
- **Threshold:** Displays the threshold of KPI violation.
- **Value:** Displays the value at which violation has occurred.
- **Result:** Displays the result for the scheduled instance. The violated records are viewed with red alert and the non-violated records are viewed with the tick mark.
- **View Case:** A case is generated on KPI violation, if a case template is attached on KPI configuration. Click to view the case generated. You the output in the PAS.
- **Graph:** On right hand side, you can view the graph plotted. The graph is displayed for KPI run date vs Number of violations per run.
- To learn more about KPI Configuration, refer to [Configuration KPI](#).
- To learn more about how to configure pipeline, refer to [Configure a Pipeline](#).

Features

Pipeline

Pipeline Repository Actions

[Last updated on May 12, 2023](#)

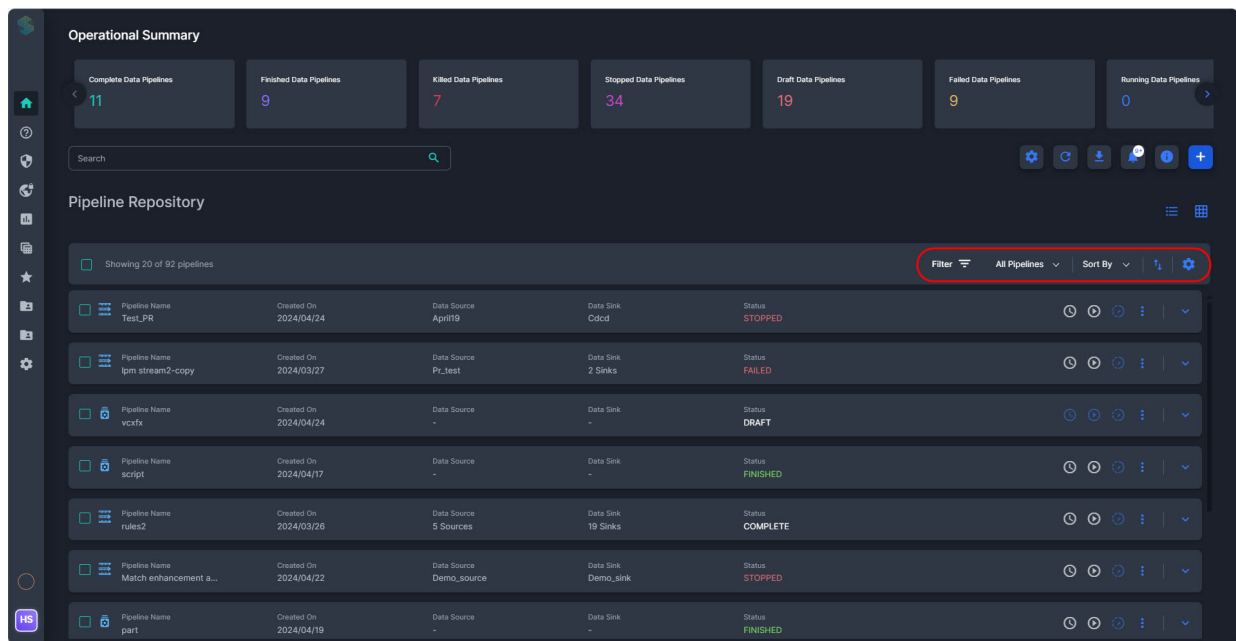
The workspace of BMS is the **Pipeline Repository**. It has the list of pipelines created in the system.

The following details of pipeline can be viewed by default:

- **Pipeline Name:** Name of the pipeline.
- **Created On:** Date on which the pipeline was created. You can configure the required date format through the [Common Settings](#) available in Admin settings of HyperSense.
- **Data Source:** Data Source used in the pipeline.
 - If a pipeline has one data source, you can view it in pipeline listing view.
 - If multiple data sources, then it displays the number of data sources associated with it. You can [expand](#) and view the details.
- **Data Sink:** Data Sink used in the pipeline.
 - If a pipeline has one data sink, you can view it in pipeline listing view.
 - If multiple data sinks, then it displays the number of data sinks associated with it. You can [expand](#) and view the details.
- **Status:** Status of the pipeline. Example: draft, complete, active, or running. In addition, you can view the following:
- **ID:** Unique ID of the pipeline.
- **Modified On:** Date on which the pipeline was edited or modified.
- **Tags:** Tags associated with the pipeline.
 - If a pipeline has one tag, you can view it in pipeline listing view.
 - If multiple tags, then it displays the number of tags associated with it. You can [expand](#) and view the details
- **Created By:** Name of the user who has created the pipeline.
- **Modified By:** Name of the user who has modified the pipeline.
- To learn more Home page features, refer to [Home page Functionalities](#).

Home page Functionalities

1. **Import/Export:** It allows you to import and export the selected, listed and configured rows in excel format. For more, refer [Import/Export](#).



The screenshot displays the 'Operational Summary' dashboard. At the top, there are seven summary cards: Complete Data Pipelines (11), Finished Data Pipelines (9), Killed Data Pipelines (7), Stopped Data Pipelines (34), Draft Data Pipelines (19), Failed Data Pipelines (9), and Running Data Pipelines (0). Below these is a 'Pipeline Repository' section with a search bar and a table of pipelines. The table has columns for Pipeline Name, Created On, Data Source, Data Sink, and Status. A red box highlights the 'Filter' button and the 'All Pipelines' dropdown menu. The table shows several pipelines with various statuses like STOPPED, FAILED, DRAFT, FINISHED, and COMPLETE.

2. **All Pipelines:** It allows you to display the pipelines as per All Pipelines, Offline and Online category.
3. **Sort By:** It allows you to sort the pipeline details based on the selection from the dropdown list. Few sorting options as follows None, ID, Pipeline Name, Status and Created On.
4. **Settings:** It allows you to select the column that you wish to see in asset panel. For configuration, refer to [Customization of Column](#).
5. **Filter:** Filter feature allows you to refine and filter the data of the asset by search. Assets are searched on the basis of criteria given in the fields. For more details, refer [Filter Pipelines](#).
6. **Status:** The status categories that characterize the pipeline status are listed below:

This allows you to view the status of the each created pipelines in workspace of the listing grid.

- **Completed:** The pipelines that are created and saved are marked as completed.
- **Draft:** The pipelines that are configured and saved as draft are marked as draft.
- **Submitted:** When a pipeline is run, it is marked submitted.
- **Stopped:** If you stop the scheduled pipeline, it is marked stopped.
- **Active:** The pipelines that are in active state as marked active.
- **Running:** The pipelines that start running on the scheduled time, they are viewed as running in pipeline repository.
- **Failed:** If there is any error in the streaming engine, the pipeline execution fails and status is changed to FAILED.
- **Killed:** If the running drivers for any pipelines are killed, then pipelines in pipeline repository are viewed as killed. If there are any upcoming schedules, they are further scheduled to run automatically.

Note:

You cannot edit the pipeline when it is in submitted or running state. To edit, you must stop the pipeline.

- Upon editing or expanding the particular pipeline from the workspace you can view, which measure is currently running during the pipeline execution. For edit pipeline information, refer [Edit Pipeline](#).

Example:

Note:

This feature is only applicable for Measure Transform Operators and Custom Operator.

7. To learn more, go back to the BMS home page [Pipeline Repository](#).

Customization of Column

Last updated on May 12, 2023

The pipeline information available in the Pipeline Repository are determined by the columns chosen for viewing. To configure the columns for a pipeline follow the below steps:

1. Go to **All Pipeline** Tab and click .
2. The **Select Columns** window is displayed and which has the following:
 - **Choose Columns:** Lists the columns that you can choose/move to Selected Columns.
 - **Selected Columns:** Lists the selected columns to be displayed on Pipeline.
 - **Make this as my default viewing criteria:** If enabled, the selected column details are viewed for the pipelines.
 - **Reset:** Click to reset the values to the previous state.
3. Drag and drop the required columns from the **Choose Columns** pane to the **Selected Columns** pane.
 - You can move multiple columns at a time. To select multiple columns, click **Ctrl** and **Select** required columns.
 - **On multiple column selection, you can see the grey highlighted color for the selected columns.**
 - The selected columns are viewed in the selected order. To change the order, drag and drop it to the respective position
 - The columns that are not required to be viewed can be moved back to **Choose Columns** pane.
 - The selected columns pane can have maximum 6 columns at a time.

4. Click **Save**.
5. To learn more, go back to the page [Pipeline Repository Actions](#).

Participating Record

Last updated on December 05, 2024

PR stands for Participating Records. It is used to uniquely identify the records in the input table that has contributed to the case generation. This feature is applicable for **Case Operator Sink**. The PR connection is configured with only **Streaming Transform Operators**.

Pre Configuration : This feature is continuation of the data copier. Before performing PR you need to register data copier. For more information, refer to [Data Copier](#).

Note:

- You can't work on PR without configuring the Data copier. Make sure Data Copier is registered.

To perform PR, follow below steps:

Note:

- If the checkbox of **Participating Record Enabled** is disabled or unchecked, Then the PR will not generate.

1. Go to the Canvas, go to **Operators grid** > **click** > **Configure the Data Reader**. To configure the Data readers for particular source types, refer to [DSV](#), [Kafka](#), [Database](#), [S3](#), [Azure](#) and [GCS](#).
2. In the **Configure Data Reader** window, upon configuring the **Connection Details** > **Preview**, go to **Dataset Tagging**.
3. In **Dataset Tagging**, tag the dataset with the data reader.
Complete the below details:
 - a. **Dataset Tagging**: It populates the data table. Select the required data table to tag with the source connection.
 - b. **XDR_ID**: It populates the column associated with the data table. Select primary key or column with unique values. It is mandatory field.

Note:

- Data table whose dataset tagging is done, make sure **Enable Local Cache** checkbox is selected. Select columns from the dropdown list for mapping. For more information, see [Data Table Setting](#).

- c. **Rated Column**: Select the column name from dropdown list to calculate fraud loss for the generated cases. Column with Data Type: Integer, Long, Decimal and Float are displayed as option in dropdown list.
- d. Click **Save**.

4. Upon saving the Data reader, connect the required streaming transform operators and configure. For example connect the Filter. To configure the Filter, refer to [Filter](#).

Note:

You can add multiple sources from **Data Reader** and connect it with **Mapper** component to combine the output through **Union Mode** check box. It combines the data of selected columns from the multiple sources and provides single combined source data. For more information, see [Mapper](#).

5. Connect the Filter with SummarizeX. To know more about configuration, refer to [SummarizeX](#).
6. After SummarizeX configuration, drag and drop the **Case Operator Sink**. In the **Case Operator Sink**, select **Participating Record Enabled** checkbox to generate the PR. Click Save. For more information, refer to [Case Operator Sink](#).
7. Save the Pipeline. In Canvas, Click Save available on top right corner of the screen.
8. Click **Start** icon on the particular pipeline. For more information of pipeline actions, refer to [Pipeline More Actions](#).
9. Once the pipeline **Run** is completed, the recorded output or data displayed in the Data Catalog.
10. To view output, hover the mouse on left pane and click on **Data Catalog**.
11. Go to **Data Catalog Home page** > Click of any table instance > you are navigated to the respective table instance in a new window.
12. In the **System_PR_Results** table, by default three additional columns are displayed that helps user to analyse the data. Those columns are mentioned as below:
 - a. **Case_id**: This column contains the case's id, which enables the user to access information about the case created for that specific PR.
 - b. **Pipeline_id**: This column contains the BMS pipeline id, which allows the user to obtain information about the pipeline that generated that particular PR.
 - c. **Case_operator_name**: In general, we can have n number of case operators in a BMS pipeline, and to understand which PR is generated, we are adding this column case_operator_name, which has the specific case_operator name from which that particular PR preset is generated.
13. You can view the PR details for any FM case in respective case investigation page.
14. To learn more about how to configure pipeline, refer to [Configure a Pipeline](#) or navigate back to the [Database](#) page.

Filter Pipelines

Last updated on May 12, 2023

Filter function has various operating features with data fields.

Below are the listed features:

1. Click **Filter** icon and select **Expand** icon, filter options displays .
2. Enter the criteria in the **Search Field**, It allows you to search for the specific field as **search option**.
Or
Select the required options manually from the list and clicking specific field checkbox.
3. Click **Add** icon. The selected fields added to on the window with respect to category.
4. The categories are:
 - **Created On:** Select the required search format from the **Choose Format** drop down list. The options are **Absolute**, **Quick** and **Relative**. Depends upon the choose format, next fields fields gets activated those are:

Absolute: In the **Absolute** option, select the required option from the **Start Date** and **End Date** drop down calendar.

Relative: In the **Relative** option, select the required Time option from the **Period From** and **Period To** drop down list.

Quick: In the **Quick** option, select the required option from the **Interval** and **Entry** drop down list.

- **Modified On:** Select the required search format from the **Choose Format** drop down list. The options are **Absolute**, **Quick** and **Relative**. Depends upon the choose format, next fields gets activated, those are:

Absolute: In the **Absolute** option, select the required option from the **Start Date** and **End Date** drop down calendar.

Relative: In the **Relative** option, select the required Time option from the **Period From** and **Period To** drop down list.

Quick: In the **Quick** option, select the required option from the **Interval** and **Entry** drop down list.

- **Fields:** Select the required fields to filter the data. The fields are:

Pipeline Names: Select the required pipeline from the **Pipeline Name** drop down list.

Data Source: Select the required source name from the **Data Source** drop down list.

Status: Select the required status of the pipeline from the **Status** drop down list.

Tags: Select the required tag from the pipeline from the **Tags** drop down list.

Created By: Select the User name from the **Created By** drop down list, which the pipeline is created by.

Modified By: Select the User name from the **Modified By** drop down list, which the pipeline is

modified by.


5. Click **Apply** to apply the filter.
6. Click **Reset**, to bring the filter configuration to initial stage.
7. Click **Clear All**, the application clears all the added filter options.
8. Click **Collapse** to collapse the filter panel.
9. Click and select the required specific field from the checkbox.
10. Click **Cancel** to close the filter screen.
11. To learn more, go back to the page [Pipeline Repository Actions](#).

Create New Pipeline

Last updated on May 12, 2023

Creating a pipeline in business modeling involves designing and establishing a systematic flow of activities or processes that align with the business model. It entails defining the sequential steps and stages through which products, services, or information moves.

To create a new pipeline:

1. Click  to **Create New Pipeline**.
2. **Create New Workflow** window is displayed.

Complete below details:

- a. **Pipeline Name:** Enter the pipeline name.
- b. **Tags:** Allows you to add the tags associated with the pipeline. It will be useful for further analysis of pipeline.
- c. **Description:** Allows you to enter the description to this pipeline.
- d. **Use case(Domain):** A field use case refers to a specific application or scenario in which a product, service, or technology is used in real-world environments or situations. Select the required Use Case scenario from the drop-down list.
- e. **Choose Processing Mode:** "Choose Processing Mode" refers to the act of selecting a specific mode or method for processing data or executing tasks within a system, application, or **software**. Select the required Processing Mode from the drop-down list. The process modes are as below:
 - i. **Online :** It is also called as **Stream line mode**. "Online Processing Mode" refers to a method of data processing that occurs in real-time or near real-time, providing immediate

responses or actions. In this mode, data is processed as it is received or generated, allowing for quick and interactive operations without significant delays.

ii. **Offline : It is also called as Batch Measures.** "Offline Processing Mode" refers to a method of data processing that occurs without real-time or immediate interaction. In this mode, data is processed without requiring an active connection to a network or relying on real-time inputs. It involves performing computations, analyses, or operations on stored or pre-collected data.

f. **Enable Global Exception:** Select the checkbox to allow global exception while running the pipeline. Applicable business rules are not applied to the pipeline. If checkbox is not selected, global exception is disabled and business rules are applied to the pipeline. For more information, see [Global Exception](#).

3. Click **Save**. A new canvas is displayed.
4. Upon saving work flow, a new Canvas page will be displayed.

Below is the example of canvas with pipeline:

5. You have a **two** choices, and the canvas **page** Operators pane depends on the processing mode you choose.
 - If you chose the **Online** processing mode, return to the Configure a Pipeline page and **refer to** [Online Pipeline\(Streaming\)](#) to learn how to configure the online pipeline.
 - If you chose the **Offline** processing mode, return to the Configure a Pipeline page and refer to [Offline Pipeline \(Batch Measures\)](#) to learn how to configure the **offline** pipeline.

Canvas Features

Last updated on June 13, 2025

Canvas is the page where you can create the pipeline for any of your required operations.

It has the following sections:

1. **Canvas - Pipeline Creation pane:** It is the creation pane, where you can drag and drop the operators; connect them, and create the pipeline.
2. **Top Toolbar:** Has the list of options that helps you in pipeline creation. For more information, see [Top Toolbar](#).
3. **Bottom Toolbar:** Provides you the Results preview on pipeline executions/ when a pipeline is run.

Pipeline Actions in Canvas

Top Toolbar

Top toolbar has the following options:

1. : Click to undo the changes.
2. : Click to redo the changes.
3. : Click to view the history of the pipeline actions. For more information, see [History](#).
4. : Click to add the notes associated to the pipeline, during creation. For more information, see [Annotation](#).
5. : Click to schedule the pipeline. For more information on scheduling, see [Scheduler information](#).
6. : Click to view the run history of the pipeline. For more information, see [Run History](#).
7. : Click to view the **Operators** grid, where you can see all the operators used for pipeline creation.
8. : Click to reset the changes to the previous save point.
9. : Click to zoom in the pipeline in the canvas.
10. : Click to zoom out the pipeline in the canvas.
11. : **Click to clear the pipeline in the canvas.**
12. : **Click to delete the selected component from the pipeline in the canvas.**
13. : **Click to clear the data, if you are running the same pipeline repeatedly.**

Note:

Clear checkpoint function will be activated only when the Enable/Disable checkpoint is True.

14. : Click to view and configure pipeline properties value. For more information on scheduling, see [Pipeline Settings](#).
15. : **When clicked, the Reporting Table Configuration** window is displayed. It helps you to select the data-storing strategy (Append / Delete).

Note:

- It is applicable for Batch Pipelines.
- It supports only partition tables.
- It supports all the measures (Data Match Measure, Query Measure, and SQL Measure).
- ***Click to know Usage Table Configuration***

When you want to perform a rerun on the Reporting Table, go to the Reporting tab of the selected measure (Query, Data Match, and SQL) and complete the below steps:

- Click and enable the **Enable Report Table Toggle** switch.
- Mention the **Reporting Table Name**.
- **Clear the Truncate Before Load checkbox.**

When you want to perform a rerun on the Run History Table, go to the Reporting tab of the selected measure (Query, Data Match, and SQL) and complete below steps:

- Click and enable the **Enable Report Table Toggle** switch.
- Mention the **Reporting Table Name**.
- Select the **Truncate Before Load** checkbox.
- Select the **Retention** checkbox in the **Run History** window.

Select the radio button mentioned below:

- Append:** If selected upon rerun, the reporting table or run history table is appended with new output records. For example: Consider Query Measure is run twice. Now if an append action is performed for the third run, then the third run data is added to the reporting table. The last two runs data is not deleted.
- Delete:** If selected upon rerun, the last run data in the reporting table or run history table is replaced with new output records. For example: Consider Query Measure is run twice. Now if delete action is performed for third run, then second run output data is deleted and third run data is added to reporting table.
- Click **Submit** to save the configuration.
- Logs are generated for each rerun and captured in Audit Trail.** For more information, see [Audit Trail](#).
- In case any error is generated during the rerun, it can be viewed in Error Logging. For more information, see [Error Logging](#).
- To know more about operator level rerun, refer to [Operator Level Rerun](#).

16. : Click to cancel the changes in pipeline.

17. : Click to save the changes in pipeline.

Canvas

Drag and drop/ Re-position the Operator

Select the operator, and drag and drop to the required position.

Note :

The operator is highlighted with the blue outline, when it is selected.

Operator Actions

Click vertical ellipses associated with the component. Here are few options associated with the nodes in pipeline.

- Total number of output records processed by the operator is displayed.
- **Edit:** Allows you to edit the operator configurations.
- **Detach:** Allows you to detach the input and output connections of the operator.
- **Enable:** Allows you to enable the operator configurations.
- **Disable:** Allows you to disable the operator configurations.
- **Look Back:** Allows you to configure the look back period at operator level. Each operator can have a different look back period. This feature is applicable to Query Measure and Data Match Measure. Below points to be considered for look back:
 - When pipeline is running, look back period set at operator level is given priority.
 - If the look back period is not set at operator level, look back period set in pipeline scheduler is considered. For more information, refer to [Look Back Configuration](#).

Note:

During pipeline re-run, lookback is prioritizes as per following order:

- Run History lookback has highest priority, followed by operator-level lookback.
- In absence of above lookback, pipeline level lookback is considered.

In case of task-rerun, task level lookback is considered. It does not refer lookback period set in BMS.

- **Create /View KPI:** Allows you to create and View the KPI configuration in the Pipeline.
- **Additional Settings:** Applicable only to Query Measure, SQL Measure, and Data Match operator. It allows you to configure the properties value of the component. For more information, see [Pipeline Settings](#).
- **Re-run:** Allows you to rerun the operator. Below points to be considered for rerun:
 - The pipeline should have run at least once, otherwise re-run will be in a disabled state.
 - When the parent operator is rerun, all the child operators are re-run automatically.
 - You can disable child operators if you do not want to rerun.
 - Consider below example for understanding:

Node 1 and Node 2 are the parent operators of Node 3 and Node 4. Node 3 is parent operator of Node 4. When you rerun Node 1, Node 3 and Node 4 will also rerun. Similarly when you re-

run Node 2, Node 3 and Node 4 will also rerun. If you disable Node 3 and rerun parent Node 1, Node 1 will rerun and Node 4 will not rerun. If you disable Node 4 and rerun Node 1, Node 1 and Node 3 will rerun.

- **Delete:** Allows you to delete the operator.

Scheduler

Scheduler helps in starting or stopping the pipelines. For more information, see [Scheduler](#).

Pipeline Settings

There are two categories of pipelines settings;

1. **Pipeline-level settings:** It is common pipeline settings for all the component on the canvas.
2. **Component-level settings:** It is the pipeline settings only for individual component.

Pipeline-level Setting

To configure pipeline settings.

1. Go to canvas page of any pipeline.
2. From the top toolbar, click on **Open Setting** icon .
3. Pipeline Setting window is displayed.
4. By default, there are 37 spark property values.
5. To edit a property, hover the cursor over the end of the row and click icon.
6. Modify the details.
7. To add more property, click **Add New** and specify the property details.
8. To delete any property and it's value, hover the cursor over the end of the row and click **delete** icon.
9. Click **Save**.

Note :

- The spark property **Enable/Disable Checkpoint** is which allows user to run the pipeline in two different ways. This Checkpoint is not applicable for **Database Source**.
- The purpose of Checkpoint is as below:

- If the checkpoint is **Disable** or **False**: When the user pause the running pipeline and run the same pipeline again after sometime, then the application runs the pipeline newly from the starting point.
- If the checkpoint is **Enable** or **True**: When the user pause the running pipeline and run the same pipeline again after sometime, then the application runs the pipeline right from where the run abandoned or paused.

Tip :

If the user Edit the **Directory name in the Data Readers configuration** or **replace the Data Reader** with new one, user must have to save the pipeline and run the pipeline.

Component-level Settings

Component-level settings is only supported for measures(batch).

1. **Go to canvas page of any pipeline.**
2. Click vertical ellipses associated with the component, and open **Additional Setting**.
3. Pipeline Setting window is displayed.
4. To edit a property, hover the cursor over the end of the row and click icon.
5. Modify the details.
6. To add more property, click **Add New** and specify the property details.
7. To delete any property and it's value, hover the cursor over the end of the row and click **delete** icon.
8. Click **Save**.

Schedule Pipeline

Last updated on September 30, 2023

Scheduler helps you in scheduling the pipelines. When scheduled, the pipelines are executed in the scheduled time and the data on data sources is processed dynamically based on the schedules. You can perform any of the following:

- **Start Pipeline**: Allows you to start the pipeline.
- **Save Schedule**: Allows you to save the schedule configurations.
- **Reset**: Allows you to reset the values configured to the previous saved state.
- **Edit Schedule**: Allows you to edit the schedule details.

Scheduler Configuration

To configure the Scheduler:

1. For the configured pipeline in the Pipeline Canvas, click .
2. Configure Scheduler window is displayed. Configure the scheduling details.
3. Click **Submit**.

OR

1. Go to the pipeline in the Pipeline Repository, Click .
2. Configure Scheduler window is displayed. Configure the scheduling details.
3. Click **Submit**.

Scheduler Modes

Scheduler has two ways of scheduling the pipeline:

- *Streaming Data*
- *Batch File*

Streaming Data

This mode is applicable for the pipelines with the following data sources - Kafka, S3, DB and DSV. If you try to schedule **Online** pipeline, only **Streaming Data** radio button is enabled in **Stream Scheduler** window. **Batch File** radio button is automatically disabled for all online pipeline.

- **Start On:** This section allows you to configure the start date and time for scheduling streaming data.
- **Run Now:** If enabled, the pipeline will run at the moment the pipeline is started irrespective of the timelines configured (in Scheduler) to run. If disabled, the pipeline will run on the timelines configured.

Batch File

This mode is applicable to schedule the pipelines with Query Measure and the pipelines with Database inputs. In this mode, - For pipelines with QM: after the pipelines are executed, immediately after the pipelines are processed, they are stopped, and the status is automatically updated to FINISHED state. - For pipelines with Database inputs, scheduling is continuous process that occurs in batches.

If you try to schedule **Offline** pipeline, only **Batch File** radio button is enabled in **Batch Scheduler** window. **Streaming Data** radio button is automatically disabled for all Offline pipeline.

1. **Frequency:** Indicates how frequently the pipeline should run. It can be Daily, Hourly, Minutely, Weekly, or Monthly. You can also repeat the run for the specified day group and time.

Example:

- a. If you have configured frequency as **minutely** and **Repeat Every** as 10 minutes, then pipeline is scheduled once in every 10 minutes.
 - b. If the frequency is **daily** and **Repeat Every** is 100 days, then the pipeline is scheduled once in every 100 days.
2. **Day Groups:** Allows you to create specific day groups. The pipeline will run on the configured **day group (based on dates, days and date range specified in the day groups)**. You can create the new day group or select the existing day groups. Multiple runtime for different Day Groups is allowed. For information on creating new day group, see [Create Day Group](#).

You can either use Default run time or Manual run time to schedule the pipeline run. Based on requirement you can select the radio button.

- a. **Default Runtime:** In this, you should specify the start and end time for the selected day group. Complete the below details:
 - i. **Add Row:** Click Add Row if you want to customize pipeline run for different day group and run time.
 - ii. **Day Group:** Select the Day Group for the pipeline to schedule.
 - iii. **Start Time:** Indicates the start time for the pipeline to be scheduled in the configured day group.
 - iv. **End Time:** Indicates the last time for the pipeline to schedule in the configured day group.
 - b. **Manual Runtime:** In this, you should specify the run time for the selected day group. Multiple run time for different Day Groups is allowed. You can configure different lookback period for each manual run. Complete the below details:
 - i. **Add Row:** Click Add Row if you want to customize pipeline run for different day group and run time.
 - ii. **Day Group:** Select the Day Group for the pipeline to schedule.
 - iii. **Run Time:** Indicates the run time for the pipeline to be scheduled in the configured day group.
3. **Look Back:** Look back period indicates the duration or time span to be looked back to process the data, from the scheduled time. You can either configure the absolute dates or relative time to.
4. **Exceptions:** Indicates the exceptional cases for pipeline scheduling. The pipeline execution will be ignored to execute in the configured time. You can select the configured day group or [create new day group](#). Complete the below details:
 - a. **Add Row:** Click Add Row if you want to customize pipeline run for different day group and run time.
 - b. **Day Group:** Select the Day Group for the pipeline not to schedule.
 - c. **Rule:** Indicates the rule for the pipeline not to schedule, with the configured day group.
 - d. **Start Time:** Indicates the start time for the pipeline not to schedule, with the configured day group.

- e. **End Time:** Indicates the last time for the pipeline not to schedule, with the configured day group.

Create Day Group

To create a day group:

1. To schedule a pipeline on custom day group, you should create Day Group:
 - a. In **Day Group** drop down, click **Create Day Group**. Create Day Group window is displayed.
 - b. Enter the custom day group name.
 - c. Select the day group type.
 - **Days:** Pipeline is executed only on selected days.
 - **Individual Dates:** Pipeline is executed only on the selected dates.
 - **Date Range:** Pipeline is executed within the configured From and To dates.
 - d. Click **Create Day Group**.
2. To schedule the pipeline for all days:
 - In **Day Group** drop down, select **All Days**. Select **Start Time** and **End Time** for the pipeline to be scheduled.

Use of Holiday List

Holiday list contains set of holidays that can be used to schedule the pipeline run. Before scheduling the pipeline run, you must create the holiday list. To know about how to create and configure holiday for pipeline run, refer to [Holiday List](#).

In Batch Scheduler, Holiday list is used in two way:

1. **Skip pipeline run on holiday:** When you configure holiday list in Exceptions, it will skip pipeline run on specific holidays. Follow below steps to skip pipeline run on holiday:
 - a. Create [Holiday List](#).
 - b. In Exceptions, configure Day Group with Holiday List.
 - c. Select **Start Time**, **End Time**, and **Rule**. Holiday list is added to Exceptions.
 - d. Click **Submit** to schedule the pipeline run.
2. **Schedule pipeline run on holiday:** When you configure holiday list in Day Group, it will run pipeline on specific holidays. Follow below steps to schedule pipeline run on holiday:
 - a. Create [Holiday List](#).
 - b. In Day Group, configure Day Group with Holiday List.
 - c. Select **Start Time** and **End Time**. Holiday list is added to Day Group.
 - d. Complete the configuration.

- e. Click **Submit** to schedule the pipeline run.

Start Pipeline

A pipeline can be started in the following ways:

- Go to the respective pipeline in the **Pipeline Repository** (either in list view or grid view), click **Start Pipeline** button.

Note :

- The pipeline can be scheduled when it is in **COMPLETE, FINISHED, FAILED, or KILLED** state.
- If you schedule a pipeline with **Run now** option enabled, then the pipeline will run irrespective of the timelines configured.
- If you schedule the pipeline with the specified time then the pipeline goes to **SUBMITTED** state only after you click **Start Pipeline** button. Pipeline goes to **RUNNING** state only after the pipeline reaches scheduled time.
- If the pipeline status is in **DRAFT** state, the **Start** and **Scheduler** buttons are disabled.
- For more information on different statuses, see *Pipeline Status*.

Stop Pipeline

The pipelines in **RUNNING** or **SUBMITTED** state can be stopped. To stop the pipeline:

- Go to the respective pipeline in the **Pipeline Repository** (either in list view or grid view), click **Stop Pipeline** button.

Edit Schedule

To edit the configured schedule:

1. Go to the pipeline in the Pipeline Repository (either in list view or grid view).
2. Click . In Configure Scheduler Window, perform the required updates.
3. Click **Submit**.

OR

1. Go to the Pipeline Canvas - Pipeline Edit Mode.
2. Click in the top tool bar. In Configure Scheduler Window, perform the required updates.
3. Click **Submit**.

Scheduler in Grid View

To schedule the pipelines in grid view, you must hover the mouse on the grid. The options associated with the scheduler are viewed. The configurations or edit functionalities are same as in list view.

The configurations or edit functionalities are same as in list view. To learn more, go back to the [Configure a Pipeline](#).

History

Last updated on May 12, 2023

History pane displays the history of the last performed actions with the following details:

- **Date and Time of the pipeline edit operation.**
- Number of components edited.
- User who has edited the pipeline.

To view the history details, you must click in the top toolbar of the canvas.

Following are the options associated with History window:

- : Click to view the History window in expanded view/ full view mode.
- : Click to search for any of the instances in the history of operations performed.
- : Click to close the History window.
- : Click to switch back from full view mode to the actual mode.

Annotation

Last updated on May 12, 2023

Annotation helps you to add the notes for the pipeline edit actions or observations. These notes might be helpful for any other user/himself for future analysis. To annotate or add note:

1. **Click on the top tool bar of the Canvas.**
2. The Annotation window is displayed.
3. **In the Add Notes field, enter the required note and click .**
4. The added note will be posted.
5. **You can also comment on the existing notes. The Comments section will display the added comments.**

Following are the options associated with Annotation window:

- : Click to view the Annotation window in expanded view/ full view mode.
- : Click to search for any of the instances in the Annotations of operations performed.
- : Click to close the Annotations window.
- : Click to switch back from full view mode to the actual mode.

Import/Export Pipelines

Last updated on May 12, 2023

It allows you to export pipelines from one database environment and Import in another database environment. If required, we can also do the export and import of pipelines in same client / environment as well. Import or Export of pipelines can be done only in the pipeline list screen. Import/Export Button will be enabled in Pipeline list screen and it is disabled in Grid View.

Note :

- User having Import and Export privilege can do the Import and Export of Pipelines.
- For performing import/export operation, the Clients should be of same version. Suppose if we are exporting from one client (ver 2.2.1) and trying to import in another client (ver 2.2.2), then it will give an error.
- Import/Export is not supported for pipelines which has dependency with *data catalog*.

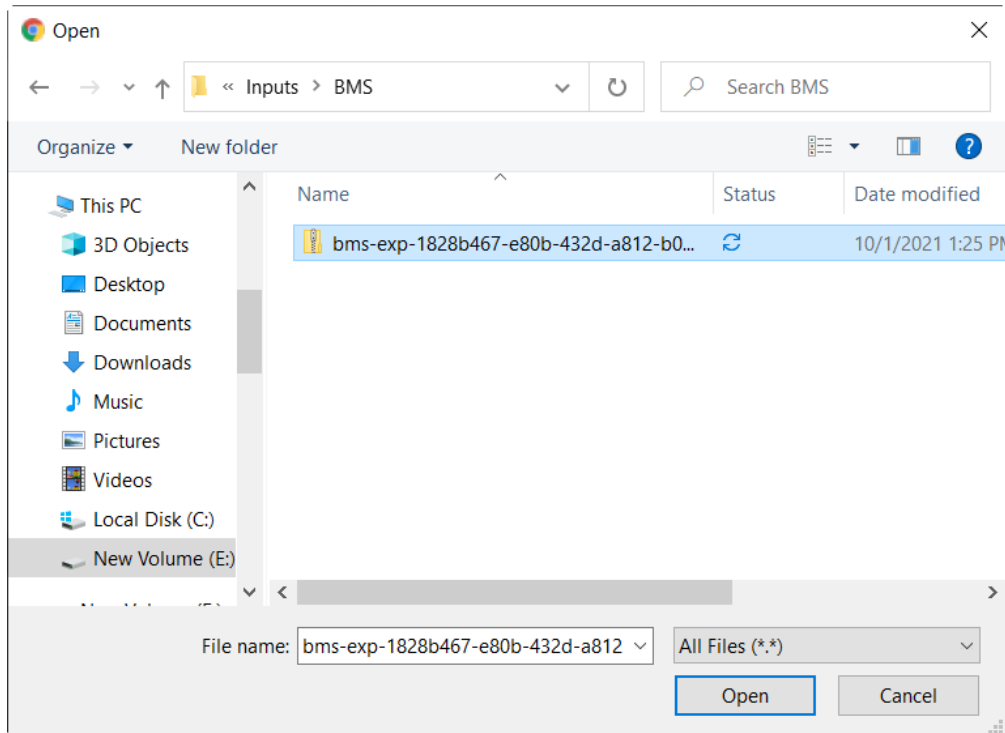
Export To export pipeline(s):

1. On the landing page of BMS, select the checkbox against the required pipeline.
2. Click on import/export button and then click **Export** option under it.
3. Export success dialog box is displayed.
4. Zip file containing JSON File is exported which is used to import pipeline in same client or in another system environment.

Import To import pipeline(s):

1. Go to the landing page (*Pipeline Repository*) of BMS.
2. Click Import/Export pipelines icon and select Import.




3. Browse and select for the exported zip file.



4. An Importing pipelines window is displayed.

Configure the following:

- **Search:** It allows you to query the pipeline name. Enter the expected pipeline name and select from the suggested choice(s).
- **Pipeline Name Checkbox:** Allows you to select the desired pipeline(s).
- **New Pipeline Name:** You can enter any desired name. If you don't specify any new name, then system will suffix the name with **Import_1**. For example, if you are importing a pipeline with a name as **Test**, and you don't provide any new name then the system will add the imported pipeline as **Test_Import_1**. If you import the same **Test** pipeline again, then it increases the suffix by 1. New pipeline name will be **Test_Import_2**, **Test_Import_3** and so on.
 - If the imported **Source** and **Sink** name already exist in the client, then system will suffix the name with **import_1** to the name of **Source** and **sink**.
 - Suppose the source name is **Test Source** and it is already available in the system then it will append **Test Source_Import_1** and again if you import then **Test Source_Import_2**.
 - Similarly, if the **Sink** name is **Test Sink** and it is already available in the system then it will append the name as **Test Sink_Import_1** and again if you import then **Test Sink_Import_2**.
- Click **Import**.
- At the top right corner of the pipeline repository screen, you will find the import/export loading indicator. It shows the import/export state when it is being loaded into the client.

-  Spinning circle indicates importing or exporting is in progress and,
-  Complete circle indicates the execution of Import or Export operation.
-  Complete red circle indicates that the import/export is failed.

5. Import status dialog will be displayed.

- If the selected zip file is empty, it returns an error message as Invalid or Empty ZIP File. If user makes any changes inside the JSON File available in the exported zip File, then it will returns an error message as Invalid or Empty ZIP File.
- If you are importing 3 pipelines and only 2 pipelines got imported and 1 got failed, then a dialog message will be displayed stating that Successful import 2 failed import 1.
- Successful dialog message will be shown as below:

6. After every import, a result file in csv format will be downloaded. It contains the import success or fail status along with other details.

Note : In case of importing to another client for SQM and QM, after import, you need to *configure* table to resolve unable to fetch tables error.

| | A | B | C | D | E | F | G |
|---|-------|-------------------|-------------------------|---------------------------|---------|-------------------|---|
| 1 | Sl.No | Old Pipeline Name | Preferred Pipeline Name | New Pipeline Name | Status | Error Description | Uploaded File Name |
| 2 | 1 | testgcpsink12345 | testgcpsink12345 | testgcpsink12345_import_1 | Success | | bms-exp-1828b467-e80b-432d-a812-b07d70357de3-20211001075531.zip |
| 3 | 2 | testjai8777 | testjai8777 | testjai8777_import_1 | Success | | bms-exp-1828b467-e80b-432d-a812-b07d70357de3-20211001075531.zip |
| 4 | | | | | | | |
| 5 | | | | | | | |
| 6 | | | | | | | |
| 7 | | | | | | | |

7. The pipeline will be added in the **pipeline repository**.

Run History

Last updated on May 15, 2025

Run History pane displays the run history of the last performed run operations on the pipeline.

Note :

- It is applicable only for Query Measure, SQL Measure, and Data Match Measure.
- **Truncate Before Load** option must be disabled in QM and SQLM *reporting configuration*, So that the run history can display the results in the results preview based on the selected date.

The Run History window includes the following details:

1. **Action Icons:** Run History window has below action icons:

- a. : Click to view the Run History window in expanded view/ full view mode.

- b. **Search Bar:** Click to search for any of the instances in the run history of operations performed.
- c. : Click to close the Run History window.
- d. : Click to switch back from full view mode to the actual mode.
- e. **Retention:** It allows you to store the run history detail for the operators. You should mention the days and select the operator whose run history to be stored.

Note:

Retention is available for Query Measure, Data Match Measure, and SQL Measure.

Below options are available on Retention Window:

- i. **Days:** Mention the days to store the run history.
- ii. **Search:** Allows you to search the specific operator.
- iii. **Retention Checkbox:** Select the check box whose run history to be stored.
- iv. **Submit:** Click Submit to save the retention configuration.

2. **Display Pane:** Display pane has the below list of columns:

- a. **SL.No. :** Displays the run instance sequence.
- b. **Start Date:** It displays the datetime of pipeline run operation.
- c. **Complete Date:** It displays the completion datetime of the run operation on the pipeline.
- d. **Pipeline From:** It displays the pipeline level look-back period. For more information on look-lack period, refer to [Look-back](#) period.
- e. **Pipeline To:** It displays the pipeline level look-back period. For more information on look-lack period, refer to [Look-back](#) period.
- f. **Status:** It displays the status of the run operation. The following status can be viewed:
 - i. **Finished:** If the run operation is successful.
 - ii. **Killed:** If the driver is terminated.
 - iii. **Failed:** If there is any database environmental issue.
 - iv. **Stopped:** If the run is terminated by user.

3. **Ellipsis Action:** In Run History window, you will find below ellipsis actions:

- a. **View Details:** It displays the run details of the particular run instance. Click to view the details.
- b. **Enable Lookback:** It allows you to enable the lookback for the run instance. It helps if any late record arrives. When you click, the below actions are performed:
 - i. **Lookback** tag is added to run instance.
 - ii. A snackbar message is displayed. The message is read as "*Lookback selected for Re-Run is associated with runId: <run_id>.*"

- c. **Disable LookBack:** This option is displayed when you set lookback for any run instance. When clicked, the lookback tag is removed from the run instance.

How lookback works for run history?

Consider the below example that shows the number of incoming records.

| Date | Incoming CDR Records |
|-------------|--------------------------|
| 13-Feb-2025 | 20 (Late CDRs of 11-Feb) |
| 12-Feb-2025 | 50 (Late CDRs of 10-Feb) |
| 11-Feb-2025 | 350 |
| 10-Feb-2025 | 500 |

Lookback work as per the data storing strategy. Data-Storing strategy either append or delete the records from the usage table. Below are the data-storing strategies:

1. **Append Strategy:** In this strategy, after every re-run, the usage table is updated with a new set of records.

- *Steps for data storing according to above example:*

1. Enable the lookback for the 10th Feb run instance and rerun the pipeline.
2. 500 records are processed and stored in the usage table.
3. Enable the lookback for the 11th Feb run instance and rerun the pipeline.
4. **350 records from 11th Feb are processed, and a total of 850 records are stored in the usage table.**
5. On 12th Feb., 50 late CDRs of 10th Feb are received. Enable the lookback and rerun the pipeline.

Note: On 10th Feb., 500 records are received. On 12th Feb., 50 late records of 10th Feb are received. So on 10th Feb., a total of 550 records are available.

6. The system will process 550 records instead of 50 records. There will be a total of 1400 records in the usage table.
7. On 13th Feb., 20 late CDRs of 11th Feb are received. Enable the lookback and rerun the pipeline.

Note: On 11th Feb., 350 records are received. On 13th Feb., 20 late records of 11th Feb are received. So on 11th Feb., a total of 370 records are available.

8. The system will process 370 records instead of 20 records. There will be a total of 1770 records in the usage table.
9. For every rerun, a new run instance is displayed in the run history window. As there are 4 reruns, there will be 4 run instances.

Table for the above procedure as follows:

| Date | Run Id | Incom- ing CDR Records | Pipeline Start Time | Pipeline End Time | Lookback Start Time | Lookback End Time | Records Processed as per lookback | Total number of records processed in usage table |
|------------|--------|-------------------------------------|------------------------|----------------------|------------------------|----------------------|--|--|
| 10- Feb | 203 | 500 | 10-05-2025 00:30 | 10-05-2025 23:40 | 10-05-2025 10:40 | 10-05-2025 13:40 | 500 | 500 |
| 11- Feb | 204 | 350 | 11-05-2025 00:30 | 11-05-2025 23:40 | 11-05-2025 10:40 | 11-05-2025 13:40 | 350 | 850 |
| 12- Feb | 205 | 50 (late CDRs of 10th Feb) | 12-05-2025 00:30 | 12-05-2025 23:40 | 10-05-2025 10:40 | 10-05-2025 13:40 | 550 | 1400 |
| 13- Feb | 206 | 20 (late CDRs of 11th Feb) | 13-05-2025 00:30 | 13-05-2025 23:40 | 11-05-2025 10:40 | 11-05-2025 13:40 | 370 | 1770 |

2. **Delete Strategy:** In this strategy, after every re-run, the system will delete the records as per the lookback period, and the usage table is updated with a new set of records.

- *Steps for data storing according to above example.*

1. Enable the lookback for the 10th Feb run instance and rerun the pipeline.
2. 500 records are processed and stored in the usage table.
3. Enable the lookback for the 11th Feb run instance and rerun the pipeline.
4. **350 records from 11th Feb are processed, and a total of 850 records are stored in the usage table.**
5. On 12th Feb., 50 late CDRs of 10th Feb are received. Enable the lookback and rerun the pipeline.

Note: On 10th Feb., 500 records are received. On 12th Feb., 50 late records of 10th Feb are received. So on 10th Feb., a total of 550 records are available.

6. The system will look for the records as per the set lookback. It will delete previous 500 records from usage table and process total 550 records of 10th Feb that includes 50 late CDRs.
7. On 13th Feb., 20 late CDRs of 11th Feb are received. Enable the lookback and rerun the pipeline.

Note: On 11th Feb., 350 records are received. On 13th Feb., 20 late records of 11th Feb are received. So on 11th Feb., a total of 370 records are available.

8. The system will look for the records as per the set lookback. It will delete previous 350 records from usage table and process total 370 records of 11th Feb that includes 20 late CDRs.
9. For every rerun, the system will delete the run instance as per the lookback and create a new run instance. In total, there will be 2 run instances.

Table for the above procedure as follows:

| Date | Run Id | Incom- ing CDR Records | Pipeline Start Time | Pipeline End Time | Lookback Start Time | Lookback End Time | Records Processed as per look- back | Total number of records processed in usage table |
|------------|--------|-------------------------------------|------------------------|----------------------|------------------------|----------------------|--|--|
| 10- Feb | 203 | 500 | 10-05-2025 00:30 | 10-05-2025 23:40 | 10-05-2025 10:40 | 10-05-2025 13:40 | 500 | 500 |
| 11- Feb | 204 | 350 | 11-05-2025 00:30 | 11-05-2025 23:40 | 11-05-2025 10:40 | 11-05-2025 13:40 | 350 | 850 |
| 12- Feb | 205 | 50 (late CDRs of 10th Feb) | 12-05-2025 00:30 | 12-05-2025 23:40 | 10-05-2025 10:40 | 10-05-2025 13:40 | 550 (processed) 500 (delet- ed) | 900 |
| 13- Feb | 206 | 20 (late CDRs of 11th Feb) | 13-05-2025 00:30 | 13-05-2025 23:40 | 11-05-2025 10:40 | 11-05-2025 13:40 | 370 (processed) 350 (delet- ed) | 920 |

Note:

Re-Run Condition for Disabled Measures

- For disabled measures upon rerun, the usage (reporting / run history) table remains unchanged.
- When the delete data storing strategy is selected, the run instance of the enabled measures is deleted from the older run instance. Follow the below example for better understanding:
 - Consider you have 3 measures: QM1, SQLM1, DMM1. All 3 measures are in an enabled state. Upon rerun, run id:211 is created with all 3 measure run details.
 - Now consider SQLM1 is disabled. Upon rerun, new run id:212 is created with 2 measure run details. It will have run details of measure QM1 and DMM1. From run id:211, SQLM1 run detail is available, whereas QM1 and DMM1 run details are deleted.

How to view run history details?

1. Click **Run History**, available in the top toolbar of the canvas. For pipelines with no component, the run history option will be in a disabled state.
2. By default, the latest run instance is selected, and the run details are displayed at the bottom right corner of the canvas. If you want to change the displayed run instance:
 - a. Single-click on the required run instance to view the run history.
 - b. A confirmation message is displayed and the selected run instance is displayed at the bottom right corner. **The run results for all the component in this pipeline will be directed to the selected datetime.**
 - c. **Total count of the number of records processed by the operator is displayed as per the selected Run Instance.**

Note:

Count for number of records processed is available for Query Measure, Data Match Measure, and SQL Measure.

View - Pipeline and Configuration

Last updated on January 19, 2024

It allows you to view the data for the respective pipeline. In the view mode, you can only view and read the pipelines, pipeline connections and pipeline configurations. You cannot edit and save any data information in the view mode.

To View pipeline(s):

1. On the workspace of **BMS**, go to ellipsis action of the required pipeline.
2. Click on **View** option.
3. The application displays respective pipeline connections in the canvas window.
4. To view the respective **Source(s)**, **Transform Operator(s)**, **Data Writer(s)** and **Data Reader(s)** configuration, follow below steps:
 - a. Click on the vertical ellipses.
 - b. Select **View** option. The application displays the configuration inside the pipeline.
5. Click **Cancel** to go back to the workspace.

Actions Icons on Canvas:

You can use below actions icons in the view mode:

1. History: It will display the actions performed on the pipeline,
2. Run History: It will displays the run history of the last performed run operations on the pipeline such as when pipeline has started, completed and status. Run History is applicable for batch pipelines.
3. Zoom In: Click to Zoom-In the canvas.
4. Zoom Out: Click to Zoom-out the canvas.

Note :

You can view the KPI pipeline configuration details by selecting the **View KPI**. This is applicable for the **Query Measure and SQL Measure only**.

To view KPI configuration, follow below steps:

- On the pipeline, click vertical ellipses.
- Select the **View KPI**. The application displays the data inside the pipeline.

Copy Pipelines

Last updated on May 12, 2023

It allows you to copy pipelines configurations. Copying an existing pipeline and using it as a starting point is one method for developing a pipeline. If the pipeline you want to duplicate is in the same environment, you may copy it; if it is in a different environment, you can export it from that instance and import it into yours.

Copying creates a duplicate of a pipeline and allows you to give it a unique name. Even if a pipeline has errors, you can clone it; however, the new pipeline will be in the DRAFT state until you manually change it.

To copy a pipeline(s):

1. On the workspace of BMS, go to ellipsis action of the required pipeline.
2. Click on **Copy** icon .
3. The application displays **Copy <pipeline name> window**.
4. Configure the following in **Basic Configuration** tab > **Pipelines Details**:
 - **Pipeline Name**: Enter a name for the new pipeline.
 - **Tag**: Enter a tag name to organize the pipeline.
 - **Description**: Enter a brief information for the pipeline.
 - **Use Case(Domain)**: Select the use case from the existing dropdown list.
5. Click **Copy**.
6. New pipeline is listed in the Pipeline Repository.

Rename Pipeline

[Last updated on May 12, 2023](#)

It allows you to update or change the Pipeline Name, Tags, Description, and Use Case(Domain) of the pipeline with existing data source of pipeline configuration.

Renaming an existing pipeline and using it with updated name and configuration is one method for developing a pipeline.

To rename pipeline(s):

1. On the workspace of BMS, go to ellipsis action of the required pipeline.
2. Click on Rename icon.
3. The application displays **Edit/Rename <pipeline name> window**.
4. Configure the following in **Basic Configuration** tab > **Pipelines Details**:
 - **Pipeline Name**: Enter the pipeline with new name.
 - **Tag**: Enter a tag name to organize the pipeline.

- **Description:** Enter a brief information for the pipeline.
- **Use Case(Domain):** Select the use case from the existing dropdown list.
Or,
Enter the new use case and click **ADD**. Upon adding, the application add's the new created use case in dropdown list and then select from dropdown list.

5. Click **Update**.

6. Updated pipeline name is listed in the Pipeline Repository with existing pipeline configuration.

Edit Pipeline and Functions

Last updated on May 12, 2023

It allows you to Edit the data for the respective pipeline and related Operators (Data Reader, Transform Operators, Data Writer). In the edit mode, you can edit the pipelines, pipeline connections and pipeline configurations.

To Edit pipeline(s):

1. On the workspace of **BMS**, go to ellipsis action of the required pipeline.
2. Click on **Edit** option.
3. The application displays respective pipeline connections in the canvas window.
4. To edit the respective **Source(s)**, **Transform Operator(s)**, **Data Writer(s)** and **Data Reader(s)** configuration, follow below steps:
 - a. Click on the vertical ellipses.
 - b. Select **Edit** option. The application displays the configuration inside the pipeline.
 - c. Update the required changes.
5. Click **Save**. The application saves and navigate back to the workspace.

Checkpoint rules Edit Operators checkpoint rules for Transform Operator(s), Data Writer(s) and Data Reader(s) are as below:

1. **Join Operator:** Upon Editing the Join Operator, the application shows two types of checkpoint features:
 - a. Upon Editing the timestamp details of Join Operator, the application displays the popup window as shown below.
 - i. Select **Delete Checkpoint** checkbox to run the pipeline from the starting point.

- ii. Deselect the **Delete Checkpoint** checkbox to continue the run pipeline from the paused or **previous point**. For more information, refer to the [Checkpoint](#).
 - iii. Click **Save**.
 - iv. Click **Close** to close the popup.
 - b. Upon Editing the Join details of Join Operator, the application displays the popup window as shown below.
 - i. Click **Save**. The application clear checkpoint to run the pipeline from the starting point.
 - ii. Click **Close** to close the popup.
2. **Formula Operator: Upon Editing the Formula Operator, the application shows below checkpoint message:**

Upon Editing the details of Formula Operator, the application displays the popup window as shown below.

 - a. Select the **Delete Checkpoint** checkbox to run the pipeline from the starting point.
 - b. Deselect the **Delete Checkpoint** checkbox to continue the run pipeline from the paused or **previous point**.
 - c. Click **Save**.
 - d. Click **Close** to close the popup.
3. **Summarizer Operator: Upon Editing the Summarizer Operator, the application shows two types of checkpoint features:**
 - a. Upon Editing the timestamp details of Summarizer Operator or editing/adding/deleting any aggregate column, the application displays the popup window as shown below.
 - i. Select the **Delete Checkpoint** checkbox to run the pipeline from the starting point.
 - ii. Deselect the **Delete Checkpoint** checkbox to continue the run pipeline from the paused or **previous point**.
 - iii. Click **Save**.
 - iv. Click **Close** to close the popup.
 - b. **Upon Editing the group by Columns of Summarizer operator, the application displays the popup window as shown below.**
 - i. Click **Save**. The application clear checkpoint to run the pipeline from the starting point.
 - ii. Click **Close** to close the popup.

Result Preview Features

Last updated on September 30, 2023

1. On the canvas page, single-click on the SQL Measure operator to select it.
2. At the bottom of the canvas, click Results Preview.
3. Preview window with table records details is displayed. In the below tab, the tab header represents the table name which was configured in [Reporting tab](#). If it was kept blank in the reporting tab, then a system generated table name will be displayed.
 - a. Two default columns are added in output/ reporting table. The columns are "measure_from_dttm" and "measure_to_dttm". Value for the column "measure_from_dttm" and "measure_to_dttm" are taken from Start Date and End Date configured in Scheduler Lookback.
4. Single-click on the serial number of the record, the row gets selected.
5. Now, right-click on the row to open menu. **Copy**, **Export**, and **Drilldown** actions can be performed.
 - a. **Copy**: Allows you to copy the records. To copy, select the row, right click > select Copy.
 - b. **Export**: A single or multiple selected rows in a table instance can be exported. To export, select the row, right click > select Export > select the required format.
 - c. **Drilldown**: It allows you to view the parent table. Drill down will filter the parent table data based on the row which we selected. Example: Suppose in result preview, there is one column with ID.

| Expected result preview | |
|---|---|
| ID | If you select row having value as 3 and if you drill down then it will filter the parent table with condition ID=3. In Expression Filter (ID =3 will be applied.) Accordingly it will display the data in results preview tab. If we are having multiple columns like ID and Name then condition will be like (ID=3 and Name='Jai') |
| | 2 |
| | 3 |
| | If the data on which we are doing the drill down is not available in the selected table then it will display no records found. |
| If we select the row data and after selecting the wrong table it will display one error message saying corresponding row data does not exist in the selected table. Please drill down to the other table. | |

To drilldown, follow below steps:

- Select the row, right click > go to **Drilldown** and click on the displayed parent table name.
- Preview window with parent table detail is displayed. Below points to be considered for drilldown operation.

- Schema will come in the drill down option along with table name only when we use schema in the Query of SQLM.
- In case if aggregations like Count, AVG, SUM etc., After doing drill down to the Table it will display the complete Table Data instead of filtering with particular value.
- SQLM Drill Down will not be supported in-case if we are having multiple select in same Query.
- In-case the table which is used in the query is not available in the data catalog or it got **deleted then that table will not be displayed in the Drilldown option.**

Example:

If we have used 2 tables(A,B) in sqlm query and 'A' table got deleted or not available in the Data catalog. Then only B table will be displayed in drill down option."

- Drill Down to the table completely depends on the tables available in the Data catalog. If the table is not available in Data Catalog then it wont display the table in drill down option.

6. You can perform common actions on the data tables. For more information, see [Common actions on Preview window](#).

Common Actions on Preview window

You can perform below common actions on the data tables:

1. : Allows you to *export* the selected rows or selected columns/complete table instance in any of these formats: **PDF, CSV, RTF**, and **TEXT**.
2. : **Allows you to share the selected table with the selected Users or User Groups. Share Data Table** Share feature has the list of users and user groups to share the required table context of any table instance.

You can select required users and user groups from the list, set the validity of the table access, and share it. The shared table details is received on e-mail by the respective users/ user groups and they can access it via the link received.

Note:

The link will be valid only for the duration that is configured in Share window.

- To **Share** any table:

- a. On the preview window, click . **Share** window is displayed.
- b. Select the **Link Expiry date**.
- c. Select the required **Users/ User Groups**.
- d. Click .
- e. After the table is shared, you are notified the successful message.

- f. After the table is shared, an email notification will be sent to you. For more information on configuring email notifications, see [Configure Email Notifications](#).
3. : Allows you to perform required settings for any table instance. On click, it will display the [Configure Grid](#) and [Color Rules](#) of respective table instance.
4. : Allows you to fetch the data from the data tables using predefined functions. For more information, see [Fx](#) section.
5. : Allows you to filter the records based on the filter condition configured. On click a **Choose Fields** pane is displayed. Search for the columns and configure the conditions associated with it. For more information, see [Filters](#).
6. : Allows you to filter the records based on the expression defined. For more information on expression filter, see [Expression Filter](#).
7. To learn next step of the SQL measure configuration, refer to [SQLM](#), [QM](#) and [DMM](#).

Configure Grid

Last updated on May 12, 2023

This feature allows you to perform the settings required for the Table to be visualized.

It has the following details:

Note:

All the rows in this configure grid are available as columns in table instance window. Hence, row refers to the row in configure grid and which is one column in a Table Instance.

- **Search:** Allows you to search any column from the configuration grid.
- **Column Name:** Displays the column name and which can be edited. After you edit, the updated column name is viewed further.
- **Column Type:** Indicates the data type of the column.
- **Visibility:** Enable this option if you want the selected column to be displayed in a table instance.
- **Alignment:** Select the alignment of column to be left, right or centered. By default, the first column is set as ascending.
- **Order:** Select the order of the data of the respective row. By default, it is ascending order for the first row and other row data will be displayed based on the first row. If required, you can set the order for all the rows as ascending or descending.
- **Null:** This property allows you to set Null or other values for any row. If a Null value is set for a column, the assigned value is displayed to that column in the Table Instance. Example: If it is set as 1 MB then 1MB is assigned to that row.
- **Clear All:** click this if you want to clear all the settings or selections in the configuration grid.
- : Click to undo the changes.

- : Click to redo the changes.
- **Apply:** After all the changes are made, click **Apply**. The updates will be reflected in Table Instance window.
- **Change column position:** Select any row in the configure grid, drag and drop to the position where you want that row to be displayed in Table Instance window (as columns). After you apply the changes, you can view the selected row in the selected position.
- Click **Color rules** tab to apply the color to columns. For more color information go to the below topic [Color Rules](#).

Color Rules

This tab allows you to define color rules for any table instance for the better visualization of data. When the color rule defined is matched with any of the operation, then you can view the data based on color rule. **Scenario:** For the records whose sequence ID is greater than 2, it should highlight the records in grey color. **Result: Records with sequence ID >2 will be highlighted in grey color. To create/add a color rule:**

1. On the preview window > click and click Color Rules tab. Click **Add New Rule** on Create New Rule pane.
2. A **Create Color Rule** section is displayed.
3. Configure the following:
 - a. **Rule Name:** Enter the name for the rule.
 - b. **Add Rule:** Enter the rule. *For suggestions, click Ctrl+Space.*
 - : Validates the rule or expression based on the syntax.
 - : **Allows you to save the expression.**
 - : Clears the configured expressions in the window.
 - : Allows you to indent the data.
 - : Allows you to import other expression filters, if required in the current expression.
 - c. Some of the Color Rules functional features:
 - : Allows you to edit the color rule.
 - : Allows you to delete the color rule.
 - : Allows you to set the color for the configured rule. You can either choose the color from existing list or you can add the colors to the list from the color picker. Color picker allows you to select the colors in **RGB/Hexadecimal** format. You can also pick the color from the color palette.

- : If enabled, the respective color rule is enabled.
- **Order of the Color Rules:** The order of the color rules available in the list can be defined. They have an effect on the data based on the order. Drag and drop any color rule to the desired position to reorder it.

Note:

The order gets updated automatically.

- d. **Delete Color Rule:** Through Delete icon you can remove the existing or added color from the list.
 - Click associated with the color rule. A confirmation dialog box is displayed.
 - Click **Confirm** to delete.

4. Click **Save**.

5. To learn how to configure fx, refer to the step [Fx_Function](#) in the Result Preview Features page.

Share Pipeline

Last updated on May 12, 2023

You can share any pipeline pipeline along with all the versions with user group(s) within an organization, After you share, the user group can able to access the pipeline as per the given access for the particular features.

Note :

- User needs to login HyperSense before viewing the pipeline.
- The user who wants to share the pipeline that user must have an access privilege of the **Share** option. The access is provided by the Administrator.
- Before sharing the pipeline to the required user group(s), the receiver ID must have to be added in the particular group list. To create new user group refer to [Creating New User Group](#).

To share pipeline follow the below procedure:

1. On the landing page of BMS, go and click ellipsis option of the required pipeline.
2. Click on **icon**.
3. The application displays **Share** window with **User Groups** tab.
4. Configure the below features with providing the access to the shared user groups(s) on the required features through selecting or de-selecting the respective feature checkbox.

- **View:** Allows to view the pipeline.
- **Edit:** Allows to Edit or change the configuration of the pipeline.
- **Delete:** Allows to remove the pipeline.
- **Rename:** Allows to re-name the pipeline.
- **Schedule:** Allows to schedule for the pipeline.
- **Start:** Allows to run the pipeline.
- **Copy:** Allows to copy or duplicate the pipeline.
- **Share:** Allows to share the pipeline.

5. Click **Share**.

6. Click **Cancel** to cancel the changes and close the window.

Fx Function

Last updated on May 12, 2023

Fx function allows you to fetch the data from the data tables using predefined functions. The query results of fx/pivot function are opened in new tabs with names as Fx_Results1, Fx_Results2, Fx_Results3, and so on. You can perform tool bar and other operation in fx result tab in similar way as in table grid screen. The following are the list of functions available in Fx function:

- **PIVOT:** Allows you to perform pivot operations(group by) on columns with functions.
- **MAX:** Returns the maximum value of selected column which has integer data type.
- **MIN:** Returns the minimum value of selected column which has integer data type.
- **COUNT:** Returns the count of selected column.
- **DISTINCT:** Returns the the distinct values present in the column.
- **AVG:** Returns the average of selected column which has integer data type .
- **SUM:** Returns the sum of selected column which has integer data type.
- **HOURLY:** Allows you to perform operations(group by) on timestamp for particular hour columns in pivot functions.
- **DAY:** Allows you to perform operations(group by) on timestamp and date data type columns for particular day in Pivot function.
- **YEAR:** Allows you to perform operations(group by) on timestamp and date data type columns for particular year in Pivot function.
- **MONTH:** Allows you to perform operations(group by) on timestamp and date data type columns for particular month in Pivot function.

Example: To fetch a distinct name column value from a table.

1. Click on fx icon, fx input field is open.
2. Type **Distinct** and select from the drop-down.
3. Type "name or string" as the column name and select from the drop-down.
4. On selecting column in fx query, selected column will get highlighted in table grid.
5. Press Enter key to query the column. The query result is opened in a new tab.
6. Navigate back to the Result Preview page for next step [Filter](#).

Filter in Preview Results

Last updated on May 12, 2023

Filters allows you to fetch the table instance details based on the filter values configured. To configure Filters:

1. Click in the top bar. A **Choose Fields** pane is viewed.
2. Click , **Choose Fields** is expanded where you can search for column names of the table and select them to configure filters.
3. Select the column and click **Add**. A configuration pane is displayed. Configure the filter conditions.
4. Choose Fields Pane has the following options:
 - **Remove Applied Filter**: Allows you to remove the applied filter when clicked on **Remove Applied Filter**.
 - **Reset**: Allows you to reset the values to previous state.
 - **Clear All**: Allows you to clear all the entries.
 - **Delete**: Select the filter and click delete button. It will be deleted from the **Choose Fields** pane.

- **Delete Multiple Fields:** To delete multiple fields, select those fields and click delete button. To select multiple fields, click Ctrl and select the fields.
5. **After you make required entries, click Apply.** The filter condition is applied and the **Table Instance** can be viewed with these filters.
 6. For expression filter, refer to [Expression Filter](#).

Expression Filter

This feature will help you to filter the records based on the expression defined. To create Expression filter:

1. **Click in the top bar of the Table Instance.** A new window to configure expression filter is displayed.
2. Configure the expression based on the following options available:
 - : Validates the expression based on the syntax.
 - : Clears the configured expressions in the window.
 - : Allows you to indent the data.
 - : Allows you to import other expression filters if required in the current expression. For more information, see [Import Expression Filters](#).
 - **Remove Applied Filter:** Allows you to remove the applied filter when clicked on **Remove Applied Filter** icon.
3. Click . An expression is saved. For more information on saving into folders, see [Save Expression Filters](#).

Note:

For any expression suggestions, click Ctrl+Enter.

Import Expression Filters

To import any expression filter:

1. Click . A **Select Filter pop up window is displayed, where you can choose the folders and in which you can select the required expression filters.**
2. Select the required filters and click **Import Filters.** The selected filters will be used with the current expression.

Save Expression Filters

You can save the expression filters in any folders. The folders could be existing or you can create a new folder. And hence the folders are logically grouped for easy understanding. To save any filter in the specific folder:

1. Click . A **Save Filter** window is displayed.
2. Configure the **filter name**, and select the **folder**. If you want to create a new folder, you can also click **Create New Folder** from the drop down.
3. Click **Save Filter**.
- 4.

Run Pipeline

Last updated on May 12, 2023

After any pipeline is created, it will be in **Drafted** status, you must Save the pipeline and run the pipeline to generate the results of any configurations performed.

Follow the below steps to run the pipeline:

1. After saving the pipeline, click **Schedule** icon to schedule the pipeline. For more information, refer to [Schedule Pipeline](#).
2. The Play button is enabled once the pipeline is scheduled.
3. Click to run. In case, if you want to pause a running pipeline, click .
4. : It allows you to Resume the configured pipeline from the stopped or failed point.

Note :

The resume function is applicable only for batch pipelines. It is not applicable for stream pipelines.

5. If the pipeline runs successfully, you will receive a successful **message**, otherwise, you will receive an error message, if the pipeline get fails.

Flooding Control

Last Updated on September 19, 2024

With the increasing number of fraud cases across the industries, it has become important to be vigilant to avoid the flooding of fraudulent activities. This feature allows you to be more vigilant against the fraud and notify you whenever such cases are observed to avoid flooding. This feature allows you to configure the rule and notify whenever cases generated from the rule engine breaches the

set threshold limit. It also provides you the authority to disable the flow or pipeline. You can further analyze and clear the abnormality.

Note:

- Flooding Control helps to avoid the flooding of cases due to wrong configuration created by user.
- Flooding Control is applicable to Case Operator Sink for both Streaming and Batch Pipelines.
- Multiple rules can be configured for the same pipeline with Case Operator Sink that was not used in the configuration earlier.
- If violation breach occurs for any Case Operator Sink, that particular Case Operator is disabled and colored grey. You can enable the disabled Operator. For more information, see *Operator Action*.
- In case of **Streaming Pipeline**: If there is single Case Operator Sink and violation breach occurs, operator is disabled and Pipeline Status is displayed as "**Killed**". When there are multiple Case Operator Sink and violation breach occurs, operator is disabled and Pipeline Status is displayed as "**Running**".
- In case of **Batch Pipeline**: If there is single Case Operator Sink and violation breach occurs, operator is disabled and Pipeline Status is displayed as "**Finished**". When there are multiple Case Operator Sink and violation breach occurs, operator is disabled and Pipeline Status is displayed as "**Finished**".

Follow below steps to navigate to the Flooding Control:

1. In Pipeline Repository, Click **Setting Icon** > **Select Flooding Control**.
2. You are navigated to **Flooding Control** window.
3. In this window, you can create *new*, *modify* and *delete* the configuration.

How to create new configuration?

Note:

- For **Streaming Pipelines** which is in **Running Status** and you try to create new alarm rule, confirmation message is displayed - "**Pipeline is already running. Do you want to restart pipeline?**". If you Click **Yes**, pipeline will restart with new alarm rule. If you Click **No**, alarm rule is applied to pipeline in next run.
- For **Batch Pipelines** which is in **Running Status** and you try to create new alarm rule, confirmation message is displayed - "**Pipeline is already running, alarm-configuration will take effect from next run**".

Follow below steps to create new configuration:

1. Click **Add New** icon.
2. **New Configuration: Flooding Control** window is displayed.
3. Complete the below details:

- a. **Pipeline:** It populates the pipeline name those have Case Operator Sink. Select the pipeline from the dropdown list. It is mandatory field.

Note:

Snackbar message is displayed when pipeline is selected.

- For Streaming Pipeline - "Pipeline is of type - Online".
- For Batch Pipeline - "Pipeline is of type - Offline".

- b. **Select Flow:** It populates the name of the Case Operator Sink. Select the Flow from the dropdown list. It is mandatory field.
- c. **Threshold Configuration:** Select the type of configuration you want to configure. It is mandatory field. Below are the options:
- Manual Configuration:** With this selection, user is able to disable the Case Operator Sink.
 - Automatic Configuration:** It allows system to monitor the violation count. If selected, below fields are enabled:
 - **Duration in Days:** Set the number of days you want alarm rule engine to monitor the violation count.
 - **Duration in Hours:** Set the number of hours you want alarm rule engine to monitor the violation count.
 - **Violation Count:** Set the violation count to trigger the alarm and notify the user.
 - Let us consider an example: Duration in Days is 1, Duration in Hours is 1, and Violation Count is 250. This means once pipeline has started, the system will monitor the 250 violations for the next 25 hours (Duration in day is 1 that is 24 hours, and Duration in hour is 1). If violation limit is breached within 25 hours, action will be performed based on selection. If violation happens after 25 hours, no action is performed.
- d. **Actions:** This field contains the list of action that should be performed when violation occurs. It is mandatory field. Below are the available options:
- Disable Flow:** It allows you to disable the configured Case Operator Sink.
 - Disable Pipeline:** It allows you to disable the pipeline. This action is not available for Manual Configuration.

Note:

Both actions are not allowed together.

- e. **Send Notification:** It is enabled once action is selected. Select the checkbox, if you want to notify user regarding the violation. If selected below options are displayed:
- To Email(s):** Select the mail recipients of email.
 - Cc Email(s):** Select the person to whom copy of email should be sent.

- iii. **Email Template:** It displays the list of email template for sending the email. Select the template from the dropdown list. To know more about how to create email template, see [Email Repository](#).
- f. **Clear All:** Click to clear the configured fields.
- g. **Create:** Click to create the configuration.

Configuration Example:

4. Once alarm rule is created, snackbar message is displayed "**Configuration Saved**".

How to edit configuration?

Note:

- For **Streaming Pipelines** which is in **Running** Status and you try to edit alarm rule, confirmation message is displayed - "**Pipeline is already running. Do you want to restart pipeline?**". If you Click **Yes**, pipeline will restart with new alarm rule. If you Click **No**, alarm rule is applied to pipeline in next run.
- For **Batch Pipelines** which is in **Running** Status and you try to edit alarm rule, confirmation message is displayed - "**Pipeline is already running, alarm-configuration will take effect from next run**".

Follow below steps to edit configuration:

1. Click **Edit** icon of configuration you want to update.
2. **Edit Configuration: Flooding Control** window is displayed.
3. Update the *field* details.
4. Click Update to update the latest configuration.
5. Upon update, snackbar message is displayed "**Configuration Updated Successfully**".

How to delete configuration?

Follow below steps to delete configuration:

1. Click **Delete** icon of configuration you want to delete.
2. Upon deletion, snackbar message is displayed "**Configuration Deleted**".

Global Exception

[Last Updated on November 29, 2024](#)

Global Exception is a feature that overrides the configured business rules. Exception list is global for all the rules. When rule is executed, entity is checked in the global exception list and the rule is not applicable on the global exception entity.

Note:

- Make sure **Enable Global Exceptions** checkbox is selected while creating new workflow. For **more information**, see [Create New Workflow](#).
- Global Exception is provided as functionality access to the important user(s) or group(s) to override the business rules. For more information, see [Functionality Access](#).

Follow below steps to navigate to the Global Exception:

1. In Pipeline Repository, Click **Setting** Icon > Select **Global Exception**.
2. You navigate to **Global Exception** window.
3. In this window, you can perform below actions:
 - a. **Create New Exception:** You can create new exception. Click Create New Exception. For more information, see [Create New Exception](#).
 - b. **Edit Exception:** You can modify the existing exception. Click Edit icon available in the exception. For more information, see [Edit Exception](#).
 - c. **Delete Exception:** You can delete the existing exception. Click Delete icon available in the exception. For more information, see [Delete Exception](#).
 - d. **Search:** Use search bar to search specific exception from the list.
 - e. Click on expand icon to view the exception details.

How to create new exception?

Follow below steps to create new exception:

1. Click **Create New Exception** icon.
2. **Global Exception** window is displayed.
3. Complete the below details:
 - a. **Select Table:** It is source table whose column to be mapped with global exception table to override business rules. You can either search or scroll data table and select from the dropdown list. Data table are fetched from data catalog.

Note:

- **Select table** is source table whose dataset tagging should be done while configuring the Data Reader. For more information, see [Data Reader](#).
- b. **Column:** It populates the column name from the selected table. Select the required column from the dropdown list.
 - c. **Global Exception Table:** It populates the global exception data table. Select the data table from the dropdown list.

Note:

- For the Global Exception Table perform Local Cache. Otherwise data table will not populate in the dropdown list. Select checkbox **Enable Local Cache** in Data table Settings. For more information, see Data Table *Setting*.
- Columns with string data type are populated in the dropdown list.

- d. **Entity:** It populates the column name from the global exception data table. Select the column from the dropdown list to be mapped with selected data table column.

Note:

- After **Enable Local Cache** checkbox is selected, select columns from **Select Match Columns** dropdown list for mapping. For more information, see Data Table *Setting*.
- **Enable Local Cache** is applicable to String Datatype.

- e. **Add New:** Click if you want to add more entity for exception. If you try to add new row without completing mapping details, snackbar message is displayed "**Configure all global exception mapping details**".
- f. **Delete Icon:** Click delete icon to delete the row. If you want to delete multiple rows, select the checkbox and click delete icon.
- g. **Save:** Click **Save** button to keep the global exception configuration.

Configured example:

4. Once Global Exception is created successfully, snackbar message is displayed "**Global exception created successfully**".
5. If you try to create exception using source table for which global exception exist, snackbar message is displayed "Global exception for source table already exists".

How to edit exception?

Follow below steps to edit exception:

1. Click **Edit** icon of exception you want to update.
2. **Edit Exception** window is displayed.
3. Update the *field* details. Click **Save** to update the exception details.
4. Once Global Exception is updated successfully, snackbar message is displayed "**Global exception updated successfully**".

How to delete exception?

Follow below steps to delete configuration:

1. Click **Delete** icon of exception you want to delete.
2. Confirmation message is displayed. Click **Yes** to delete.
3. Upon deletion, snackbar message is displayed "**Successfully deleted the configuration**".

Functionality Access

Follow below steps to provide Object Based Access:

1. From Common Left Panel, select **Admin > User Management**.
2. For selected User(s) / Group(s), Click Vertical Ellipsis > **Functionality Access**.
3. In **Business Management Studio** tab, select checkbox: "**Configure Global Exception**" , "**Enable Global Exception**".
4. Click **Save** to provide functionality access.

Version History

Last updated on April 8, 2025

Version History refers to the revision history which helps you to track and manage different versions of the same pipeline.

Note:

- In **Pipeline Repository** window, there are two versions: **Current** and **Latest**. **Current** version refers to **Active version which is displayed in pipeline repository**, and **Latest** version refers to version generated from the last created version.
- In **Pipeline Repository** window, **Current Version** and **Latest Version** may or may not have same number.
- **For a particular pipeline**, **Active** version is displayed in Pipeline Repository. You can run only active version of the pipeline.
- Initially, **Latest** Version is in drafted state and version number is 1.0. Edit and save the pipeline.
- New generated version is incremented by 1 from the latest version.
- **For example:** If there are 3 versions for a pipeline - 1.0, 2.0, and 3.0. Out of 3 versions, **Active version is 2.0. Then Current Version is 2.0 and Latest Version is 3.0. If you create new version**, then **Latest Version** will be 4.0.

View Version History

To view version history of pipeline follow below steps:

1. In **Pipeline Repository** window, for particular pipeline, Click vertical ellipsis > **Version History**.
2. **Version History** window for the particular pipeline is displayed. Title of this window is named as "<pipeline name>-Version History".
3. You will find below details in this screen:

- a. **Sr. No:** It displays the serial number to count how many version exist for a particular pipeline.
- b. **Version:** It displays the number of versions available for a particular pipeline. For example: 1.0, 2.0, 3.0 and so on.
- c. **Status:** It displays the pipeline status such as Failed, Drafted, Deployed, and Running.
- d. **Modified On:** It displays the date on which pipeline was modified.
- e. **User:** It displays the user name who has created the pipeline version.
- f. **Active:** It displays which version of the pipeline is available in the Pipeline Repository.

Common Actions

In Version History window, you will find common actions that you can perform. Click on the vertical ellipsis and you will find the below common actions:

1. **Create Version:** It allows you to create new version. You can create new version from the latest version. For more information, see [Create Version](#).
2. **Activate:** It allows you to activate the version which is not active. Active version is displayed in Pipeline Repository.

Note:

- Version History status changes when you activate any version.
- You can activate any lower version, unless there is no schema change. Schema change could be addition or deletion of column, change in column data type.
- If there is schema mismatch, snackbar message is displayed as "Only Versions with same schema can be activated".
- For example: There are 2 versions: 1.0, and 2.0. Current and Latest Version is 2.0. In Version 2.0 you change the schema and try to activate Version 1.0, you will get snackbar message.

Steps to activate version are listed as below:

- a. For particular version, Click vertical ellipsis > Click **Activate**.
 - b. **Particular Version status is marked Active.**
3. **View:** It allows you view the pipeline configuration of particular version.
 4. **Edit:** It allows you to edit the latest version. You can edit and update the pipeline component configurations. Edit option is disabled for the lower versions.
 5. **Delete:** It allows you to delete any particular version.

Note:

- You cannot delete Active version. Delete option is disabled for active version.

- If you delete the latest version which is not active and create new version. Latest version is created with deleted version number.
- For example: Pipeline has 3 versions: 1.0, 2.0, and 3.0. Active version is 1.0 and Latest version is 3.0. You cannot delete 1.0. Suppose you delete 2.0 and create new version. New generated version is incremented by 1 from the latest version, it will be 4.0 which is latest version. Also if you delete latest version 4.0 and create new version, then new version will be 4.0 not 5.0.

6. **Copy:** It allows you to copy any particular version. New pipeline is created with version 1.0. New Pipeline is displayed in Pipeline Repository window.

7. **Export:** It allows you to export any particular version.

Create Version

You can create new version from the Version History window. Follow below steps to create new version:

1. For particular pipeline, Click Vertical Ellipsis > **Version History**.
2. Click Vertical Ellipsis of Latest Version > Click **Create Version**.
3. Pipeline Canvas is displayed in **edit mode**. If you want to make changes in pipeline component you can edit them and Click **Save and View**. Else without modifying any pipeline component, Click **Save and View**.
4. In **Version History** window, new Version is displayed which is the latest version.
5. If there is any schema change in the latest version, version is activated automatically.

Data Reader

Data Reader Overview

Last updated on May 12, 2023

This allows you to create the new data source with respect to the different data readers. To configure the data source select the required data source from the Data Reader drop-down list, drag and drop the data reader and configure the source.

Data readers are as below:

- DSV allows you load the input files **CSV** or **text files with delimiter configurations**. For more information, refer to [DSV](#).
- **The Data Reader - Database** allows you to read the data in tabular format from the connected database. For more information, refer to [Database](#).
- Kafka Source is a data source file system input, which allows you to fetch or select the data from the configured Kafka source. For more information, refer to [Kafka Source](#).

- **S3 Source** is an object storage suite capable of handling structured and unstructured data including log files, artifacts, backups, container images, photos and videos. For more information, refer to [S3 Source](#).
- **GCS Source** that provides object storage through a web service interface. For more information, refer to [GCS Source](#).
- **Azure Source** is used to read data in CSV or Json format from the mentioned azure storage. For more information, refer to [Azure Source](#).

DSV Source

Last updated on December 05, 2024

DSV allows you load the input files **CSV** or **text files with delimiter** configurations. It allows the input patterns or expressions with following delimiters: [,:;][][\t]. The DSV file is loaded from the local file system.

To configure the DSV Source, you can either use the existing DSV configurations, or create a new Data Source connection.

Create Data Reader - DSV

Follow steps to create a new Data Reader - DSV:

1. In the Canvas, go to **Operators grid > click** . The **Configure Data Reader** window is displayed.
2. Select **DSV** and click **Next**. **Connection Details** tab is displayed. You can either **Create New Connection** or use the existing connections.
3. **To create new connection, click** . The configuration panes are enabled.
4. **Click the expand icon to configure General Details.**

Configure the following:

- a. **Data Reader Name:** Enter the name for DSV Data Reader.
- b. **Description:** Allows you to enter the description for DSV data reader.
- c. **Select Category:** Select the category to which the DSV Source belong to. You can also create a new category to add the sources. For more information on creating category, see [Create New Category](#).
- d. **Tag:** Enter the tags for the DSV source, that helps in identifying the purpose of data reader in pipeline.
- e. **Directory:** Enter the directory in the DSV storage, to store the output data.
- f. **Delimiter:** Enter the delimiter to separate the values that are being stored in DSV source.
- g. **Is Header Present:** If enabled, the file without header is also stored.
- h. **Click Next.**

5. Click the expand icon to configure **Configuration**.

Configure the following:

- a. **Compression Type:** Select the compression type. It accepts the input file in that selected format.
 - b. **Backup Strategy:** Select the backup strategy for the loaded file if it must be deleted, or archived after the file is processed.
 - i. **OFF:** Does not consider any backup strategy.
 - ii. **Archive:** All the files except the recent processed file from source directory are archived to the configured archive directory path after the file is processed.
 - iii. **Delete:** All the files except the recent processed file from source directory are deleted after the file is processed.
 - c. **File Accept Pattern:** Decides the file accept pattern. Example: *.txt- Accepts only files which has .txt extension.
csv and txt files are supported.
 - d. **Max File Age(in seconds):** Indicates the files to fetch within the configured time when a pipeline with this Data Source is run. **Example:** If we have configured 60 seconds; When the pipeline with this data source is run, it will the files will be fetched within 60 seconds based on the **Max File Per Trigger** parameter value.
 - e. **Max Files Per Trigger:** Enter the number of files that must be accepted. Example: If there are 5 files in the source directory, and value configured for **Max Files Per Trigger** is 2, then 2 files are processed at a time. And the 5 files are processed in 3 attempts.
 - f. Click **Save**.
6. Click **Next**. A **Preview** tab is displayed. Upload the file and preview.
 7. Click **Next**. The application navigates to the **Dataset Tagging**.

Complete the below details:

- a. **Dataset Tagging:** It populates the data table. Select the required data table to tag with the source connection.

Note:

- Select the **None** option from the **Tag Dataset** dropdown field, to create basic data source configuration.
- Select table options from the **Tag Dataset** dropdown field. This process is called as **Participation Record**. For more information, refer to *Participation Record*.

- b. **XDR_ID:** It populates the column associated with the data table. Select primary key or column with unique values. It is mandatory field.

Note:

- This field is disabled if **Dataset Tagging** is selected as **None**.

- **Data table whose dataset tagging is done, make sure Enable Local Cache checkbox is selected. Select columns from the dropdown list for mapping. For more information, see [Data Table Setting](#).**

c. **Rated Column:** Select the column name from dropdown list to calculate fraud loss for the generated cases. Column with Data Type: Integer, Long, Decimal and Float are displayed as option in dropdown list.

d. Click **Save**.

8. New Data source connection is created.

Use Existing Data Source

1. **Example of saved existing DSV Data Source** - In the Canvas, go to **Operators** grid > DSV Source category that it belongs to > Select required **DSV Source**.
2. Drag and drop the **DSV Source** to **Canvas > Configuration** pane.
3. Connect the DSV Source to the required Transformation Operators.
4. To learn how to configure Transform Operator, refer to the step [Transform Operator](#) in the **Configure a Pipeline** page.

Create New Category

To create a category in the DSV Source configuration grid:

1. Click on **Add/Search Category** configuration element. The configuration element is expanded.
2. Enter the category name in **Add/Search** category.
3. Click . A category is added.
4. Go back to step [Tag](#), to configure the next step of the DSV configuration.

Database Source

[Last updated on December 05, 2024](#)

The Data Reader - Database allows you to read the data in tabular format from the connected database.

You can connect to the database and run a query to fetch the required data, which can be provided as input to the Transformation Operators.

To configure the Database - Data Reader:

1. In the **Canvas**, go to **Operators** grid > click . The **Configure Data Reader** window is displayed.
2. In **Source Type** tab > select **Database**, and click **Next**. The **Connection Details** tab is displayed. You can either **Create New Connection** or use the existing connections.
3. To create new connection, click . The configuration panes are enabled.

4. Click **General Details** to configure.

Configure the following:

- a. **Connection Name:** Enter the connection name.
- b. **Description:** Allows you to provide the description for the connection.
- c. **Add/Search category:** Select the category to which you this connection to belong to. You can also create a new category to add the sources. For more information on creating category, see [Create New Category](#).
- d. **Tag:** Enter the tag for database connection, that helps in identifying the purpose of data reader (in future analysis).

5. Click **Next to configure Configuration**.

Complete below details:

- a. **User Name:** It is unique name used to distinguish an individual user from others. Enter the user name to connect to the host machine where the database resides.
- b. **Password:** It is a secret combination of characters that is used to authenticate and verify the identity of user. Enter the password to connect to the host machine where the database resides.
- c. **Database Type:** It refers to database category used to organize, store and manage data within DBMS. Select the database type from which the data is to be fetched.
- d. **IP/Host:** It is unique address assigned to each device connected to computer network that can receive or send data. Enter the IP or host details of the machine where database resides.
- e. **Port:** It is logical construct that allows multiple services to use same network interface on device. Enter the host server port number.
- f. **Database Name:** It refers to name assigned to specific database used for managing and storing data. Enter the database name for the selected database type.
- g. Click **Test Connection**. If the connection is successful, **Connection Successful** message is displayed. Click **Save**.

6. Click **Next**. A **Preview** tab is displayed.

Complete the below details:

- a. In **SQL Select Query** configuration grid, enter the SQL query to fetch the table details. **Example:** Select * from pipeline - indicates to fetch all the columns from the **pipeline** table from the connected database.
- b. Click to submit the query. After the query is run successfully, the data fetched from table can be visualized.

7. Click **Next**. The application navigates to the **Dataset Tagging**.

Complete the below details:

- a. **Dataset Tagging:** It populates the data table. Select the required data table to tag with the source connection.

Note:

- Select the **None** option from the **Tag Dataset** dropdown field, to create basic data source configuration.
- Select table options from the **Tag Dataset** dropdown field. This process is called as **Participation Record**. For more information, refer to *Participation Record*.

- b. **XDR_ID:** It populates the column associated with the data table. Select primary key or column with unique values. It is mandatory field.

Note:

- This field is disabled if Dataset Tagging is selected as **None**.
- **Data table whose dataset tagging is done, make sure Enable Local Cache checkbox is selected. Select columns from the dropdown list for mapping. For more information, see Data Table *Setting*.**

- c. **Rated Column:** Select the column name from dropdown list to calculate fraud loss for the generated cases. Column with Data Type: Integer, Long, Decimal and Float are displayed as option in dropdown list.

- d. Click **Save**.

8. After the data source is created, you can view the same in **Operators** grid > **Data Reader**.

9. To learn how to configure Transform Operator, refer to the step *Transform Operator* in the Configure a Pipeline page.

Create New Category

To create a category in the Database Source configuration grid:

1. Click on **Add/Search Category** configuration element. The configuration element is expanded.
2. Enter the category name in **Add/Search** category.
3. Click . A category is added.
4. Go back to step *Tag*, to configure the next step of the **Database** configuration.

Kafka Source

Last updated on December 05, 2024

Kafka Source is a data source file system input, which allows you to fetch or select the data from the configured Kafka source.

To create a new Data Reader - Kafka:

1. In the Canvas, go to **Operators grid** > click . The **Configure Data Reader** window is displayed.
2. Select **Kafka** and click **Next**. **Connection Details** tab is displayed. You can either create new connection or use the existing connection.
3. To create new connection, click . The configuration panes are enabled.
4. Click the expand icon to configure General Details.

Configure the following:

- a. **Data Reader Name:** Enter the name for Kafka Data Reader.
 - b. **Select Type:** Select the type of file to be processed. Following file types are supported: JSON, XML.
 - c. **Description:** Allows you to enter the description for Kafka data reader.
 - d. **Add/Search Category:** Select the category to which the Kafka Source belong to. You can also create a new category to add the sources. For more information on creating category, see [Create New Category](#).
 - e. **Tag:** Enter the tags for the Kafka source, that helps in identifying the purpose of data reader in pipeline.
5. Click **Next** or click the expand icon to configure Configuration.

Configure the following:

- a. **Enter URL:** Enter the Broker URL(with machine IP and port) for Kafka consumer connection.
 - b. **Topic:** Enter the Kafka topic name.
 - c. **Error Record Action:** Select the action to perform when there is an error in fetching the data from the source.
 - i. **Discard:** Discards the error condition and continues with the actual flow.
 - ii. **Send To Error:** Notifies with the error message when error in fetching the data.
 - iii. **Stop Pipeline:** Stops the pipeline (running)when there is error in fetching the data.
 - d. **Library Version:** This field pre selected by default.
 - e. **Additional Configurations:** Allows you to configure the additional properties in the key - value pairs. For more information, see [Kafka - Additional Configurations](#).
 - f. Click **Save**.
6. Click **Next**. A **Preview** tab is displayed. Upload the file and preview.
 7. Click **Next**. The application navigates to the **Dataset Tagging**.

Complete the below details:

- a. **Dataset Tagging:** It populates the data table. Select the required data table to tag with the source connection.

Note:

- Select the **None** option from the **Tag Dataset** dropdown field, to create basic data source configuration.
- Select table options from the **Tag Dataset** dropdown field. This process is called as **Participation Record**. For more information, refer to *Participation Record*.

- b. **XDR_ID**: It populates the column associated with the data table. Select primary key or column with unique values. It is mandatory field.

Note:

- This field is disabled if Dataset Tagging is selected as **None**.
- **Data table whose dataset tagging is done, make sure Enable Local Cache checkbox is selected. Select columns from the dropdown list for mapping. For more information, see Data Table Setting.**

- c. **Rated Column**: Select the column name from dropdown list to calculate fraud loss for the generated cases. Column with Data Type: Integer, Long, Decimal and Float are displayed as option in dropdown list.

- d. Click **Save**.

8. New Data source connection is created.

Create New Category

To create a category in the Kafka Source configuration grid:

1. Click on **Add/Search Category** configuration element. The configuration element is expanded.
2. Enter the category name in **Add/Search** category.
3. Click . A category is added.
4. Go back to step *Tag*, to configure the next step of the **Kafka source** configuration.

Kafka - Additional Configurations

To configure the additional key - value pairs:

1. In **Additional Configurations** pane, click . A new row is added.

Configure the following:

- **Key**: Indicates how to deserialize the message keys to string.
- **Value**: Indicates how to deserialize the message values to string.

2. Click **Save**.

3. To learn how to configure Transform Operator, refer to the step [Transform Operator](#) in the Configure a Pipeline page.

S3 Source

Last updated on December 05, 2024

S3 Source is an object storage suite capable of handling structured and unstructured data including log files, artifacts, backups, container images, photos and videos. You can either use Amazon S3 or MinIO as your S3 Source.

To create a new Data Reader - S3 Source:

1. In the Canvas, go to **Operators grid** > click . The **Configure Data Reader** window is displayed.
2. Select **S3 Source** and click **Next**. **Connection Details** tab is displayed. You can either use the existing connections or can create new S3 Source connection.
3. To create new connection, click . The configuration panes are enabled.
4. Click the expand icon to configure General Details.

Configure the following:

- a. **Data Reader Name:** Enter the name for S3 Source Data Reader.
 - b. **Description:** Allows you to enter the description for S3 Source data reader.
 - c. **Add/Search Category:** Select the category to which the S3 Source belong to. You can also create a new category to add the sources. For more information on creating category, see [Create New Category](#).
 - d. **Tag:** Enter the tags for the S3 source, that helps in identifying the purpose of data reader in pipeline.
5. Click the expand icon to configure Configuration.

Configure the following:

- a. **IP/Host:** Enter the IP or host details (URL) for S3, that you want to connect with. **Example:** `http://10.1yy.1xx.2:9aaa/` indicates the URL of S3 server with the machine ip: 10.1yy.1xx.2 and port: 9aaa.
- b. **Access Key:** Enter the access key associated with the configured S3 server.
- c. **Secret Key:** Enter the secret key associated with the configured S3 server.
- d. **Directory:** Enter the directory of S3 storage, from where you want to fetch the data.
- e. **Compression type:** Select the compression type. Example: If the selected format is gzip, the input data is compressed and stored in gzip format after it is fetched.
- f. **Backup Strategy:** Select the backup strategy for the loaded file if it must be deleted, or archived after the file is processed.

- i. **OFF:** Does not consider any backup strategy.
 - ii. **Archive:** All the files except the recent processed file from source directory are archived to the configured **archive directory path** after the file is processed.
 - iii. **Delete:** All the files except the recent processed file from source directory are deleted after the file is processed.
- g. **File Extension Pattern:** Decides the file extension pattern. Example: **.*.txt**- Accepts only files which has **.txt** extension.
csv and **txt** files are supported.
- h. **Max File Age(In seconds):** Indicates the files to fetch within the configured time when a pipeline with this Data Source is run. **Example:** If we have configured 60 seconds; When the pipeline with this data source is run, it will the files will be fetched within 60 seconds based on the **Max File Per Trigger** parameter value.
- i. **Max Files Per Trigger:** Enter the number of files that must be accepted. Example: If there are **5 files in the source directory, and value configured for Max Files Per Trigger is 2, then 2** files are processed at a time. And the 5 files are processed in 3 attempts.
- j. **File Format:** Select the file format to to be fetched from S3. **CSV** and **JSON** formats are supported.
- k. **Delimiter:** Enter the delimiter to separate the values that are being fetched in S3.
- l. **Is Header Present:** If enabled, the file without header is also fetched.
- m. **Click Save.**
6. Click **Next**. A **Preview** tab is displayed. Upload the file and preview.
7. Click **Next**. The application navigates to the **Dataset Tagging**.

Complete the below details:

- a. **Dataset Tagging:** It populates the data table. Select the required data table to tag with the source connection.

Note:

- Select the **None** option from the **Tag Dataset** dropdown field, to create basic data source configuration.
- Select table options from the **Tag Dataset** dropdown field. This process is called as **Participation Record**. For more information, refer to *Participation Record*.

- b. **XDR_ID:** It populates the column associated with the data table. Select primary key or column with unique values. It is mandatory field.

Note:

- This field is disabled if Dataset Tagging is selected as **None**.

- Data table whose dataset tagging is done, make sure **Enable Local Cache** checkbox is selected. Select columns from the dropdown list for mapping. For more information, see [Data Table Setting](#).

c. **Rated Column**: Select the column name from dropdown list to calculate fraud loss for the generated cases. Column with Data Type: Integer, Long, Decimal and Float are displayed as option in dropdown list.

d. Click **Save**.

8. New Data Source connection is created.

Note:

- The created S3 Source is available in the [Operators > Data Reader Grid](#) for future usage.
- After all the configurations are completed, save and run the Pipeline.

After S3 source is configured, the output of S3 Source can be connected to Transform Operators.

9. To learn how to configure Transform Operator, refer to the step [Transform Operator](#) in the [Configure a Pipeline](#) page.

Create New Category

To create a category in the S3 Source configuration grid:

1. Click on **Add/Search Category** configuration element. The configuration element is expanded.
2. Enter the category name in **Add/Search Category**.
3. Click . A category is added.
4. Go back to step [Tag](#), to configure the next step of the S3 source configuration.

GCS Source

Last updated on December 05, 2024

GCS Source that provides object storage through a web service interface.

To create a new Data Reader:

1. In the Canvas, go to **Operators grid > click** . The **Configure Data Reader** window is displayed.
2. Select **GCS** and click **Next**. **Connection Details** tab is displayed. You can either **Create New Connection** or **use the existing connections**.
3. To create new connection, click . The configuration panes are enabled.

4. Click expand icon to configure General Details.

Configure the following:

- a. **Data Reader Name:** Enter the name for GCS Source Data Reader.
- b. **Description:** Allows you to enter the description for GCS Source data reader.
- c. **Add/Search Category:** Select the category to which the GCS Source belong to. You can also create a new category to add the sources. For more information on creating category, see [Create New Category](#).
- d. **Tag:** Enter the tags for the GCS source, that helps in identifying the purpose of data reader in pipeline.

5. Click expand icon to configure Configuration.

Configure the following:

- a. **Upload File:** Allows you to upload a json file to authenticate the GCS connection. Click icon to select file from the system.
- b. **KeyJson:** Displays the preview of the JSON file.
- c. **Directory:** Enter the directory of GCS, from where you want to fetch the data.
- d. **Compression type:** Select the compression type. Example: If the selected format is gzip, the input data is compressed and stored in gzip format after it is fetched.
- e. **Backup Strategy:** Select the backup strategy for the loaded file if it must be deleted, or archived after the file is processed.
 - i. **OFF:** Does not consider any backup strategy.
 - ii. **Archive:** All the files except the recent processed file from source directory are archived to the configured **archive directory path** after the file is processed.
 - iii. **Delete:** All the files except the recent processed file from source directory are deleted after the file is processed.
- f. **File Extension Pattern:** Decides the file accept pattern. Example: *.txt- Accepts only files which has .txt extension. csv and txt files are supported.
- g. **Max File Age(in seconds):** Indicates the files to fetch within the configured time when a pipeline with this Data Source is run. **Example:** If we have configured 60 seconds; When the pipeline with this data source is run, it will the files will be fetched within 60 seconds based on the **Max File Per Trigger** parameter value.
- h. **Max Files Per Trigger:** Enter the number of files that must be accepted. Example: If there are 5 files in the source directory, and value configured for **Max Files Per Trigger** is 2, then 2 files are processed at a time. And the 5 files are processed in 3 attempts.
- i. **File Format:** Select the file format to to be fetched from GCS. **CSV and JSON formats are supported. If you select the file format as CSV,** then the below two additional fields are enabled:
 - j. **Delimiter:** Enter the delimiter to separate the values that are being fetched in GCS.

- ii. **Is Header Present:** If enabled, the file without header is also fetched.
 - j. Click **Save**.
6. Click **Next**. A **Preview** tab is displayed. Upload the file and preview.
7. Click **Next**. The application navigates to the **Dataset Tagging**.

Complete the below details:

- a. **Dataset Tagging:** It populates the data table. Select the required data table to tag with the source connection.

Note:

- Select the **None** option from the **Tag Dataset** dropdown field, to create basic data source configuration.
- Select table options from the **Tag Dataset** dropdown field. This process is called as **Participation Record**. For more information, refer to *Participation Record*.

- b. **XDR_ID:** It populates the column associated with the data table. Select primary key or column with unique values. It is mandatory field.

Note:

- This field is disabled if Dataset Tagging is selected as **None**.
- **Data table whose dataset tagging is done, make sure Enable Local Cache checkbox is selected. Select columns from the dropdown list for mapping. For more information, see Data Table *Setting*.**

- c. **Rated Column:** Select the column name from dropdown list to calculate fraud loss for the generated cases. Column with Data Type: Integer, Long, Decimal and Float are displayed as option in dropdown list.
 - d. Click **Save**.
8. New Data Source connection is created.

Note :

- The created GCS Source is available in the Operators > Data Reader grid for future usage.
- After all the configurations are done, save and run the Pipeline.

After GCS source is configured, the output of GCS Source can be connected to Transform Operators.

9. To learn how to configure Transform Operator, refer to the step *Transform Operator* in the *Configure a Pipeline* page.

Create New Category

To create a category in the GCS Source configuration grid:

1. Click on **Add/Search Category** configuration element. The configuration element is expanded.
2. Enter the category name in **Add/Search** category.
3. Click . A category is added.
4. Go back to step [Tag](#), to configure the next step of the **Database** configuration.

Azure Source

Last updated on December 05, 2024

Azure Source is used to read data in CSV or Json format from the mentioned azure storage.

To create a new Data Reader - Azure Source:

1. In the Canvas, go to **Operators grid** > click . The **Configure Data Reader** window is displayed.
2. Select **Azure Source** and click **Next**. **Connection Details** tab is displayed. You can either **Create New Connection** or use the existing connections.
3. To create new connection, click . The configuration panes are enabled.
4. Click the expand icon to configure **General Details**.

Configure the following:

- a. **Data Reader Name:** Enter the name for Azure Source Data Reader.
 - b. **Description:** Allows you to enter the description for Azure Source data reader.
 - c. **Select Category:** Select the category to which the Azure Source belong to. You can also create a new category to add the sources. For more information on creating category, see [Create New Category](#).
 - d. **Tag:** Enter the tags for the Azure Source, that helps in identifying the purpose of data reader in pipeline.
 - e. Click **Next**.
5. Click the expand icon to configure **Configuration**.

Configure the following:

- a. **Account Name:** Enter the account name.
- b. **Account Key:** Enter the account key.
- c. **Container Name:** Enter the container name.
- d. **Directory:** Enter the directory of Azure storage, from where you want to fetch the data.

- e. **Compression type:** Select the compression type. Example: If the selected format is gzip, the input data is compressed and stored in gzip format after it is fetched.
 - f. **File Extension Pattern:** Decides the file e-pattern. Example: *.txt- Accepts only files which has .txt extension.
csv and txt files are supported.
 - g. **Backup Strategy:** Select the backup strategy for the loaded file if it must be deleted, or archived after the file is processed.
 - i. **OFF:** Does not consider any backup strategy.
 - ii. **Archive:** All the files except the recent processed file from source directory are archived to the configured **archive directory path** after the file is processed.
 - iii. **Delete:** All the files except the recent processed file from source directory are deleted after the file is processed.
 - h. **Max File Age(in seconds):** Indicates the files to fetch within the configured time when a pipeline with this Data Source is run. **Example:** If we have configured 60 seconds; When the pipeline with this data source is run, it will the files will be fetched within 60 seconds based on the **Max File Per Trigger** parameter value.
 - i. **Max Files Per Trigger:** Enter the number of files that must be accepted. Example: If there are 5 files in the source directory, and value configured for **Max Files Per Trigger** is 2, then 2 files are processed at a time. And the 5 files are processed in 3 attempts.
 - j. **File Format:** Select the file format to to be fetched from Azure. CSV and JSON formats are supported. If you select the file format as CSV, then the below two additional fields are enabled:
 - i. **Delimiter:** Enter the delimiter to separate the values that are being fetched in Azure.
 - ii. **Is Header Present:** If enabled, the first row of the file is ignored by considering it to be the header.
 - k. **Click Save.**
6. Click **Next**. A **Preview** tab is displayed. Upload the file and preview.
7. Click **Next**. The application navigates to the **Dataset Tagging**.

Complete the below details:

- a. **Dataset Tagging:** It populates the data table. Select the required data table to tag with the source connection.

Note:

- Select the **None** option from the **Tag Dataset** dropdown field, to create basic data source configuration.
- Select table options from the **Tag Dataset** dropdown field. This process is called as **Participation Record**. For more information, refer to *Participation Record*.

- b. **XDR_ID:** It populates the column associated with the data table. Select primary key or column with unique values. It is mandatory field.

Note:

- This field is disabled if Dataset Tagging is selected as **None**.
- **Data table whose dataset tagging is done, make sure Enable Local Cache checkbox is selected. Select columns from the dropdown list for mapping. For more information, see Data Table *Setting*.**

- c. **Rated Column:** Select the column name from dropdown list to calculate fraud loss for the generated cases. Column with Data Type: Integer, Long, Decimal and Float are displayed as option in dropdown list.

- d. Click **Save**.

8. New Data Source connection is created.

Note :

- The created Azure Source is available in the Operators > Data Writers grid > specific category for future usage.
- **After all the configurations are done, save and run the Pipeline.**

After Azure source is configured, the output of Azure Source can be connected to Transform Operators.

9. To learn how to configure Transform Operator, refer to the step [Transform Operator](#) in the Configure a Pipeline page.

Create New Category

To create a category in the Azure Source configuration grid:

1. Click on **Select Category** configuration element. The configuration element is expanded.
2. Enter the category name in **Search Category**.
3. Click . A category is added.
4. Go back to step [Tag](#), to configure the next step of the **Database** configuration.

Transform Operators

Filter

Last updated on October 19, 2023

Filter component allows you to filter the input data received from Data Reader/ Streaming related Transform Operators. You can provide the expression to validate, based on which the data is fil-

tered. The expression validation is based on the Boolean value. Filter operator contains 3 ports (I,T and F). I is for Input, T is for True and F is for False and Filter will provide both the Valid and Invalid Output based on the condition added, via Ports T and F.

Example: If Condition in Filter is $ID > 10$ then

T Port will give output based on true condition $ID > 10$ and F Port will give output based on False Condition $ID < 10$.

Prerequisite: You must configure the Data reader in the canvas page before proceeding with the Filter configuration. Once the Data reader is configured, follow the below process to configure the Filter.

Note:

This filter is used for online or streaming pipelines.

To configure Filter:

1. In the Canvas, go to **Operators grid > Transform Operators**.
2. Drag and drop the **Filter** to **Canvas > Configuration pane**.
3. "I" represents the input port of the filter. Provide the output of other operator to the **Filter component "I"** port. Only one input can be given to a filter.
4. There are two types of output from a filter component, **True (T)** and **False (F)** ports. Instead of creating two different filter component for true and false output, the user can use the same filter component and connect other operators to the **True** and **False** port. Multiple output can be connected to the filter output T & F port.
5. Click **vertical ellipses > Edit**. The **Configure Filter** window is displayed.
6. Configure the following:
 - a. **Filter Name:** Specify the filter name.
 - b. **Expression:** You can enter the Expression by selecting required columns from suggestions after performing Ctrl+ Space or you can enter the column name through the keyboard. In the Expression field you can configure the formula with different mathematical operations like arithmetic and trigonometry form. For more expression information, refer to [Add formula](#).

Note:

Suggestions are displayed if Auto Suggestion option is enabled in Data Catalog Table Setting. For more information, refer to [Auto Suggestion](#).

- c. Filter component has following actions:
 - : Click to validate the expression and check if there are any syntactical errors.
 - : On click, the selected expression is indented to right.
 - : Click to clear the expression configured.
7. Click **Save**.
8. **Holiday List is used in the filters to process the records in online pipelines. You can use operators In and Not In.**
 - a. **For Example: You have created Holiday List HL1 for Christmas that is 25th December. We are using Date column to use the calendar date. Suppose present calendar date is 25th December. In expression box, you mention expression as **Date In HL1**, which is True. In this case, 25th December record is processed and sent to Filter True terminal. All other date records are skipped and sent to Filter False terminal.**
 - b. **For Example: You have created Holiday List HL2 for New Year that is 1st January. We are using Date column to use the calendar date. Suppose present calendar date is 2nd January. In expression box, you mention expression as **Date Not In HL2**, which is True. In this case, 1st January record is skipped and sent to Filter False Terminal. All other date records are processed and sent to Filter True Terminal.**
9. You can connect the output of Filter component to other operators also. To learn about other Operators, refer to [Transform Operator](#).
10. To learn how to configure Data Writer, refer to the step [Data Writer](#) in the Configure a Pipeline page.

Batch Filter

Last updated on October 19, 2023

Filter component allows you to filter the input data received from Transform Operators. You can provide the expression to validate, based on which the data is filtered. The expression validation is based on the Boolean value. Filter operator contains 3 ports (I,T and F). I is for Input, T is for True and F is for False and Filter will provide both the Valid and Invalid Output based on the condition added, via Ports T and F.

Example: If Condition in Filter is $ID > 10$ then

T Port will give output based on true condition $ID > 10$ and F Port will give output based on False Condition $ID < 10$.

Prerequisite: To configure batch filter, input port should be connected to any operator output.

Note:

This filter is used for offline or batch processing pipelines.

To configure Batch Filter:

1. In the Canvas, go to **Operators** grid > **Transform Operators**.
2. Drag and drop the **Batch Filter** to **Canvas > Configuration pane**.
3. "I" represents the input port of the filter. Provide the output of other operator to the **Filter component "I" port**. Only one input can be given to a filter.
4. There are two types of output from a filter component, **True (T)** and **False (F) ports**. Instead of creating two different filter component for true and false output, the user can use the same filter component and connect other operators to the **True** and **False port**. **Multiple output can be connected to the filter output T & F port**.
5. Click **vertical ellipses > Edit**. The **Configure Batch Filter** window is displayed.
6. Configure the following:
 - a. **Filter Name**: Specify the filter name.
 - b. **Expression**: You can enter the Expression by selecting required columns from suggestions after performing Ctrl+ Space or you can enter the column name through the keyboard. In the Expression field you can configure the formula with different mathematical operations like arithmetic and trigonometry form. For more expression information, refer to [Add formula](#).
 - i. You can use statement functions to write expression. For more information, refer to [Statement Functions](#).
 - c. Filter component has following actions:
 - i. : Click to validate the expression and check if there are any syntactical errors.

Note:

Suggestions are displayed if Auto Suggestion option is enabled in Data Catalog Table Setting. For more information, refer to [Auto Suggestion](#).
 - ii. : On click, the selected expression is indented to right.
 - iii. : Click to clear the expression configured. It is enabled once you write the expression.
7. Click **Save**.

8. Holiday List is used in the batch filters to process the records in offline pipelines. You can use operators **In** and **Not In**.
 - a. **For Example: You have created Holiday List HL1 for Christmas that is 25th December. We are using Date column to use the calendar date. Suppose present calendar date is 25th December.** In expression box, you mention expression as **Date In HL1**, which is True. In this case, 25th December record is processed and sent to Filter True terminal. All other date records are skipped and sent to Filter False terminal.
 - b. **For Example: You have created Holiday List HL2 for New Year that is 1st January. We are using Date column to use the calendar date. Suppose present calendar date is 2nd January.** In expression box, you mention expression as **Date Not In HL2**, which is True. In this case, 1st January record is skipped and sent to Filter False Terminal. All other date records are processed and sent to Filter True Terminal.
9. You can connect the output of Batch Filter component to other operators also. To learn about other Operators, refer to [Transform Operator](#).
10. To learn how to configure Data Writer, refer to the step [Data Writer](#) in the Configure a Pipeline page.

Statement Functions

Below are the list of the statement functions that you can use in the expression.

| Sl. No | Function | Description | Format | Example |
|--------|--------------------------|--|---|---|
| 1 | cast() / tointeger() | it will convert data type from string to integer and return the integer value. In Batch Filter , use this function as tointeger(). | ~ cast(Arg1 as int) (or) ~ tointeger(Arg1) | Example: cast('900' as int) Explanation: Here data type for 900 is string. Data type is first converted to integer. And 900 is stored as integer value. Output: 900. |
| 2 | char_length() / length() | This function will calculate the length of the string. It will count white space, special characters. | ~ char_length(Arg1) (or) ~ length(Arg1) where, Arg1 is String | Example 1: char_length('Mumbai') Output: 6 Example 2: length('India') Output: 5 |

| | | | | |
|---|----------------------------|---|--|--|
| | | <p>In case of numeric value, Date or Time-stamp, column type should be varchar or char, else function will return 0.</p> <p>In Batch Filter, use this function as length().</p> | | |
| 3 | coalesce() / tonvl() | <p>This function will return first non-null value. In case all the arguments are null, then it will consider only last argument.</p> <p>In Batch Filter, use this function as tonvl().</p> | <p>~ coalesce(Arg1,Arg2,Arg3,...) (or) ~ tonvl(arg1,arg2)</p> | <p>Example 1: coalesce(null,null,'John') Output: John</p> <p>Example 2: tonvl(null,'Marie') Output: Marie</p> <p>Example 3: tonvl(a,b) Output: a [Here both a and b are non null value].</p> |
| 4 | concat() | <p>This function will join 2 or more strings. You can use this function to join 2 or more array.</p> | ~ concat(Arg1, Arg2, Arg3,...) | <p>Example 1: concat('We',' are',' human') Output: We are human</p> <p>Example 2: concat(array(1, 2, 3), array(4, 5), array(6)) Output: [1,2,3,4,5,6]</p> |
| 5 | date_trunc() / datetrunc() | <p>This function will truncate the timestamp as per the specified format.</p> <p>In Batch Filter, use this function as datetrunc().</p> | <p>~ date_trunc(String Format, Column Timestamp) where, Format can be either year, month, date, hour, minute, seconds or any specific combination. example: date_trunc('yyyy'/'mm'/dd'/'HH'/'ss'/'MM', dd-mm-yyyy hh:mm:ss)</p> | <p>Example 1: date_trunc('YEAR', '2015-03-05T09:32:05.359') Output: 2015-01-01 00:00:00</p> <p>Example 2: date_trunc('MM', '2015-03-05T09:32:05.359') Output: 2015-03-01 00:00:00</p> <p>Example 3: date_trunc('DD', '2015-03-05T09:32:05.359')</p> |

| | | | | |
|---|-------------------------------|---|--|--|
| | | | | <p>Output: 2015-03-05 00:00:00</p> <p>Example 4: date_trunc('HOUR', '2015-03-05T09:32:05.359')</p> <p>Output: 2015-03-05 09:00:00</p> <p>Example 5: date_trunc('MILLISECOND', '2015-03-05T09:32:05.123456')</p> <p>Output: 2015-03-05 09:32:05.123</p> |
| 6 | datediff() / datedifference() | <p>This function will provide date difference between 2 timestamp/ date.</p> <p>In Batch Filter, use this function as datedifference().</p> | ~ datediff(timestamp end, timestamp start) | <p>Example 1: datediff("2018-01-10 00:00:00", "2018-01-08 23:59:59")</p> <p>Output 1:2</p> <p>Example 2: datediff("2018-01-08 00:00:00", "2018-01-10 23:59:59")</p> <p>Output 2: -2</p> |
| 7 | dayofmonth() | <p>This function will provide the day of the month for the given date or timestamp.</p> | <p>~ dayofmonth(timestamp)</p> <p>(or)</p> <p>~ dayofmonth(date)</p> | <p>Example: dayof-month('2009-07-30')</p> <p>Output: 30</p> |
| 8 | dayofweek() | <p>This function will provide the day of the week for the given date or timestamp.</p> | <p>~ dayofweek(timestamp)</p> <p>(or)</p> <p>~ dayofweek(date)</p> | <p>Example: dayofweek('2009-07-30')</p> <p>Output: 5 [30th July 2009 is Thursday. Considering Sunday as start of week.]</p> |
| 9 | dayofyear() | <p>This function will provide the day number of the year for the given date or timestamp.</p> | <p>~ dayofyear(timestamp)</p> <p>(or)</p> <p>~ dayofyear(date)</p> | <p>Example: dayofyear('2016-04-09')</p> <p>Output: 100. [Here, 9th April is 100th day of year 2016]</p> |

| | | | | |
|----|-------------------------|--|--|---|
| 10 | hour() / hourofday() | <p>This function will provide the hour of the given time-stamp.</p> <p>In Batch Filter, use this function as hourofday().</p> | <p>~ hour(timestamp) (or) ~ hourofday(timestamp)</p> | <p>Example: hour('2023-07-30 12:58:23')</p> <p>Output: 12</p> |
| 11 | instring() | <p>This function will search for the first occurrence of the substring within a given string and return the index position of occurrence. It will count white spaces and special characters.</p> <p>If Substring is not found return value will be zero.</p> | <p>~ instring(Arg1, Arg2) where, Arg1 - String Arg2 - Substring to be found</p> | <p>Example 1: instr('Hello World', 'World')</p> <p>Explanation: Index position for above string as follows, H:1, E:2, and so on. Arg2 (World) is searched in Arg1 and its index position is returned as 7.</p> <p>Output: 7 [as World is found in Arg1 and index position of occurrence for 'W' in string is at 7th place].</p> <p>Example 2: instr('Hello USA Citizen', America)</p> <p>Explanation: Index position for above string as follows, H:1, E:2, and so on. Position for N is 17. Arg2 (America) is searched in Arg1 and its index position is returned as 0.</p> <p>Output: 0 [as America is not found in the Agr1].</p> |
| 12 | lower() | <p>This function converts string into lower case.</p> | <p>~ lower(Arg1) where, Arg1 is String</p> | <p>Example: lower('John MATHew')</p> <p>Output: john mathew</p> |
| 13 | nestedfunctions() | <p>This function allows you to pass function as argument within a function.</p> | <p>~ function1(function2(function3(Arg2)))</p> | <p>Example: substring('Kangaroo', 1,instr('lower('INDIA')','d'))</p> <p>Explanation: Here, there are 3 functions: substring(), instr(), and lower(). Output of function lower() act as argument to in-</p> |

| | | | | |
|----|------------------------------------|--|---|---|
| | | <p>In this case, the innermost function is executed first and act as argument to nested function.</p> <p>The outermost function is executed last.</p> | | <p>str(). Output of instr() act as Example to substring(). For function lower('INDIA'), output is india. For function instr('india',d), output is 3. For function substring('Kangaroo',1,3), output is Kan.</p> <p>Output: Kan</p> |
| 14 | round() | <p>This function round of the first argument to the nearest value as per second argument.</p> | ~ round(Arg1, Arg2) | <p>Example: round(2.537,2)</p> <p>Output: 2.54</p> |
| 15 | SubString() | <p>This function will extract substring from the given string as per start and end index.</p> | <p>~ substring(Arg1,Arg2,Arg3) where, Arg1 - String Arg2 - Starting Index in String Arg3 - Ending Index in String</p> | <p>Example: substring('Australia', 2,6)</p> <p>Explanation: Index position for above string as follows A:1, U:2, and so on.</p> <p>Output: ustra</p> |
| 16 | timediff() / timestampdifference() | <p>This function will provide time difference between 2 timestamp in millisecond.</p> <p>In Batch Filter, use this function as timestampdifference().</p> | ~ timediff (timestamp end, timestamp start) | <p>Example: timediff("2018-01-10 05:06:23", "2018-01-10 07:06:23")</p> <p>Output: 7200</p> |
| 17 | to_date() / to-date() | <p>This function will convert the column into date type.</p> <p>In Batch Filter, use this func-</p> | <p>~ to_date(Column A, String Format) where, Format can be either year, month, date, hour, minute, seconds or any specific combination.</p> <p>example: to_date(dd-mm-yyyy hh:mm:ss, 'yyyy-mm-dd')</p> | <p>Example 1: to_date(10-12-2023 05:10:20, 'dd-mm-yyyy')</p> <p>Output: 10-12-2023</p> <p>Example 2: todate(10-12-2023 05:10:20, 'mm/dd/yyyy')</p> |

| | | | | |
|----|-------------------|---|---|---|
| | | tion as to-date(). | | Output: 12/10/2023 |
| 18 | tochar() / cast() | This function will convert date or time-stamp into string. In Query Measure(QM) and SQL Meaure(SQLM) , use this function as cast(). | ~ tochar(timestamp) (or) ~ cast(arg1 as string) | Example: tochar(2018-01-10 05:26:43) Output: "2018-01-10 05:26:43" |
| 19 | upper() | This function converts string into upper case. | ~ upper(Arg1) where, Arg1 is String | Example: upper('John MAtthew') Output: JOHN MATHEW |

Reducer

Last updated on May 12, 2023

The Reducer allows you to define the number of columns which will be carry forwarded to further process or output. You can select or deselect the columns from the list of all the columns in the input.

Prerequisite: You must configure the Data reader in the canvas page before proceeding with the Reducer configuration. Once the Data reader is configured, follow the below process to configure the Reducer.

To configure Reducer:

1. In the Canvas, go to **Operators grid > Transform Operators > Reducer**.
2. **Drag and drop the Reducer operator to Canvas > Configuration pane**.
3. Connect the output of other component as an input to the Reducer.
4. Configure the **Reducer**.
5. Click **Save**.

Note :

The output of Reducer can be given to the data sink or any other components (Transform operators).

6. To reduce the columns:

- Select the columns that must be part of the output and click **Save** to save the details.
 - To use the reduced columns, connect the output of Reducer to the component that needs to use the data.
7. To learn how to configure Data Writer, refer to the step [Data Writer](#) in the Configure a Pipeline page.

Mapper

Last updated on May 12, 2023

The Mapper allows you to stitch together output from previous stages and form a single output. You can map fields from different input sources.

To configure the Mapper:

1. In the Canvas, go to Operators > Transform Operators pane.
2. Drag and drop the **Mapper** into the configuration grid.
3. Connect the output of other components as an input to the Mapper.
4. Click vertical ellipses of Mapper > **Edit. Configure Mapper** window is displayed.

Here, as we have provided Scheduler and Reducer as inputs, they are available as columns to be mapped in Mapper.

5. Click to add the new fields into the mapper grid. A new row is added.
6. Configure the following:
 - **Mapper Name:** Enter a name for the Mapper.
 - **Union Mode:** Union mode check box combines the data of selected columns from the multiple sources that are connected to mapper and gives the combined output. Select the checkbox to enable the union mode.
 - **Column Name:** Enter the mapper column name where the data from the different source are combined and stored.

- **Data Type:** After column name is entered, select the data type of the column from the drop-down list. **Example:** If you select Integer datatype, all the columns with integer data type will be available in the respective streams.
- **Multiple Sources:** These columns are streams, where you can view list of all the columns from the multiple source.
- Click **Add New Data Field** and select the columns to be mapped from the drop down list.
- **Delete:** To delete any row, hover the mouse on the row. The delete icon is viewed. Click .

7. Click **Save**. This configuration helps the user to combine the data in **Union Mode**.

Formula

Last updated on May 21, 2024

Formula is the transformation operator that allows you to configure the **formulas based on valid expression** and is executed to compute the results.

To configure Formula:

1. In the Canvas, go to Operators grid > Formula.
2. Drag and drop the Formula operator to Canvas > Configuration pane. Connect the input component/any other transform operator to the Formula component.
3. Click the vertical ellipses > Edit. The Configure Formula window is displayed.
4. Configure the following:
 - **Formula Name:** Enter the formula name.
 - **Add Field:** Click icon to add a new column to configure formula.
 - **Output Column:** Specify the new column on which Formula needs to be executed.
 - **Function:** In the configuration window, enter the expressions to validate. Click Ctrl+Space for suggestions. In the function field you can configure the formula with different mathematical operations like arithmetic and trigonometry. Also, condition can be given in function and output of that condition is boolean. Below are few points to consider while defining the functions.

Note :

- The trigonometry value's will be considered as radian.

- Suggestions are displayed if Auto Suggestion option is enabled in Data Catalog Table Setting. For more information, refer to *Auto Suggestion*.

- 1) You can select the formula from the drop-down suggestions and also enter the operator, operands and functions from the keyboard.
 - 2) You can add expression builder to support for single and multiple argument function.
 - 3) You can use expression builder to support object variable and get value from that.
 - 4) You must have to choose the appropriate data type for the column.
- Data Type: Select the appropriate data type from the drop-down.

Note :

User must have to select the appropriate Data Type with respect to the functional expressions. If the selected Data Type is not matches the functional expression, then the output result will be null.

Example: 1) Condition example: $\text{column1} > \text{column2}$ $\text{column1} = \text{column2}$ $\text{column} = \text{string}$ for condition output data type must be selected as Boolean. 2) Expression example: $\text{column1} + 12$ $\text{column2} + 15.5$

- To add more formula, refer to below table:

| Sl.No | Function / Expression | Use Cases | Output Data Type |
|-------|-----------------------|--|---------------------------------|
| 1 | Age | System timestamp - given timestamp column | Long |
| 2 | Time-stamp Difference | 1. In the same record difference two fields 2. In the current record.timestamp - previous record.timestamp 3. Supports difference in Days/Hours/Minutes/Secs | Long |
| 3 | Date Difference | Date difference between 2 fields | Integer |
| 4 | DateAdd | Add Days to the current date | Date |
| 5 | Absolute | Absolute of value | Integer / Float / Double / Long |

| | | | |
|----|--------------------|--|---------------------------------|
| 6 | Square-Root | SquareRoot of value | Float / Double |
| 7 | Sin | Sin of Value | Float / Double |
| 8 | Cos | Cos of value | Float / Double |
| 9 | Tan | Tan of value | Float / Double |
| 10 | Exponent | Exponent of value | Float / Double |
| 11 | Power | power to factor of n that can be entered | Float / Double |
| 12 | + | Add | Integer / Float / Double / Long |
| 13 | - | Subtract | Integer / Float / Double / Long |
| 14 | / | Divide | Integer / Float / Double / Long |
| 15 | = | Equal | Boolean |
| 16 | And | And condition | Boolean |
| 17 | Or | Or condition | Boolean |
| 18 | If else If | Should be able to make if else condition and further nested if is not supported. For more configuration, Refer to Conditions_for_Configuration . | Boolean |
| 19 | HourOf-Day | Gets the Hour of the day | Integer |
| 20 | Day-OfYear | Gets day of Year | Integer |
| 21 | DayOf-Month | Gets the day of Month | Integer |
| 22 | Day-OfWeek | Gets the day of Week | Integer |
| 23 | Regex-Match | Matches a column with regular expression provided in 2nd arg. | Boolean |

| | | | |
|----|----------------------------|---|--|
| 24 | Round | Rounds the value to precision mentioned | Float / Double |
| 25 | Mod | Returns the remainder or signed remainder of a division | Integer / Float / Double / Long |
| 26 | Match | Returns Boolean, If the data present in the input column matches to the data present in the column of RDBMS table. For more information, refer to Match Function. | String / Boolean |
| 27 | Match-Count | Gets the count or number of repetitive matched data from the input and RDBMS table | String / Integer / Float / Double / Long |
| 29 | External API(AIML) Objects | It will give you prediction,Probability_0.0, Probability_1.0 as output from AI trained model objects with Configured input & output information. For example: The simbox, which contains Configured input & output field. In the Configured input & output field provide the input information in the respective fields and select the output method to configure API function. The data type for respective Output is as below: 1. Prediction - Integer 2. Probability_0.0 - Float 3. Probability_1.0 - Float | Integer / Float |
| 30 | LPMV | Returns the selected reference table column value, If the data present in the input column matches to the data present in the column of reference table. | String |
| 31 | Has | Returns value as True if the input column has mentioned subscriber group. Function Syntax: Arg1 Has (Arg2), where Arg1 is input column name and Arg2 is Subscriber Group name. Data Type for Arg1 and Arg2 is String. | Boolean |
| 32 | Not Has | Returns value as True if the input column does not have mentioned subscriber group. Function Syntax: Arg1 Has Not (Arg2), where Arg1 is input column name and Arg2 is Subscriber Group name. | Boolean |

| | | | |
|--|--|--|--|
| | | Data Type for Arg1 and Arg2 is String. | |
|--|--|--|--|

Note :

User must know, how to configure the mathematical expressions in the Function tab with the particular Data Type, for more, refer above table.

- Below are few points to consider while configuring the formula:
 - Inside the if and elseif functional expression, you must have to provide only conditional or logical expressions.
 - Inside the then and else functional expression, you must have to provide only arithmetic expressions.
- Match function condition is as below:
 - Before run the pipeline, enable the Cache Settings as below:
 - Go to Data Catalog > select the added table > click View > select Cache Settings tab.
 - Click checkbox and enable the cache settings.
 - Select the String_Text column from the Select Columns dropdown list.
 - Click Apply. For more information, Refer to [Data Catalog](#).

Note :

If you want to perform the pattern matching(Inside match function) then only enable the cache otherwise, you can disable it. (Pattern matching will work only when the cache is enabled).

- The Match function pattern is as below:
 - If the RDBMS table contains data as "123%" and Input table contains data as "12378". In side the Match function, Regex Match pattern finds and provide the output by matching the regular expression.

Note :

Regex Match is a functional pattern which is present in the Match function.

For example Regex Match pattern is:

| pattern | Match values |
|---------|--------------|
| | |
| 123% | 12378 |
| %444% | 27384447282 |
| 666_ | 6667 |
| %789_ | 1237890 |

For example Match function formula configuration is:

5. Click **Save**.
6. The output of Formula operator can further be connected to other operator or data sink.

Configured Example:

Batch Formula

Last updated on May 21, 2024

Batch Formula is the transformation operator used in batch processing (offline) pipelines that allows you to configure the **formulas** based on valid expressions and is executed to compute the results.

Steps to configure Batch Formula as follows:

1. Create a new (or) Edit the existing **Offline Pipeline**.
2. In the Canvas, expand **Transform Operators**. Drag and Drop **Batch Formula** from panel.
3. **Connect the input component/any other transform operator to the Batch Formula.**
4. Click the vertical ellipses > Edit. **Configure Batch Formula window is displayed.**
5. Configure the following:
 - a. **Formula Name:** It is unique name to identify the formula. Enter the formula name.
 - b. **Add Field:** Click Add Field to add a new column to configure a formula. Once new row is added, complete below details:
 - i. **Output Column:** Enter the output column name to which Formula output needs to be stored.
 - ii. **Function:** In the configuration window, enter the expressions to validate. **Click and Press Ctrl+Space for suggestions. In the function field you can configure the formula with different mathematical operations such as arithmetic and trigonometry. Also, condition can be given in function and output of that condition is boolean. For more information, refers to [Functions](#).**

Below are few points to consider while defining the functions.

- 1) You can select the formula from the drop-down suggestions and also enter the operator, operands and functions from the keyboard.
- 2) You can add expression builder to support for single and multiple argument function.
- 3) You can use expression builder to support object variable and get value from that.
- 4) You must have to choose the appropriate data type for the column.

Note:

- The trigonometry value's will be considered as radian.
- Suggestions are displayed if Auto Suggestion option is enabled in Data Catalog Table Setting. For more information, refer to [Auto Suggestion](#).

- iii. **Data Type:** Select the appropriate data type from the drop-down.

Note:

User must have to select the appropriate Data Type with respect to the functional expressions. If the selected Data Type is not matches the functional expression, then the output result will be null.

6. Click **Save**. Snackbar message is displayed - "Formula saved successfully".
7. Output from Batch Formula is connected to other operators or data writer.
8. Configured Example:

Functions available in Batch Formula

Batch Formula has various list of function that you can use to define formula. Few functions are listed as below:

| Sl.No | Function / Expression | Description | Output Data Type |
|-------|-----------------------|--|---------------------------------|
| 1 | Age | System timestamp - given timestamp column | Long |
| 2 | Timestamp Difference | 1. In the same record difference two fields 2. In the current record.timestamp - previous record.timestamp 3. Supports difference in Days/Hours/Minutes/Secs | Long |
| 3 | Date Difference | Date difference between 2 fields | Integer |
| 4 | DateAdd | Add Days to the current date | Date |
| 5 | Absolute | Absolute of value | Integer / Float / Double / Long |
| 6 | SquareRoot | SquareRoot of value | Float / Double |

| | | | |
|----|------------|--|---------------------------------|
| 7 | Sin | Sin of Value | Float / Double |
| 8 | Cos | Cos of value | Float / Double |
| 9 | Tan | Tan of value | Float / Double |
| 10 | Exponent | Exponent of value | Float / Double |
| 11 | Power | power to factor of n that can be entered | Float / Double |
| 12 | + | Add | Integer / Float / Double / Long |
| 13 | - | Subtract | Integer / Float / Double / Long |
| 14 | / | Divide | Integer / Float / Double / Long |
| 15 | = | Equal | Boolean |
| 16 | And | And condition | Boolean |
| 17 | Or | Or condition | Boolean |
| 18 | If else If | Should be able to make if else condition and further nested if is not supported. For more configuration, Refer to Conditions for Configuration. | Boolean |
| 19 | HourOfDay | Gets the Hour of the day | Integer |
| 20 | DayOfYear | Gets day of Year | Integer |
| 21 | DayOfMonth | Gets the day of Month | Integer |
| 22 | DayOfWeek | Gets the day of Week | Integer |
| 23 | RegexMatch | Matches a column with regular expression provided in 2nd arg. | Boolean |
| 24 | Round | Rounds the value to precision mentioned | Float / Double |
| 25 | Mod | Returns the remainder or signed remainder of a division | Integer / Float / Double / Long |
| 26 | Match | Returns Boolean, If the data present in the input column matches to the data present in the column of | String / Boolean |

| | | | |
|----|----------------------------|--|--|
| | | RDBMS table. For more information, refer to Match Function . | |
| 27 | MatchCount | Gets the count or number of repetitive matched data from the input and RDBMS table | String / Integer / Float / Double / Long |
| 29 | External API(AIML) Objects | It will give you prediction,Probability_0.0, Probability_1.0 as output from AI trained model objects with Configured input & output information. For example: The simbox, which contains Configured input & output field. In the Configured input & output field provide the input information in the respective fields and select the output method to configure API function. The data type for respective Output is as below: 1. Prediction - Integer 2. Probability_0.0 - Float 3. Probability_1.0 - Float | Integer / Float |
| 30 | LPMV | Returns the selected reference table column value, If the data present in the input column matches to the data present in the column of reference table. | String |
| 31 | Has | Returns value as True if the input column has mentioned subscriber group. Function Syntax: Arg1 Has (Arg2), where Arg1 is input column name and Arg2 is Subscriber Group name. Data Type for Arg1 and Arg2 is String. | Boolean |
| 32 | Not Has | Returns value as True if the input column does not have mentioned subscriber group. Function Syntax: Arg1 Has Not (Arg2), where Arg1 is input column name and Arg2 is Subscriber Group name. Data Type for Arg1 and Arg2 is String. | Boolean |

Points to be considered while configuring formula

Points to be considered while configuring different formulas:

1. If Else Function:

- Inside the if and elseif functional expression, you must have to provide only conditional or logical expressions.
- Inside the then and else functional expression, you must have to provide only arithmetic expressions.

2. Match Function:

- Before run the pipeline, enable the Cache Settings as below:

- i. Go to Data Catalog > select the added table > click View > select Cache Settings tab.
- ii. Click checkbox and enable the cache settings.
- iii. Select the String_Text column from the Select Columns dropdown list.
- iv. Click Apply. For more information, Refer to [Data Catalog](#).

Note :

If you want to perform the pattern matching(Inside match function) then only enable the cache otherwise, you can disable it. (Pattern matching will work only when the cache is enabled).

b. Pattern is as below:

- i. If the RDBMS table contains data as "123%" and Input table contains data as "12378". Inside the Match function, Regex Match pattern finds and provide the output by matching the regular expression.

Note :

Regex Match is a functional pattern which is present in the Match function.

- ii. **For example:** Regex Match pattern is:

| pattern | Match values |
|---------|--------------|
| | |
| 123% | 12378 |
| %444% | 27384447282 |
| 666_ | 6667 |
| %789_ | 1237890 |

Configuration as follows:

Summarizer

Last updated on March 18, 2025

Summarizer allows you to configure the profiles, aggregate functions and execute them based on the configured timestamp. For the conditions configured in the Profile, if the filter condition is validated then the aggregator **Function** is executed. The filter condition in **Filter** pane can be mathematical expressions or validating expressions.

Note:

- You cannot add the Join after the Summarizer has been added.
- Summarizer will consider the input data only if it has the input DATETIME column (based on the preview of the timestamp value in uploaded file or input).

To configure Summarizer:

1. In the Canvas, go to Operators grid > Summarizer.
2. Drag and drop the **Summarizer operator** to **Canvas > Configuration** pane. **Connect the input component/ any other transform operator to the Summarizer component.**
3. Click the vertical ellipses > **Edit**. The **Configure Summarizer** window is displayed.
4. Configure the following:
 - a. **Summary Details:**
 - i. **Name:** Enter the Summarizer name.
 - ii. **Timestamp:** Select the timestamp column from the list. Timestamp values are available here only when the input provided to condition has the time stamp values configured.
 - iii. **Watermark:** Allows you to configure the number of days. Data will be in cache until the specified number of days and after which the data is truncated. **Example:** If watermark = 2 days then data in cache is truncated after 2 days.
 - iv. **None:** If selected, then Window Interval and Micro Batch Interval are not considered. Fields are disabled.
 - v. **Sliding:** If selected, data is collected in the batch interval.
 - **Value:** Enter the value to start collecting the data.
 - **Window Interval:** Select the window interval from the list. **Example:** If selected **CURRENT YEAR**, data of that year (based on the last processed timestamp) is collected to conditionalize.
 - **Sliding Interval:** Enter the micro batch interval. The data is collected in the configured batch intervals. **Example:** If timestamp value of processed data has current date and time, window Interval is 10 days and micro batch interval is 2 hours: The 10 days previous data of current date and time is collected. It is fetched in 2 hour batches for those 10 days.
 - vi. **Static:** If selected, data is collected for the static interval such as day, week, and month.
 - **Static:** Select the static interval from dropdown list to collect the data. Available options are as follows:
 - **Current Day:** If this is selected, system will collect and process the CDR for current day till current(system) timestamp. **For example:** Current Day is 30 Nov'23, and Current(System) Timestamp is 15:46 hours. System will collect and process all the CDRs from 30 Nov'23 00:00 hours till 30 Nov'23 15:46 hours.
 - **Current Week:** If this is selected, system will collect and process the CDR for current week till current(system) timestamp. **For example:** You have defined the Start of week as Monday in **Admin > Common Settings > Locale Settings**. Consider Monday as 23 Oct'23, Thursday as 26 Oct'23. You select static option as **Current Week** on Thursday, and Current(System) Timestamp is 13:20 Hours. System will collect and

process the CDRs from Monday 23 Oct'23 00:00 hours till Thursday 26 Oct'23 13:20 hours. To know how to set the Start day of the week, see [Start Day of the Week](#).

- o **Current Month:** If this is selected, system will collect and process the CDR for current month till current(system) timestamp. **For example:** Current Month is November, Day is 25 Nov'23, and Current(System) Timestamp is 20:20 hours. System will collect and process all the CDRs from 01 Nov'23, 00:00 hours till 25 Nov'23, 20:20 hours.
 - **Value:** Enter the value to start collecting the data.
 - **For Example:** Consider Current Day is 30 Nov'2023, and Current(System) Timestamp is 15:45 hours. You have mentioned Static as Current Day, Value as 1, Sliding Interval as Hours. Now, system will start collecting the CDR from 30 Nov'2023 00:00 hours till 30 Nov'2023 15:45 hours. CDR data is collected for 16 different batches, as sliding interval value is 1 hour. First Batch duration will be from 30 Nov'2023 00:00 hours to 30 Nov'2023 00:59 hours, and Last Batch duration will be from 30 Nov'2023 15:00 hours to 30 Nov'2023 15:45 hours.
5. **Filter:** It allows you to add the filter expression, if any. Below are the details to configure:
- a. **Expression:** You can enter the Expression by selecting required columns from suggestions after performing Ctrl+ Space or you can enter the column name through the keyboard.
- Note:**

While using "In" and "Not In" operator, in **Select / Enter Reference Value** window:

 - Maximum allowed entries are:1000.
 - If more than 1000 entries are entered, snack bar message is displayed "**You have exceeded the maximum allowed entries: 1000**".
- b. : Click to validate the expression and check if there are any syntactical errors.
 - c. : Click to clear the expression configured. It is enabled once you write the expression.
6. **Left Selection grid:** This pane has the list of column obtained as an input to the Summarizer. The input columns are listed based on the data types. You can view them based on the Select View. You can select the column into Selected Columns pane based on **Add Action**.
- a. **Search Bar:** Allows you to search specific incoming column.
 - b. **Select View:** Allows you select the required column types like integer, string, or so on.
 - c. **Add Action:** Allows you to select the required function for the selected view (column types).
7. **Select Columns Pane:** Drag and drop the columns from the left selection grid and select the aggregate functions from **Function column**.

OR

You can also select the required function in the left selection grid, to add it to Selected Columns pane.

8. After all the columns are added, edit the Profile Name if required.

- Then select the functions for the **Function** column for the respective selected data row. The options are as below in the table:

| Sl.No | Function Type | Use Cases Type |
|-------|-------------------------------|---|
| 1 | Sum | Sum of entity such duration, values. |
| 2 | Count | Count of entity such as phone number, outgoing calls. |
| 3 | Group By | The group of entities. |
| 4 | Mean | A relation from a set of inputs to a set of possible outputs where each input is related to the exactly one output. |
| 5 | Min | The smallest of the given entity. |
| 6 | Max | The largest of the given entity. |
| 7 | First Value | First value of the entity such as first recharge amount, derive time of the first call of call while roaming. |
| 8 | Last Value | First value of the entity such as last recharge amount, derive time of the last call of call while roaming. |
| 9 | Approximate Count Distinct | To get an approximate idea of the number of non-duplicate rows in a large table or result set. |
| 10 | Variance | To determine the variance of the call. |
| 11 | variance of Population | To determine the variance of the calls made by in the population. |
| 12 | Sum of distinct | This function will calculate the sum for configured field (such as value/duration/number fields) for the distinct element for given dimension. |
| 13 | Variance of sample | To determine the variance of the calls made by in the sample. |
| 14 | Average | List of the central or typical value. |
| 15 | Standard Deviation | To determine the SD of the calls made by in the sample. |
| 16 | Standard Deviation Population | To determine the SD of the calls made by in the population. |

| | | |
|----|------------------------|---|
| 17 | Count of distinct | This function will calculate the count of the distinct element for given dimension. |
| 18 | Collect List | List all value for the configured field for the given dimension. Example: List of all Called Number from Phone number in last 7 days. |
| 19 | Collect Set | List all distinct value for the configured field for the given dimension. List of all distinct Called Number from Phone number in last 7 days. |
| 20 | Size of List | Count of values in above set or list. |
| 21 | Sum per distinct | This function will calculate the sum for configured field (such as value/duration/number fields) for the distinct element for given dimension - Group by Distinct Values. |
| 22 | Count of distinct HLL | This function will calculate the count of the distinct element for given dimension using Hyper Log Log. |
| 23 | Count per distinct | This function will calculate the count of the distinct element per dimension. |
| 24 | Count per distinct CMS | This function will calculate the count of the distinct element per dimension using Count min sketch. |
| 25 | Mode | Return the most frequently appearing element for the given aggregation. What is most called Number/Country from a given called number? Into which country call is getting terminated for given I-out trunk. |
| 26 | Mode Frequency | This returns the count of the mode value. Ex: How many times did caller make calls to most called number. |
| 27 | Median | To determine the median of the calls made by the caller number. |
| 28 | Sequential Dialing | Sequential dialing function is used to figure out if a particular caller number is making outgoing calls in sequence to detect any fraudulent/spam caller numbers. |
| 29 | Has | Returns value as True if the input column has mentioned subscriber group. |

| | | |
|----|---------|---|
| 30 | Not Has | Returns value as True if the input column does not have mentioned subscriber group. |
|----|---------|---|

9. Click **Save**.
10. Summarizer is configured, which can given as an input to other operators.

Note :

If you edit the **Summarizer** details from the existing pipeline and update any changes in the **Timestamp** details and **Selected Columns** details section, then the old checkpoint directory will be deleted or cleaned automatically after saving and a new checkpoint directory will be created once you run the pipeline.

Let us consider, we have to select **String** columns with **Group By** Aggregate function.

1. Go to **Left Selection** grid > **Select View**, select the String type.
2. The selected view is highlighted and **Add Action** is enabled.
3. In **Add Action**, select the required action. Example: Group By
4. The column of **string** type with the **Group By** aggregate Function is added into the rows of **Selected Columns** pane.

SummarizeX

Last updated on December 05, 2024

SummarizeX is a transformation operator that is introduced to configure the profiles, aggregate functions, and execute them based on the configured timestamp. It has additional improvements over the existing Summarizer operator as it supports watermark configuration, multi-windows intervals, and less memory consumption. It is used to generate Participating Records using a Data Copier, which copies the cumulative data of the CDR that violates the FM rules and alarms. Whenever a case gets generated, the data with which it got violated or some data that passed through grouped entity, from the source table of data catalog, a copy action is triggered. As a result, Participated Records (PR) are generated. PR is used to uniquely identify the records in the input table that has contributed to the case generation.

SummarizeX is also used to identify the SIM Cloning Fraud using Velocity Rule.

Note:

- This operator is used for Streaming/Online Pipeline.
- This operator is connected to *Case Operator Sink*.
- This operator supports input from multiple source and multiple window interval configuration.
- To know more about how to configure and generate Participating Record, refer to *Participating Record*.

Multisource support in SummarizeX

SummarizeX support inputs from multiple sources. Use mapper when you want to use multiple source. Columns from the multiple sources must be combined into single column in mapper.

Follow below steps to configure multiple source:

1. Connect the multiple sources with mapper.
2. Configure the mapper. Click vertical ellipsis > **Edit**.
3. **Configure Mapper window is displayed. To know how to configure mapper, refer to [Mapper](#).**
4. **Connect mapper with other transformation operator or SummarizeX.**
5. To know how to configure SummarizerX, refer to [Configuration](#).

Configuration

Follow below steps to configure **SummarizeX**:

1. In the Canvas, go to Operators grid > **SummarizeX**.
2. Drag and drop the **SummarizeX** operator. Provide input link to operator.
3. To configure **SummarizeX**, Click the vertical ellipsis > **Edit**.
4. **Configure SummarizeX window is displayed. SummarizeX has 3 tabs: Filter, Computation and Output.**
5. **Filter Tab: This tab is used to filter the incoming records. You have to write the mathematical expression and validate it.**

- [Click to know the configuration](#)

Complete the below details to configure:

1. **SummarizeX Name:** Enter the name of SummarizeX. Minimum 4 characters are allowed.
2. **Expression Box:** You should write the Expression by selecting required columns. Click **Ctrl+ Space** for auto suggestions. You can configure the expression box with different mathematical formula like arithmetic, trigonometry and so on. For more information, refer to [Add formula](#).
3. Filter component has following actions:
 - a. : **Click to validate the expression and check if there are any syntactical errors. If expression is valid, when clicked **Valid Expression** is displayed. If expression is invalid, when clicked **Invalid Expression** is displayed and part of expression that is incorrect is highlighted in red colour.**
 - b. : Click to indented the expression to right side of the screen.
 - c. : Click to clear the expression configured.

4. Click **Next** button to configure **Computation** tab.
5. Configured example:
6. **Computation Tab:** This tab allows you either to create the window to configure or generate the participating records or to identify the SIM cloning fraud. Incoming CDR records are processed based on the **Group By** column. In this tab you can select either of the option from the dropdown list:
 - a. **Window:** This option allows you to configure and generate the Participating Records.
 - i. *Click to know the configuration*

Complete the below details to configure:

1. **Window Type:** It allows you to select the option to generate Participating Records. By default, **Window** is selected. In case nothing is selected, select **Window from the** dropdown list.
2. **Group By:** It populates the column names from the incoming records. Select the column names to group the records.

Note:

- Selected **Group By** column names are displayed.
- You can select multiple Group By.
- You can **search** for specific column using the search bar.
- Click **Select All** to select all the column for grouping record.
- Click **Clear All** to clear the grouping selection.
- Click **to remove the selected column**.

3. **Window Configuration:** It allows you to configure the window to look for incoming records. Here you have to configure the watermark and window interval. Once intervals are defined, you can add rows. You can also add multiple windows.
 - a. **Watermark:** It means data will be in cache until the specified number of days and after which the data is truncated. Enter the below details:
 - i. **Value:** It should be number. Minimum value should be 1. For example: 1,2,3 and so on.
 - ii. **Watermark:** Select the watermark from dropdown list: **Days, Hours, Minutes or Seconds**. **For Example:** If watermark = 2 days then data in cache is truncated after 2 days.

Note:

Watermark Interval should be greater than Window interval.

- b. **Window:** It allows you to check the incoming record within the selected window interval. Enter the below details:

- i. **Value:** It should be number. Minimum value should be 1. For example: 1,2,3 and so on.
 - ii. **Window Interval:** It allows you to check the incoming record within the selected window interval. Select the window interval from dropdown list: Days, Hours, **Minutes or Seconds**. **For Example:** If window interval = 1 days then record is checked for last 24 hours.
- Note:**
- Window Interval should be lesser than Watermark interval.
- iii. **Timestamp:** It populates the column on which timestamp should be performed. It is used to provide the reference to look-back for the incoming record. Select the timestamp column from the dropdown list.
 - iv. **Click [Click here to add new row](#).** It allows you to filter the incoming records as per mentioned condition and perform selected functions. Below details should be completed:
 - **Column:** It populates the list of column obtained as an input to the SummarizeX. Select the column from the dropdown list.
 - **Function:** It populates aggregate function to be performed on the selected column. Select the function from the dropdown list.
 - **Profiles:** It creates the alias for the selected incoming column. In case you select same incoming column to perform distinct functions, error message is displayed "**Duplicate profile exists**". Manually change the profile name to override the error.
 - **Condition:** It allows you to add the conditions to filter the incoming records. Press **Ctrl + Space** for auto suggestion.
 - v. Click **Save** to save the window configuration.
 - vi. Click **Add Row**, if you want to add more columns to filter the incoming records. You cannot add more rows if profile with similar name exist, manually change the profile name.
 - vii. Delete Icon is available against each window configuration and row. Click **delete** icon to delete the window configuration and rows.
- c. Click **Add Window** to add new window for record lookup with different watermark and window interval.

Note:

You must configure current window before adding new window.

- d. Click **Next** button to configure Output tab.
- e. **Configured example:**

- b. **Velocity:** This feature enables real-time tracking of fraudulent activities by analyzing the speed and geographical movement of calls within a specified time frame. It helps in detecting potential fraud scenarios such as SIM swapping, account takeovers, and subscription fraud.

Note:

As per the velocity configuration, cases will generate. But reoccurrence count for generated cases might vary.

- i. *Click to know the configuration*

Complete the below details:

1. **Window Type:** Select Velocity option from the dropdown list.
2. **Group By:** It populates the column names from the incoming call detail records. Select the column names to group the records. It is preferred to group the record by sender(calling) phone number.

Note:

- Selected **Group By** column names are displayed.
- You can select multiple Group By.
- You can **search** for specific column using the search bar.
- Click **Select All** to select all the column for grouping record.
- Click **Clear All** to clear the grouping selection.
- Click **to remove the selected column**.

3. **Select Key as lookup:** It populates the column name from the incoming CDR table. Select the column name from the dropdown list that will look for location in geographical position table. For example: cell_site_id, here site id from CDR table is looked in the geographical position table.
4. **Timestamp:** It populates the timestamp column available in the incoming call record. Select the timestamp column from the dropdown list to look for the fraud timing. For example: Start Time of call.
5. **Select GeoPosition:** It populates the geographical table name. Select the geographical table name from the dropdown list that will be used to calculate fraud velocity. For example: GP_002

Note:

- While creating Geographical position table in Data Catalog, mention Specify Tags as "**Velocity Rule**".
- For available geolocation table, make sure **Enable Local Cache** checkbox is selected. Select columns from the dropdown list to fetch the details. For more information, see Data Table *Setting*.

- **Select the columns from geoposition table to fetch Timezone, Longitude, Latitude, Cell Site Id, and Cell Site Radius details.**

6. **Select Timezone:** Select the column name from the dropdown list to fetch the time zone. For example: Time Zone.
7. **Select Longitude:** Select the column name from the dropdown list to fetch the longitude details. Longitude details are available in decimal format. For example: longitude_in_dec
8. **Select Latitude:** Select the column name from the dropdown list to fetch the latitude details. Latitude details are available in decimal format. For example: latitude_in_dec
9. **Select Cell Site ID:** Select the column name from the dropdown list to fetch the location id. For example: cell_site_id
10. **Select Radius coverage:** Select the column name from the dropdown list that contains the location radius. For example: radius

Configured Example:

7. **Output:** This tab contains two pane: Left Pane and Right Pane(Selected Columns). Below are the output tab details based on window type:
 - a. *Click to know the output details for Window Type - Window*
 1. **Left Pane displays the list of columns which are input to the SummarizeX.** Column list depends on the number of data source connected. If the column list is long, use search bar to find the specific column.
 2. **Right Pane displays the list of the Selected Columns that is viewed in the final output.** It has Field and Window column. Field displays the column name and Window displays the applicable window interval. Points to be noted:
 - a. **By default, the column used during window configuration is displayed.** It is an aggregated column that cannot be removed from the output table. For identification, in the field column window interval is prefixed to the aggregated column name, and in the window column window interval is displayed. **For example: 1_Days_product_name**, here 1_Days is window interval and product_name is aggregated column.
 - b. **Drag and drop columns from the Left Pane to add more columns.** They are non-aggregated columns. Click the delete icon to remove the columns.
 - c. Column selected in Group By cannot be used.
 3. **Configured example:** In Selected Columns Pane, fields "reg" and "date1" is added from Left Pane.

Note:

Connect the SummarizeX output to the Case Operator Sink. In Case Operator Sink, select Participating Record Enabled checkbox and select the window startTime and end-Time from the Dimension Variables. For more information, refer to *Case Operator Sink*.

- b. *Click to know the output details for Window Type - Velocity*
 1. **Left Pane** displays the list of columns which are input to the SummarizeX. Column list depends on the number of data source connected. If the column list is long, use search bar to find the specific column.
 2. **Right Pane** displays the list of the **Selected Columns** that is viewed in the final output. It has Field and Window column. Field displays the column name and Window displays the applicable window interval. Points to be noted:
 - a. **When configuration window type is Velocity, velocity_violated is displayed. It returns true for Fraud CDR.**
 - b. Drag and drop columns from the Left Pane to add more columns. Click the delete icon to remove the columns.
 - c. Column selected in Group By cannot be used.
 3. **Configured example:**

Note:

Output of Velocity is Boolean i.e., either True or False. Output of SummarizeX should be connected to *Filter* and then written in Data Writer such as *Case Operator Sink*.

1. **Left Pane** contains the list of column obtained as an input to the SummarizeX. If the column list is long, use search bar to find the specific column.
2. **Right Pane** contains the list of the **Selected Columns** that is viewed in the final output. It has Field and Window column. Field displays the column name and Window displays the applicable window interval. Points to be noted:
 - a. **By default, the column used during window configuration is displayed. It is an aggregated column that cannot be removed from the output table. For identification, in the field column window interval is prefixed to the aggregated column name, and in the window column window interval is displayed. For example: 1_Days_product_name,** here 1_Days is window interval and product_name is aggregated column.
 - b. **Drag and drop columns from the Left Pane to add more columns. They are non-aggregated columns. Click the delete icon to remove the columns.**
 - c. Column selected for Group By cannot be used.
3. **Configured example:** In Selected Columns Pane, fields "reg" and "date1" is added from Left Pane.

8. Click **Save** to save the SummarizeX configuration.

Join

Last updated on May 12, 2023

Join operator helps you to join the two input operators (Data Reader). The input operators can be connected together to pass the data to any transform operators.

To configure Join:

1. In the Canvas, configure the two different operators, that must be passed as input to join operator.
2. Go to **Operators** grid > **Transform Operators**.
3. Drag and drop the **Join** to **Canvas > Configuration pane**.
4. Connect the two inputs to join. Let us consider the two inputs as DSV and MinIO. To connect, drag and drop from DSV to L, and from MinIO to R of Join.
5. Provide the output of other join operator to other components. Let us consider we have to connect to three different operators: Summarizer, Mapper, and Formula. To connect, drag and drop from **Join:L** to one operator say **Summarizer**; drag and drop from **Join:R** to **Mapper**; and from **Join:L** to **Formula**.

You can also provide the joined input data to the single operator:

6. Now the operators can be used to create the required pipeline.

Note :

If you edit the **Join** details from the existing pipeline and update any changes in the **General Details of Join Details** section, then the old checkpoint directory will be deleted or cleaned automatically after saving and a new checkpoint directory will be created once you run the pipeline.

Query Measure

Last updated on February 10, 2025

Query measures are used to extract specific metrics from the different control points. Example: Use of query measures to calculate the number of records and duration distribution for each call types for traffic from switch. They are also used to refine or filter output of other measures.

Query measure comes with a query builder wizard, which helps you to create queries in an intuitive way.

A Query Measure fetches data from a table instance of Data Catalog, that allows you to segregate data based on the fields selected from the reference data.

Note :

You can configure Query Measure only with SQL Measure operators.

A KPI can be created for any query measure. For more information, see [KPI](#). To configure a query measure:

1. In the Canvas, go to **Operators** grid > **Transform Operators** > **Query Measure**.
2. Drag and drop the **Query Measure** operator to Canvas > **Configuration** pane.
3. Click the vertical ellipses > **Edit**. The **Configure Query Measure** window is displayed.
4. Click **Query Details** expandable.

Configure the following:

- a. **Query Name:** Enter the query name. **Example:** QM_001
- b. **Maximum Rows:** Enter the number of rows that you want to display in the output. **Example:** If there are 10 rows as resultant data of query measure, and if we have set maximum rows 6. It will display only 6 rows in the output.
- c. **Target Store:** It lists all the connected database and allows you to select the target database type. The output data will be stored in the selected database.
- d. **Return Distinct Row Only:** If this check box is selected, rows with similar data are not to be repeated.
- e. **Enable PR Column Config:** If selected, you can configure the data source columns for PR Configuration. For more information, see [PR Enable](#).

Note:

- To generate PR in QM, input should be a partitioned table.
- Enable Toggle Switch for PR Enable and XDR Enable.
- In Case Operator Batch Sink, Select the QM Source to generate PR. For more information, refer to [Case Operator Batch Sink](#).

5. **Configure Query Measure** window has the following tabs: **Add Source**, **Add Source**, **Join**, **Select Columns**, **Filter**, **Preview** and **Report**. Configure the each tab as below steps.

6. **Add Source:** Add the source tables, on which the query must be executed. The Add Source tab has the *Select Tables* and *Columns* panels:

a. **Select Tables:** This pane has the list of tables available in the Data Catalog. You can select the required tables, on which you want to execute the query measure. It has the features as below:

- **Search / Filter Fields:** Allows you to search the tables.
- **Show Selected toggle switch:** When enabled, it displays the selected tables.
- To add the data tables for a query measure, go to **Select Tables** pane > Hover the mouse on the tables instances. The plus icon is viewed. Click .

Note:

- If a table has associated with it, it indicates the table is not selected for query measure.
- If a table has associated with it, then it indicates the table is selected for query measure.

- **Columns:** This pane lists all the columns available in the selected table. Example: In the below mentioned example, we have added subscriber table and cdr table. Out of which, cdr is selected. The data associated with the cdr table are displayed in the Columns pane.
- **Hourly Timestamp Columns(s):** It allows to process the data as per the selected date and time stamp column. When Hourly Timestamp Column(s) is not selected, records are processed based on partition column.

7. **Joins:** You need add the join conditions in this tab. If more than one table is available as input to the Query Measure, you must create a join to link all the tables and extract the required data. To perform join on any data in query measure:

- a. Go to **Joins** tab. The added table instances are seen in the configuration pane.
- b. Connect the two table instances. The connected instances is viewed as displayed:
- c. Click the dark circle between the two instances. The Join Details window is displayed. Select the required join operation.

Following are the operations associated with join:

- **- Inner Join:** A join of two tables that returns records for which there is a matching value in the field based on which the tables are joined.
- **- Left Join:** The left outer join returns all the records from the left table and those records from the right table that matches with the condition.

- - **Right Join:** The right outer join returns all the records from the right table and those records from the left table that matches the condition.
 - - **Full Outer Join:** The full outer join retrieves all records from both the left and the right table.
- d. Click . A new row is added. Complete the following fields:
- **Column1: Subscriber Column:** Displays the columns from the first table based on the data catalog imported. Select the required column, based on which you want to define the match condition.
 - **Column2:** Displays the columns from the second table based on the data catalog imported. Select the required column, based on which you want to match the variable selected in Column1.
 - **Numeric Operator:** Select the operator to perform action between Column1 and Column2.
- e. Similarly, you can add multiple join conditions. When you add multiple conditions, the column And/Or is enabled.
- **And/Or:** If you want to consider all the configured conditions, select AND. If you want to consider either of the configured conditions, select OR.
- f. Click **Apply**.
8. **Select Columns:** This tab allows you to select the required output columns from each of the input tables that are added in **Add Source** tab. These columns are returned as part of the result when the query is run. For more information, see [Select Columns](#) and [Selected Columns panels](#).

- a. **Select Columns Pane:** The Select Columns pane contains the list of selected tables and associated columns.

Below features are available on the pane:

- i. **Search Bar:** Allows you to search specific column.
 - ii. **Checkbox is introduced at table and column level.** Table level checkbox allows you to select all the columns of that particular table. Column level checkbox allows you to select particular column from the table.
 - iii. **Add All:** It allows you to add all the available columns. When clicked, confirmation window is displayed. Click **Yes** to add all the columns.
 - iv. **Add Selected:** It is displayed when either table or column checkbox is selected. When clicked, confirmation window is displayed. Click **Yes** to add the selected columns.
- b. **Selected Columns Pane:** The Selected Columns pane contains all the columns which you select to be part of the result returned by the Query Measure.
- i. To select the columns:

- Drag and drop the columns of any Table Instance from **Select Columns** pane to **Selected Columns** pane. You can also add with the double click of the respective column in a Table Instance.
- After you drag and drop the columns, the column options displayed in Selected Columns pane.

ii. Below are the list of available columns:

- **Aggregate:** Aggregate functions are built-in functions to perform calculations on data. Aggregate functions return a single value, calculated from multiple values in a table column. Select the required aggregate functions. It has the following functions:
 - **Sum:** Calculates the sum of all the values in the selected column.
 - **Count:** Calculates the total count of values in the selected column.
 - **Min:** Identifies the least value in the selected column.
 - **Max:** Identifies the largest value in the selected column.
- **Statement:** You can select the statement functions to perform the action. For more information, refer to [Statement Functions](#).
- **Column Name:** Displays the column name.
- **Display Name:** Edit or change the display name of the column if required.

Note:



- When you select same column again, message is displayed "**Duplicate display name**".
- If you want to use same column, change the display name.

- **Type:** By default a type associated with selected column is displayed. You can change to different type if required.

Note:

In **Type**, you can select the Date data type from the dropdown list for required row to view the date column for particular data.

- **Partition Column:** It is used to get partitioned table output. It works only for Trino Table.
- **PR Enable:** Toggle switch is available for timestamp column. Enable the switch for the selected column. Toggle switch must be enabled to generate PR.
- **XDR Enable:** Enable Toggle switch to select the XDR column. Atleast one XDR column is required to generate participating record (PR). XDR column hold the unique key to generate PR. Datatype should be bigint. Toggle switch must be enabled to generate PR.
- **Sort By:** Select the sorting order for the output column after a query measure is run. Example: if we select asc, the output columns are listed in the ascending order.


- **Delete:** At the right hand side corner of table header, you can see the  button. Click to delete all the rows. On hover of each row in Selected Columns, you can see . Click to delete the corresponding row.

iii. Click **Save and Continue**.

9. **Filter:** This tab allows you to add an expression or filter to select the criterion that is used to select rows in the Output Table. The entries in this panel are optional steps and can be bypassed, if required. For more information follow the below steps:


a. **Filters** tab has **Choose Condition** and **Choose Having Condition by Group by Filter** panes.

b. **Choose Condition:** To configure the filter conditions follow below steps:


- Click . A row is added.
- Configure the following:
 - **Clause:** Select the clause (where, and, or) required for your conditional expression.
 - **Expression(1):** The selected table's columns are available in the list. Select the first expression for the condition.
 - **Operator:** Select the operator (=, <=, >=, >, <) required to define your expression.
 - **Expression(2):** The selected table's columns are available in the list. Select the second expression in the condition.

Note:

- You can use [Absolute / Relative Date](#) to filter out the record. To know more refer, [Use of Absolute / Relative Date in Expression \(2\)](#).

- **Delete:** click , to delete the row if not required.
- Go back to the [Choose Having Condition by Group by Filter](#).

c. **Choose Having Condition by Group by Filter:** To configure the filter conditions:

- Click . A row is added.
- Configure the following:
 - **Clause:** Select the clause based on the requirement (Having, and, or).
 - **Aggregate:** Select the aggregate function (sum, count, min) for an expression.
 - **Expression(1):** The selected table's columns are available in the list. Select the first expression for the condition.
 - **Operator:** Select the operator (=, <=, >=, >, <) required to define your expression.
 - **Aggregate:** Select the aggregate function (sum, count, min) for the second expression.
 - **Expression(2):** The selected table's columns are available in the list. Select the second expression for the condition.

d. Click **Save and Continue**.

Configured Example: The below example has the following conditions configured: where Subscriber ID = CDR ID and Subscriber Phone Number = CDR Phone Number Having (min CDR value > min Subscriber value)

Tip:

The above example has the following conditions configured:

- where Subscriber ID = CDR ID and Subscriber Phone Number = CDR Phone Number
- Having (min CDR value > min Subscriber value);

10. **Preview:** Click to preview the configured query measure. Test the query. After the query is run, you can view them in the output.

The Preview tab has the following configuration expandable:

- a. **Query:** Click this expandable menu. A query is displayed in the expanded view. Click **Test Query** to test the configured query. After the query has run successfully, you are notified with the successful message in the grid.
- b. **Output Table:** Click the expandable to view the output data of executed query.

11. **Reporting:** This tab allows you to configure the table details to store the resultant data. Configure the following:

- a. **Enable Report Table:** Click and enable the **Enable Report Table Toggle** switch.
- b. **Reporting Table:** Enter the reporting table name, where the output data must be stored. If the name is not specified and left blank, then the system generates the table name automatically.
- c. **Truncate before Load:** If selected, will not allow you to load duplicate data.
- d. **Specify Tags:** It allows you to segregate the table in the Data Catalog. Click on the **Type to add new** and then manually create the required tag or select the available tag from the drop down list. For more, refer to the [Data Catalog](#).
- e. Click Done.

12. Configure the details in each tab and click **Save and Continue**.

13. Click **Apply**.

14. **Preview :** It allows you to view the parent table of a column. On the canvas, use single-click to **select a Query Measure operator, and click Results Preview at the bottom of the canvas**. Table records details will be displayed.

- a. **To view the parent table:** Single-click on the required column, the row gets selected.
- b. Now, right-click on the row to open menu and go to **Drilldown** and click on the displayed parent table name.
- c. Parent Table is displayed.
- d. To learn more about Result preview, refer to the [Result Preview](#).

- e. To learn how to schedule and run, refer to the topic [Schedule Pipeline](#) and [Run pipeline](#).

Use of Absolute / Relative Date in Expression(2)

You can now use Absolute / Relative Date in Expression(2) to filter out the records. Follow below steps to use them:

1. In **Expression (1)** if Date or Time stamp column is selected to filter records, in **Expression(2) Absolute / Relative Date** option is populated in dropdown list. This option allows you to select customize date or timestamp to filter the record.
2. When option is selected, **Select Date** window is displayed. Click on toggle switch to select either Absolute Date or Relative Date.
3. To use **Absolute Date**, Select date from Calendar > Click **Apply**.
4. To use **Relative Date**, Click on toggle switch. Complete below parameters and Click **Apply**.
 - a. **Relative Value:** Enter the relative value to compare.
 - b. **Duration:** Select the duration from the dropdown list. Available options are Minutes Before, Hours Before, Days Before, Weeks Before, Months Before.
5. Example:

Statement Functions

Below are the list of the statement functions that you can use in Select Column Tab.

| Sl. No | Function | Description | Format | Example |
|--------|--------------------------|--|---|---|
| 1 | cast() / tointeger() | it will convert data type from string to integer and return the integer value. In Batch Filter , use this function as tointeger(). | ~ cast(Arg1 as int) (or) ~ tointeger(Arg1) | Example: cast('900' as int) Explanation: Here data type for 900 is string. Data type is first converted to integer. And 900 is stored as integer value. Output: 900. |
| 2 | char_length() / length() | This function will calculate the length of the string. It will count white space, special characters . | ~ char_length(Arg1) (or) ~ length(Arg1) where, Arg1 is String | Example 1: char_length('Mumbai') Output: 6 Example 2: length('India') Output: 5 |

| | | | | |
|---|----------------------------|---|--|--|
| | | <p>In case of numeric value, Date or Time-stamp, column type should be varchar or char, else function will return 0.</p> <p>In Batch Filter, use this function as length().</p> | | |
| 3 | coalesce() / tonvl() | <p>This function will return first non-null value. In case all the arguments are null, then it will consider only last argument.</p> <p>In Batch Filter, use this function as tonvl().</p> | <p>~ coalesce(Arg1,Arg2,Arg3,...) (or) ~ tonvl(arg1,arg2)</p> | <p>Example 1: coalesce(null,null,'John') Output: John</p> <p>Example 2: tonvl(null,'Marie') Output: Marie</p> <p>Example 3: tonvl(a,b) Output: a [Here both a and b are non null value].</p> |
| 4 | concat() | <p>This function will join 2 or more strings. You can use this function to join 2 or more array.</p> | ~ concat(Arg1, Arg2, Arg3,...) | <p>Example 1: concat('We',' are',' human') Output: We are human</p> <p>Example 2: concat(array(1, 2, 3), array(4, 5), array(6)) Output: [1,2,3,4,5,6]</p> |
| 5 | date_trunc() / datetrunc() | <p>This function will truncate the timestamp as per the specified format.</p> <p>In Batch Filter, use this function as datetrunc().</p> | <p>~ date_trunc(String Format, Column Timestamp) where, Format can be either year, month, date, hour, minute, seconds or any specific combination. example: date_trunc('yyyy'/'mm'/dd'/'HH'/'ss'/'MM', dd-mm-yyyy hh:mm:ss)</p> | <p>Example 1: date_trunc('YEAR', '2015-03-05T09:32:05.359') Output: 2015-01-01 00:00:00</p> <p>Example 2: date_trunc('MM', '2015-03-05T09:32:05.359') Output: 2015-03-01 00:00:00</p> <p>Example 3: date_trunc('DD', '2015-03-05T09:32:05.359')</p> |

| | | | | |
|---|-------------------------------|---|--|--|
| | | | | <p>Output: 2015-03-05 00:00:00</p> <p>Example 4: date_trunc('HOUR', '2015-03-05T09:32:05.359')</p> <p>Output: 2015-03-05 09:00:00</p> <p>Example 5: date_trunc('MILLISECOND', '2015-03-05T09:32:05.123456')</p> <p>Output: 2015-03-05 09:32:05.123</p> |
| 6 | datediff() / datedifference() | <p>This function will provide date difference between 2 timestamp/ date.</p> <p>In Batch Filter, use this function as datedifference().</p> | ~ datediff(timestamp end, timestamp start) | <p>Example 1: datediff("2018-01-10 00:00:00", "2018-01-08 23:59:59")</p> <p>Output 1:2</p> <p>Example 2: datediff("2018-01-08 00:00:00", "2018-01-10 23:59:59")</p> <p>Output 2: -2</p> |
| 7 | dayofmonth() | <p>This function will provide the day of the month for the given date or timestamp.</p> | <p>~ dayofmonth(timestamp)</p> <p>(or)</p> <p>~ dayofmonth(date)</p> | <p>Example: dayof-month('2009-07-30')</p> <p>Output: 30</p> |
| 8 | dayofweek() | <p>This function will provide the day of the week for the given date or timestamp.</p> | <p>~ dayofweek(timestamp)</p> <p>(or)</p> <p>~ dayofweek(date)</p> | <p>Example: dayofweek('2009-07-30')</p> <p>Output: 5 [30th July 2009 is Thursday. Considering Sunday as start of week.]</p> |
| 9 | dayofyear() | <p>This function will provide the day number of the year for the given date or timestamp.</p> | <p>~ dayofyear(timestamp)</p> <p>(or)</p> <p>~ dayofyear(date)</p> | <p>Example: dayofyear('2016-04-09')</p> <p>Output: 100. [Here, 9th April is 100th day of year 2016]</p> |

| | | | | |
|----|-------------------------|--|--|---|
| 10 | hour() / hourofday() | <p>This function will provide the hour of the given time-stamp.</p> <p>In Batch Filter, use this function as hourofday().</p> | <p>~ hour(timestamp) (or) ~ hourofday(timestamp)</p> | <p>Example: hour('2023-07-30 12:58:23')</p> <p>Output: 12</p> |
| 11 | instring() | <p>This function will search for the first occurrence of the substring within a given string and return the index position of occurrence. It will count white spaces and special characters.</p> <p>If Substring is not found return value will be zero.</p> | <p>~ instring(Arg1, Arg2) where, Arg1 - String Arg2 - Substring to be found</p> | <p>Example 1: instr('Hello World', 'World')</p> <p>Explanation: Index position for above string as follows, H:1, E:2, and so on. Arg2 (World) is searched in Arg1 and its index position is returned as 7.</p> <p>Output: 7 [as World is found in Arg1 and index position of occurrence for 'W' in string is at 7th place].</p> <p>Example 2: instr('Hello USA Citizen', America)</p> <p>Explanation: Index position for above string as follows, H:1, E:2, and so on. Position for N is 17. Arg2 (America) is searched in Arg1 and its index position is returned as 0.</p> <p>Output: 0 [as America is not found in the Agr1].</p> |
| 12 | lower() | <p>This function converts string into lower case.</p> | <p>~ lower(Arg1) where, Arg1 is String</p> | <p>Example: lower('John MATHew')</p> <p>Output: john mathew</p> |
| 13 | nestedfunctions() | <p>This function allows you to pass function as argument within a function.</p> | <p>~ function1(function2(function3(Arg2)))</p> | <p>Example: substring('Kangaroo', 1,instr('lower('INDIA')','d'))</p> <p>Explanation: Here, there are 3 functions: substring(), instr(), and lower(). Output of function lower() act as argument to in-</p> |

| | | | | |
|----|------------------------------------|--|---|--|
| | | <p>In this case, the innermost function is executed first and act as argument to nested function.</p> <p>The outermost function is executed last.</p> | | <p>str(). Output of instr() act as Example to substring(). For function lower('INDIA'), output is india. For function instr('india',d), output is 3. For function substring('Kangaroo',1,3), output is Kan.</p> <p>Output: Kan</p> |
| 14 | round() | <p>This function round of the first argument to the nearest value as per second argument.</p> | ~ round(Arg1, Arg2) | <p>Example: round(2.537,2)</p> <p>Output: 2.54</p> |
| 15 | SubString() | <p>This function will extract substring from the given string as per start and end index.</p> | <p>~ substring(Arg1,Arg2,Arg3) where, Arg1 - String Arg2 - Starting Index in String Arg3 - Ending Index in String</p> | <p>Example: substring('Australia', 2,6)</p> <p>Explanation: Index position for above string as follows A:1, U:2, and so on.</p> <p>Output: ustra</p> |
| 16 | timediff() / timestampdifference() | <p>This function will provide time difference between 2 timestamp in millisecond.</p> <p>In Batch Filter, use this function as timestampdifference().</p> | ~ timediff (timestamp end, timestamp start) | <p>Example: timediff("2018-01-10 05:06:23", "2018-01-10 07:06:23")</p> <p>Output: 7200</p> |
| 17 | to_date() / to-date() | <p>This function will convert the column into date type.</p> <p>In Batch Filter, use this func-</p> | <p>~ to_date(Column A, String Format) where, Format can be either year, month, date, hour, minute, seconds or any specific combination.</p> <p>example: to_date(dd-mm-yyyy hh:mm:ss, 'yyyy-mm-dd')</p> | <p>Example 1: to_date(10-12-2023 05:10:20, 'dd-mm-yyyy')</p> <p>Output: 10-12-2023</p> <p>Example 2: todate(10-12-2023 05:10:20, 'mm/dd/yyyy')</p> |

| | | | | |
|----|-------------------|--|---|---|
| | | tion as to-date(). | | Output: 12/10/2023 |
| 18 | tochar() / cast() | This function will convert date or time-stamp into string. In Query Measure(QM) and SQL Measure(SQLM) , use this function as cast(). | ~ tochar(timestamp) (or) ~ cast(arg1 as string) | Example: tochar(2018-01-10 05:26:43) Output: "2018-01-10 05:26:43" |
| 19 | upper() | This function converts string into upper case. | ~ upper(Arg1) where, Arg1 is String | Example: upper(' John MAtthew') Output: JOHN MATHEW |

KPI

Last updated on September 19, 2024

Prerequisite: You must configure either the QM or SQL, or both, in the pipeline. Otherwise, the opportunity to create and view KPI will be disabled.

Query Measures and SQL Measure provide data to Key Performance Indicators (KPIs). Key Performance Indicators (KPI) provides easily configurable range or threshold settings for Measures. This allows each customer to decide appropriate normal and alert levels for their data processing. When a measure violates the thresholds set by the KPI, an alert is generated. The results of an individual KPI on a query measure can either be a Passed or Violated value. When a measure violates a KPI, system generates an alert. A query measure with KPI configured are identified with the link icon available in the component grid.

To configure KPI for the query measure:

1. For the configured query measure, click vertical ellipses.
2. Click **Create/View KPI**.
3. **Configure:** KPI window is displayed.

It has the KPI and Create/Edit New KPI tabs.

4. **KPI:** Click KPI tab. This tab lists you the KPI's that are already created. For more information, see KPI tab.

This tab has the list of KPIs created. For the KPI listed, you can perform the following actions:

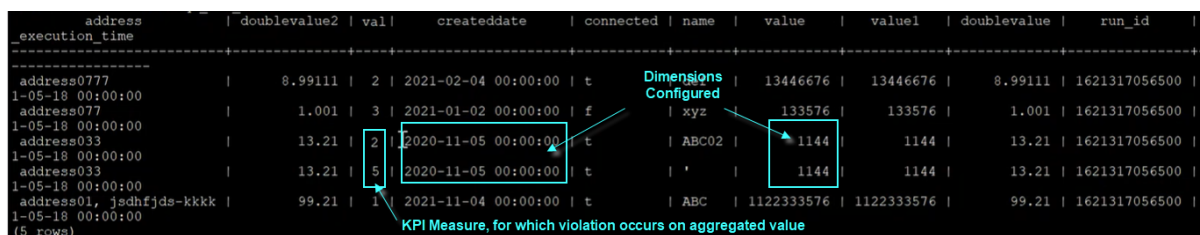
- **Edit:** Click icon to edit the KPI. The **Create/Edit KPI** tab is displayed.
- **Delete:** Click **vertical ellipses** associated with the KPI > click **Delete**. KPI will be deleted.
- **Copy:** Click icon to copy the KPI. A copy of it is created in the same tab.

5. **Create/Edit New KPI:** This tab allows you to create new KPIs.

Click and configure the following fields:

6. Configure the following:

- **KPI Name:** Enter the name for the KPI definition.
- **Dimension:** The output columns of query measure are available as dimension columns. Select the columns that impact the KPI. For the configured KPI measure and selected dimensions, if there are multiple occurrences of selected dimension (columns) in the input data, KPI is violated one time based on aggregation of the configured measure.
- In other words, KPI is violated only for distinct values of input data; if same values then KPI is violated for aggregated value of respective KPI measure.
- **Example:** Consider we have selected **created date** and **value** as dimension. If the KPI measure is **val**, **Minimum** value configured is **3** and **Maximum** value configured is **4**. KPI is violated for the values <3 and >4. If dimensions are same, KPI is violated one time for aggregated values (in this example, 4 and 5). The below image gives you the clear picture on the same.



| address execution_time | doublevalue2 | val | createddate | connected | name | value | value1 | doublevalue | run_id |
|--|--------------|-----|---------------------|-----------|-------|------------|------------|-------------|---------------|
| address0777 1-05-18 00:00:00 | 8.99111 | 2 | 2021-02-04 00:00:00 | t | | 13446676 | 13446676 | 8.99111 | 1621317056500 |
| address077 1-05-18 00:00:00 | 1.001 | 3 | 2021-01-02 00:00:00 | f | xyz | 133576 | 133576 | 1.001 | 1621317056500 |
| address033 1-05-18 00:00:00 | 13.21 | 2 | 2020-11-05 00:00:00 | t | ABC02 | 1144 | 1144 | 13.21 | 1621317056500 |
| address033 1-05-18 00:00:00 | 13.21 | 5 | 2020-11-05 00:00:00 | t | ' | 1144 | 1144 | 13.21 | 1621317056500 |
| address01, jsdhfjds-kkkk 1-05-18 00:00:00 | 99.21 | 1 | 2021-11-04 00:00:00 | t | ABC | 1122333576 | 1122333576 | 99.21 | 1621317056500 |

(5 rows)

- **Case Template:** Select the **Case template** that you want to attach to this KPI definition. When there is any KPI violation based on this KPI definition, a case will be generated based on the selected case template.
- **KPI Measure:** Select the measure, on which KPI is validated. The columns with the numeric datatypes like decimal, double, integer, long/big data type are available in this field for selection.
- **Minimum:** Select the minimum value of the KPI threshold for the selected measure/ parameter.
- **Maximum:** Select the maximum value of the KPI threshold for the selected measure/ parameter.

Note:

Minimum and Maximum support decimal places. To know how to set the decimal place, see [Decimal Settings](#).

- **Grace Time:** Enter the grace period for KPI to look back and validate the violated instances to be considered or ignored.

| Run | Dimension | Measure | Violation | Threshold | Case | Grace time | Execution time |
|-----|-----------|---------|-----------|-----------|------|------------|------------------------|
| 1 | 1234 | 2 | FALSE | 5 | No | | 6---6/5/2021 |
| | 999393 | 1 | FALSE | | No | | 6---6/5/2022 |
| | 788883 | 6 | TRUE | | No | | 6---6/5/2023 |
| | 99992 | 12 | TRUE | | No | | 6---6/5/2024 |
| | | | | | | | 6---6/5/2025 |
| 2 | 1234 | 2 | FALSE | | | 1hr | 6:30---6/5/2025 |
| | 999393 | 12 | TRUE | | No | | 6:30---6/5/2026 |
| | 788883 | 33 | TRUE | | Yes | | 6:30---6/5/2027 |
| | 99992 | 1.5 | FALSE | | | | 6:30---6/5/2028 |
| | 2222111 | 23 | TRUE | | No | | 6:30---6/5/2029 |
| | 8588833 | 4 | FALSE | | | | 6:30---6/5/2030 |
| 3 | 1234 | 2 | FALSE | | | | 10:30:00 PM---6/5/2030 |
| | 999393 | 12 | TRUE | | No | | 10:30:00 PM---6/5/2030 |
| | 788883 | 33 | TRUE | | Yes | | 10:30:00 PM---6/5/2031 |
| | 99992 | 1.5 | FALSE | | | | 10:30:00 PM---6/5/2032 |
| | 2222111 | 23 | TRUE | | No | | 10:30:00 PM---6/5/2033 |
| | 8588833 | 4 | FALSE | | | | 10:30:00 PM---6/5/2034 |

- **Frequency:** Select the **frequency** of configured grace time if it is minutes, hours, days, or so on.
- **Generate only violated rows:** If selected, the system generates only violated rows.
- **Custom Cost Computation:** The KPI measure configured is available in expression pane. Configure the expression to be considered to execute with the KPI measure. For more information on options, see [Expression Options](#). The expression configured, will be executed, and the sum of all the values are is considered to be leakage. You can view the same in KPI workspace.
- Custom Cost Computation - **Expression Options**

Note:

Only arithmetic operations are supported.

Expression builder has the following options to configure:

- : Click to syntactically validate the expression.
- : Click to indent the expression.
- : Click to erase the entered expression.

7. Click **Save KPI**. After configuring KPI.

8. To know about **Case Operator Sink**, go back to [Workflow configuration](#) page.
9. To learn more about Run Pipeline, refer to [Schedule & Run](#) from Configure a Pipeline.

SQL Measure

Last updated on September 29, 2023

SQL measures are used to extract specific metrics from the different control points. Example: Use of SQL measures to calculate the number of records and duration distribution for each call types for traffic from switch. They are also used to refine or filter output of other measures. SQL measure allows you to type a query and preview the tables, which helps you to create queries in an intuitive way. A SQL Measure fetches data from a table instance of Data Catalog, that allows you to segregate data based on the fields selected from the reference data. It supports (RDBMS - postgresSQL, Minio) in AWS/Azure and GCS (hive metastore) in Google cloud platform.

Note:



SQL Measure can only be linked and configured with Query Measure transformation operator.

A KPI can be created for any SQL measure. For more information, see [KPI](#). To configure a SQL measure:

1. In the Canvas, go to **Operators grid > Transform Operators > SQL Measure**.
2. Drag and drop the **SQL Measure operator** to Canvas > Configuration pane.
3. Click the vertical ellipses > **Edit**. The **Configure SQL Measure** window is displayed.
4. Click **Query Details** expandable.

Configure the following:

- **Query Name:** Enter the query name. **Example:** Test_SQM.
 - **Target Store:** Allows you to configure the target database type. The output data will be stored in the selected database.
 - Click **Next**.
5. Configure SQL Measure window has the following tabs:
 - **Select Database:** This tab has the list of databases available. It allows you to select and add the required database, on which you want to execute the SQL measure. To add the data tables for a SQL measure:

- Go to **Select Database** tab> Hover the mouse on the databases. A plus icon appears, click . You can also use the field **Search Database** to search by entering the database name.
- If a database has  associated with it, it indicates the database is not selected for SQL measure. If a database has  associated with it, then it indicates the database is selected for SQL measure. If you try to select more than one database, the previous selected database is deselected.
- **SQL Query** tab: it allows you to type query and validate. You can use statement functions to write in the SQL Query. For more information, refer to [Statement Function](#).
- Start typing the query and it will suggest the available tables for the selected database.

Note:

From the **Timestamp** to the **Date** data type conversion can be handled for the required data column.
- Spark Validation toggle switch: It allows you to validate all the spark SQL queries for which trino does not support. The validation will fail because some functions are supported by spark SQL but not by trino, so you can validate directly in spark SQL before running the SQL Measure. Enable the Spark Validation toggle switch to validate the query through spark functionality.
- After that click **validate** to verify the query syntax and the table availability. In case of valid query, a success message "**Valid**" will be displayed at the bottom of the SQL query text box.
- You can perform several actions such as Redo , Undo , Copy , Clear Query on this screen.
- **Result** tab: It displays the table details. After running the query, the results will be displayed in this tab.
- **Reporting:** This tab is an optional tab which allows you to configure the table detail to store the resultant data. Configure the following:
 - **Enable Report Table:** Click and enable the **Enable Report Table Toggle** switch.
 - **Reporting Table:** Enter the reporting table name, where the output data must be stored. If the name is not specified and left blank, then the system generates the table name automatically.

- **Truncate before Load:** If selected, it will not allow to append new data.
 - **Specify Tags:** It allows you to segregate the table in the Data Catalog. Click on the **Type to add new** and then manually create the required tag or select the available tag from the drop down list. Refer to the [Data Catalog](#).
6. Configure the details in each tab and click **Done**.
 7. Once the pipeline is configured you can preview the data, which allows you to view the parent **table of a column**. On the canvas, use **single-click**. To learn more, refer to [Result Preview Features](#).
 8. Click **Save**. Go back to the Configuration page to learn about [Schedule and Run](#) pipeline.

Statement Functions

Below are the list of the statement functions that you can use write in SQL Query.

| Sl. No | Function | Description | Format | Example |
|--------|--------------------------|--|---|---|
| 1 | cast() / tointeger() | it will convert data type from string to integer and return the integer value . In Batch Filter , use this function as tointeger() . | ~ cast(Arg1 as int) (or) ~ tointeger(Arg1) | Example: cast('900' as int) Explanation: Here data type for 900 is string. Data type is first converted to integer. And 900 is stored as integer value. Output: 900. |
| 2 | char_length() / length() | This function will calculate the length of the string. It will count white space, special characters . In case of numeric value, Date or Time-stamp , column type should be varchar or | ~ char_length(Arg1) (or) ~ length(Arg1) where, Arg1 is String | Example 1: char_length('Mumbai') Output: 6 Example 2: length('India') Output: 5 |

| | | | | |
|---|----------------------------|---|--|---|
| | | <p>char, else function will return 0.</p> <p>In Batch Filter, use this function as length().</p> | | |
| 3 | coalesce() / tonvl() | <p>This function will return first non-null value. In case all the arguments are null, then it will consider only last argument.</p> <p>In Batch Filter, use this function as tonvl().</p> | <p>~ coalesce(Arg1,Arg2,Arg3,...) (or) ~ tonvl(arg1,arg2)</p> | <p>Example 1: coalesce(null,null,'John') Output: John</p> <p>Example 2: tonvl(null,'Marie') Output: Marie</p> <p>Example 3: tonvl(a,b) Output: a [Here both a and b are non null value].</p> |
| 4 | concat() | <p>This function will join 2 or more strings. You can use this function to join 2 or more array.</p> | ~ concat(Arg1, Arg2, Arg3,...) | <p>Example 1: concat('We',' are',' human') Output: We are human</p> <p>Example 2: concat(array(1, 2, 3), array(4, 5), array(6)) Output: [1,2,3,4,5,6]</p> |
| 5 | date_trunc() / datetrunc() | <p>This function will truncate the timestamp as per the specified format.</p> <p>In Batch Filter, use this function as datetrunc().</p> | <p>~ date_trunc(String Format, Column Timestamp) where, Format can be either year, month, date, hour, minute, seconds or any specific combination. example: date_trunc('yyyy'/'mm'/dd'/'HH'/'ss'/'MM', dd-mm-yyyy hh:mm:ss)</p> | <p>Example 1: date_trunc('YEAR', '2015-03-05T09:32:05.359') Output: 2015-01-01 00:00:00</p> <p>Example 2: date_trunc('MM', '2015-03-05T09:32:05.359') Output: 2015-03-01 00:00:00</p> <p>Example 3: date_trunc('DD', '2015-03-05T09:32:05.359') Output: 2015-03-05 00:00:00</p> <p>Example 4: date_trunc('HOUR', '2015-03-05T09:32:05.359') Output: 2015-03-05 09:00:00</p> |

| | | | | |
|----|-------------------------------|---|---|--|
| | | | | <p>Example</p> <p>5: date_trunc('MILLISECOND', '2015-03-05T09:32:05.123456')</p> <p>Output: 2015-03-05 09:32:05.123</p> |
| 6 | datediff() / datedifference() | <p>This function will provide date difference between 2 timestamp/ date.</p> <p>In Batch Filter, use this function as datedifference().</p> | ~ datediff(timestamp end, timestamp start) | <p>Example 1: datediff("2018-01-10 00:00:00", "2018-01-08 23:59:59")</p> <p>Output 1: 2</p> <p>Example 2: datediff("2018-01-08 00:00:00", "2018-01-10 23:59:59")</p> <p>Output 2: -2</p> |
| 7 | dayofmonth() | <p>This function will provide the day of the month for the given date or timestamp.</p> | ~ dayofmonth(timestamp) (or) ~ dayofmonth(date) | <p>Example: dayofmonth('2009-07-30')</p> <p>Output: 30</p> |
| 8 | dayofweek() | <p>This function will provide the day of the week for the given date or timestamp.</p> | ~ dayofweek(timestamp) (or) ~ dayofweek(date) | <p>Example: dayofweek('2009-07-30')</p> <p>Output: 5 [30th July 2009 is Thursday. Considering Sunday as start of week.]</p> |
| 9 | dayofyear() | <p>This function will provide the day number of the year for the given date or timestamp.</p> | ~ dayofyear(timestamp) (or) ~ dayofyear(date) | <p>Example: dayofyear('2016-04-09')</p> <p>Output: 100. [Here, 9th April is 100th day of year 2016]</p> |
| 10 | hour() / hourofday() | <p>This function will provide the hour of the given timestamp.</p> | ~ hour(timestamp) (or) ~ hourofday(timestamp) | <p>Example: hour('2023-07-30 12:58:23')</p> <p>Output: 12</p> |

| | | | | |
|----|--------------------------------|--|---|---|
| | | In Batch Filter , use this function as hourof-day() . | | |
| 11 | <code>instring()</code> | <p>This function will search for the first occurrence of the substring within a given string and return the index position of occurrence. It will count white spaces and special characters.</p> <p>If Substring is not found return value will be zero.</p> | <p>~ <code>instring(Arg1, Arg2)</code> where, Arg1 - String Arg2 - Substring to be found</p> | <p>Example 1: <code>instr('Hello World', 'World')</code></p> <p>Explanation: Index position for above string as follows, H:1, E:2, and so on. Arg2 (World) is searched in Arg1 and its index position is returned as 7.</p> <p>Output: 7 [as World is found in Arg1 and index position of occurrence for 'W' in string is at 7th place].</p> <p>Example 2: <code>instr('Hello USA Citizen', America)</code></p> <p>Explanation: Index position for above string as follows, H:1, E:2, and so on. Position for N is 17. Arg2 (America) is searched in Arg1 and its index position is returned as 0.</p> <p>Output: 0 [as America is not found in the Agr1].</p> |
| 12 | <code>lower()</code> | <p>This function converts string into lower case.</p> | <p>~ <code>lower(Arg1)</code> where, Arg1 is String</p> | <p>Example: <code>lower('John MATHew')</code></p> <p>Output: john mathew</p> |
| 13 | <code>nestedfunctions()</code> | <p>This function allows you to pass function as argument within a function.</p> <p>In this case, the innermost function is executed first and act as argument to</p> | <p>~ <code>function1(function2(function3(Arg2)))</code></p> | <p>Example: <code>substring('Kangaroo', 1, instr('lower('INDIA')', 'd'))</code></p> <p>Explanation: Here, there are 3 functions: <code>substring()</code>, <code>instr()</code>, and <code>lower()</code>. Output of function <code>lower()</code> act as argument to <code>instr()</code>. Output of <code>instr()</code> act as Example to <code>substring()</code>. For function <code>lower('INDIA')</code>, output is india. For function <code>instr('india', d)</code>, output is 3. For function <code>substring('Kangaroo', 1, 3)</code>, output is Kan.</p> |

| | | | | |
|----|------------------------------------|--|---|--|
| | | <p>nested function.</p> <p>The outermost function is executed last.</p> | | <p>Output: Kan</p> |
| 14 | round() | <p>This function round of the first argument to the nearest value as per second argument.</p> | <p>~ round(Arg1, Arg2)</p> | <p>Example: round(2.537,2)</p> <p>Output: 2.54</p> |
| 15 | SubString() | <p>This function will extract substring from the given string as per start and end index.</p> | <p>~ substring(Arg1,Arg2,Arg3) where, Arg1 - String Arg2 - Starting Index in String Arg3 - Ending Index in String</p> | <p>Example: substring('Australia', 2,6)</p> <p>Explanation: Index position for above string as follows A:1, U:2, and so on.</p> <p>Output: ustra</p> |
| 16 | timediff() / timestampdifference() | <p>This function will provide time difference between 2 timestamp in millisecond.</p> <p>In Batch Filter, use this function as timestampdifference().</p> | <p>~ timediff (timestamp end, timestamp start)</p> | <p>Example: timediff("2018-01-10 05:06:23", "2018-01-10 07:06:23")</p> <p>Output: 7200</p> |
| 17 | to_date() / to-date() | <p>This function will convert the column into date type.</p> <p>In Batch Filter, use this function as to-date().</p> | <p>~ to_date(Column A, String Format) where, Format can be either year, month, date, hour, minute, seconds or any specific combination.</p> <p>example: to_date(dd-mm-yyyy hh:mm:ss, 'yyyy-mm-dd')</p> | <p>Example 1: to_date(10-12-2023 05:10:20, 'dd-mm-yyyy')</p> <p>Output: 10-12-2023</p> <p>Example 2: todate(10-12-2023 05:10:20, 'mm/dd/yyyy')</p> <p>Output: 12/10/2023</p> |
| 18 | tochar() / cast() | <p>This function will convert date or time-</p> | <p>~ tochar(timestamp) (or) ~ cast(arg1 as string)</p> | <p>Example: tochar(2018-01-10 05:26:43)</p> <p>Output: "2018-01-10 05:26:43"</p> |

| | | | | |
|----|---------|---|--|---|
| | | stamp into string. In Query Measure(QM) and SQL Meaure(SQLM), use this function as cast(). | | |
| 19 | upper() | This function converts string into upper case. | ~ upper(Arg1) where, Arg1 is String | Example: upper('John MAtthew') Output: JOHN MATHEW |



Data Match Measure

Last updated on December 01, 2023

The process of comparing and matching two different sets of data is known as data matching. The procedure's goal is to locate data that all pertain to the same entity. Often, the data comes from two or more independent collections of data with no common identities. However, data matching may also be used to find duplicate data in a database.

To configure a Data Match measure:

1. In the Canvas, go to **Operators grid > Transform Operators > Data Match Measure**.
2. Drag and drop the **Data Match Measure** operator to Canvas > Configuration pane.
3. Click the vertical ellipses > **Edit**. The **Configure Data Match Measure** window is displayed.
4. Configure the following:
 - **Data Match Name:** Enter the name. **Example:** test_data_match.
 - **Select Target Store:** Allows you to configure the target database type. The output data will be stored in the selected database.
 - **Select the checkbox for the type of data match required.** You can select multiple types if required.
 - **Exact Match:** Displays result when two columns is exactly the same.
 - **Under Match:** Displays results when there is no data matching in either of the sources i.e record available in only one of the source
 - **Over Match:** Displays result if any of the following is true: Multiple Exact Match result between the two sources. Similar Data match in the same column or data source.

- Only Compare Discrepancy: Displays only the discrepancy result for two columns after comparison. Displays only the discrepancy result for two columns after comparison.
 - Click **Next**.
5. Configure Data Match Measure window has the following tabs:
- **Add Source:** This tab has the list of databases available. It allows you to select and add the required data tables, on which you want to execute the data match measure. To add the data tables:
 - Go to **Select Tables** pane and hover the mouse on the data table. A plus icon appears, click . You can also use the field **Search** to search by entering the data table name.
 - **Show Selected:** Use this toggle to view only the selected tables.
 - If a data table has  associated with it, it indicates it's not selected for data match measure. If a data table has  associated with it, then it indicates that it's selected for data match measure. If you try to select more than two data tables, a message is prompted, "You cannot select more than two tables".
 - **Data** tab: It displays the data for the selected table. You can perform *common actions* on this window.
 - **Column** tab: It displays all the columns for the selected table. You can also search by entering any keyword in the search text field.
 - **Hourly Timestamp Columns(s):** It allows to process the data as per the selected date and time stamp column. When Hourly Timestamp Column(s) is not selected, records are processed based on partition column.
 - **Filter:** This tab allows you to add an expression to filter the data. The entries in this panel are optional steps and can be bypassed, For more information follow the below steps:
 1. Filters tab has **Table** panels.
 2. The Filter names will be reflect based on the table name that you selected from the source.
 3. To configure the filter conditions follow below steps:
 - Click . A row is added.
 - Configure the following:
 - **Clause:** Select the clause (where, and, or) required for your conditional expression.
 - **Expression(1):** The selected table's columns are available in the list. Select the first expression for the condition.
 - **Operator:** Select the operator (=, <=, >=, >, <) required to define your expression.

- **Expression(2)**: The selected table's columns are available in the list. Select the second expression in the condition.

Note:

- You can use [Absolute / Relative Date](#) to filter out the record. To know more refer, [Use of Absolute / Relative Date in Expression \(2\)](#).

- **Delete**: Click , to delete the row if not required.

4. Similarly, in the second table filter section, follow the above steps to configure the filter expression.

The given expressions allows you to filter the data from the selected tables.

5. Click **Save and Continue**.

Configured Example:

Tip:

The above example has the following conditions configured:

- **Where TestTable1.int_ < 5 and where TestTable1.int_ >= 1**
- **Match Configuration**: It allows you to define the tables and columns that contributes to the output table. Left-hand side is the first column details, and on the right-hand side the second column on which the data match needs to be executed. For more information, see [Match Configurations](#).
- **Summary**: This tab displays the detailed result of the data match. For more information, see [Summary](#).
- **Reporting**: This tab is an optional tab which allows you to configure the table detail to store the resultant data. Configure the following:
 - **Enable Report Table**: Click and enable the **Enable Report Table Toggle** switch.
 - **Reporting Table**: Enter the reporting table name, where the output data must be stored. If the enable reporting toggle is enabled and the name is not specified and left blank, then the system generates the table name automatically.
 - **Truncate before Load**: If selected, it will not allow to append new data.
 - **Specify Tags**: It allows you to segregate the table in the Data Catalog. Click on the **Type to add new** and then manually create the required tag or select the available tag from the drop down list. For more, refer to the [Data Catalog](#).

6. Configure the details in each tab and click **Done**.

7. On the canvas page, single-click on the Measure operator, and click Results Preview at the bottom of the canvas. To learn more, refer to [Result Preview Features](#).

Results preview: In the below screen, the displayed table name was configured in [Reporting tab](#). If it was kept blank in the reporting tab, then a system generated table name will be displayed.

8. Go back to the Configuration page to learn about [Schedule and Run](#) pipeline.

Match Configurations

Configure the following details:

1. **Table name:** It displays the selected table name.
2. In the **Keys** details, configure the following:
 - **Column:** Select the column to be compared from the dropdown list.
 - **Output Column Name:** Specify the new column name.
 - **Tolerance:** An allowable amount of variation from the configured match configuration.
 - **Add Keys:** Click to add more key and it's details.
 - **Delete:** To delete a key, click on delete icon.
3. In the **Comparisons** details, configure the following:
 - **Column:** You need to select the column to be matched.
 - **Output Column Name:** Specify the new column name.
 - **Comparisons:** Select from the available options Compare Equal, List Equal, List subset, List Superset.
 - **Compare Equal:** To compare columns to see if the content is an exact match in terms of strings or integers. Example: Comparing two strings such as the Billing Line Equipment Number and the Switch Line Equipment number.
 - **List Equal:** To compare columns in measures that contain lists. Data in the list must be separated by delimiter. The delimiter can be any character, but has to be specified in the Delimiter field. Example: Comparing the list of features Subex Limited Confidential 1366 in the switch to the list of features being billed on a customer-by-customer basis.
 - **List Subset:** To ensure that column 1 is a sub-set of column 2. Example: Comparing a list of features available to subscribers being Billed to list of features provisioned on the Switch,

where the first is a subset of the second. In other words, subscribers are provisioned for more features than they are actually being billed for.

- **List Superset:** To ensure that column 1 is a super-set of column 2. Example: Comparing list of features available to subscribers in Switch to list of features being billed where the first is a superset of the second. In other words, subscribers are being billed for fewer features than they are billed for.
- **Delimiter:** Enter the delimiter separating each list item in the columns being compared.
- **Color:** This color is used to highlight any discrepancy in match.
- **Add Comparison:** Click to add more comparison criteria.
- **Delete:** To delete a comparison criteria, click on delete icon.

4. In the Display details, configure the following:

- **Column:** You need to select the column to be matched.
- **Output Column Name:** Specify the new column name.
- **Delete:** To delete a display criteria, click on delete icon.
- **Add Display:** To match additional column, select from the drop-down.

5. To learn more, go back to the next step [Summary](#).

Summary

Below columns are displayed on the summary screen:

- **mbm:** Stands for match bit mask. Allows you to identify overmatch(3), missing in source 1 (1) and missing in source 2(2))
- **overmatch_group_key:** A key to identify a bag of over-matched records.
- **Is_discrepancy:** Boolean (True or False) to identify if there is any discrepancy between the compare keys.
- **run_id:** It displays the Run identifier.
- **dm_id:** It displays the DM identifier.
- **dm_execution_time:** It displays the time of DM execution.
- To learn more, go back to the [Reporting](#) tab.

Use of Absolute / Relative Date in Expression(2)

You can now use Absolute / Relative Date in Expression(2) to filter out the records. Follow below steps to use them:

1. In **Expression (1)** if Date or Time stamp column is selected to filter records, in **Expression(2) Absolute / Relative Date** option is populated in dropdown list. This option allows you to select customize date or timestamp to filter the record.
2. When option is selected, **Select Date** window is displayed. Click on toggle switch to select either Absolute Date or Relative Date.
3. To use **Absolute Date**, Select date from Calendar > Click **Apply**.
4. To use **Relative Date**, Click on toggle switch. Complete below parameters and Click **Apply**.
 - a. **Relative Value:** Enter the relative value to compare.
 - b. **Duration:** Select the duration from the dropdown list. Available options are Minutes Before, Hours Before, Days Before, Weeks Before, Months Before.
5. Example:

Custom Operator

Last updated on May 12, 2023

A custom operator is a batch operator that allows user to write their own logic, to be performed on the selected data. User can write the logic/code in a interface project provided by BMS and a jar can be built out of it. This jar is now allowed to be uploaded into our server to be executed.

To configure a **Custom**:

1. In the Canvas, go to **Operators** grid > **Transform Operators** > **Custom**.
2. Drag and drop the **Custom** operator to Canvas > **Configuration** pane.
3. Click the vertical ellipses > **Edit**. The **Configure Custom** window is displayed.
4. Configure the following:
 - a. **Operator Name:** Enter the custom operator name.
 - b. **Select Template:** Allows you to **Select/Create the Template**.
 - i. To create new Template, refer to [Create Template](#).
 - ii. To create a copy of the existing configuration with New Template Name, refer to [Create Copy](#).
 - iii. upon selecting Template, the application displays the **Engine** tab, **Execution Parameters** and **Data Tables** sections.
 - c. **Engine:** The engine tab with default option of **Spark jobs**. spark java jars to be uploaded to be upload.
5. In the **Execution Parameters** section you can upload the self defined spark java jar.

Configure the **Execution Parameters** as below:

- a. To upload the file Click icon. The application navigates to the local system. This supports .jar file format only.
- b. Select the file and upload in the Execution Parameters section.
The Execution Parameters section displays:
 - i. **SL No**: Number of parameters added.
 - ii. **Parameter**: Displays the name of the parameters types.
 - iii. **File**: Displays the Upload icon to upload the file.
 - iv. **Type**: It displays the type by-default.
 - v. **Progress**: It displays the uploading process of the file.
 - vi. : Allows you to **Delete** the parameter(s).

Note:

You can delete row wise parameter by clicking on the particular delete option of the required row or You can delete all the parameters by clicking delete option from the top header.

- c. Click icon, to add the rows of **Boot**, **Dependency**, **JVM Params** and **Arguments** features, and purpose are as below:
 - i. **Boot**: It is the jar which contains the main code in it.
 - ii. **Dependency**: It is the supporting jars provided to help the main/boot jar.
 - iii. **JVM Params**: These are parameters that are passed to specify additional information to perform a spark submit.
 - iv. **Arguments**: These are the values that can be passed by the user so that he can use them in their code internals. i.e Inside the jar.
6. To configure the Data Tables, click the icon and select the **Input** and **Output** Data table. To select the data table, refer to the [Select Table](#).

The **Data Tables** section displays:

- a. **SL No**: Number of Data Tables.
- b. **Table Names**: Displays the name of the Tables.
- c. **Data Table**: Allows you to select the **Input** and **Output** Data Tables from the **Data Catalog**.
- d. **Type**: It displays the type of table.
- e. : Allows you to **Delete** the Data Table(s).

Note:

You can delete row wise Data Table by clicking on the particular delete option of the required row or You can Delete all the Data Table by clicking delete option from the top header.

- f. Click icon, to add multiple Input rows. Located top right corner of the Data Table.

7. After configuration, click Save.

Note :

- Once the **Custom** operator is configured with self defined spark java jar, go to the **Pipeline Repository** page > search the saved custom pipeline > click **Start** to run the Custom operator.
- The **Output** of the **Custom** will be displayed, based on the java code written in the file by the user.

Create Template

To create new template configure as below:

1. Click on **Select Template**. The configuration element is expanded.
2. Enter the category name in **Search Template**.
3. Click . A Template is added in the drop-down list.

Select Tables

This pane has the list of tables available in the Data Catalog. You can select the required tables, on which you want to execute the custom logic. It has the following options:

- **Search / Filter Fields:** Allows you to search the tables.
- **Show Selected:** The **Toggle Switch** upon enable allows you to view the selected tables.

To add the data tables for a custom:

1. Go to **Select Tables** pane > Hover the mouse on the tables instances. The plus icon is viewed. Click .
2. Click **Select**.

Note :

- If a table has associated with it, it indicates the table is not selected for query measure.
- If a table has associated with it, then it indicates the table is selected for query measure.

Create Copy

To create copy, follow the below process:

1. Click on the **Select Template** field. The configuration element is expanded.

2. In the **Search Template** field, select the existing template.

Note:

If you are selecting the existing template the template configuration will be frozen. the user cannot be able to do the any changes.

3. The application displays the existing template configuration on the window with **Create Copy** button.
4. Click **Create Copy**. The application displays **New Template Name** field.
5. Enter the new name of the template and click **Save**.

Note:

Upon Create Copy, all the configuration will be copied in the new template except the operator name.

Text Match

Last updated on January 19, 2024

This operator is used to restrict potential fraudulent subscribers return to the network. Text Match operator performs a pre-check of the new subscriber credentials against the existing database which is stored in elastic using indexing. Subscriber credentials such as name, address, IMSI, etc. are taken into account for validating the new subscriber. Configure the pre-check criteria match for fraudulent subscriber, if the match result is found then investigate such cases before activating the subscriber.

Text Match operator has input either from Data Reader or Transform Operator, and output is connected either to Transform Operator or Data Writer.

Steps to configure the Text Match Operator as follows:

1. Create a new or edit the existing **online/streaming pipeline**.
2. In Canvas, expand **Transform Operator**. Drag and drop **Text Match Operator**.
3. Provide the input connection to the **Text Match Operator**.
4. To configure, **Click Vertical Ellipses > Edit**. **Configure Text Match** window is displayed.
5. Complete the below details:
 - a. **Select Index List:** Index List is the reference source to check the incoming record. It is unique and super-set for all the incoming records. Select the index list from the dropdown list that you will use for comparing with source column. Maximum 5 indexes can be selected. To know how to create index, refer to [Indexing](#).
 - b. Click **Add Row** to add rows.
 - c. **Configure Check:** Configure the below details to perform check:
 - i. **Source Columns:** Source columns are columns coming from the incoming data source/record. Select the column from the dropdown list.

- ii. **Index List:** Displays index list that you will use as reference point for comparison. Select the Index list from the dropdown list in which you will compare the source column.
- iii. **Index Column:** Display all the columns present in the selected index list. Select the index column against which you will compare the source column.
- iv. **CheckBox:** You can select the check box to perform the text match operation. In case none of the check box is selected, exact match operation is performed. Below are the available check boxes for different text match operations:
 - **Exact:** If selected, source column and index column are compared for exact match. **For example:** If incoming data column has name Satish and also index column has Satish, then it is an exact match.
 - **Fuzzy:** If selected, source column and index column are compared for difference in one or two characters of whole word. **For example:** MacMillan is fuzzy match for Macmilan.
 - **Word Match:** If selected, source column and index column are compared for word with existing words. **For example:** Source column has name "David Williams" and Index column has name "John David Williams". In this case, match will pass.
 - **Substr:** If selected, source column and index column are matched for only part of data. Incoming data should be substring of index data. **For example:** Source column has name "Sachin" and Index column has email id "sachin@abccompany.com". In this case match will pass, as "Sachin" is subset of Index column.
 - **Phonetic:** If selected, source column and index column are matched for data with similar sounds. **For example:** "Ravindra", "Raveendra".
- v. **Weightage:** This parameter identifies the importance of the field. Higher the weightage, more important is the field. Enter the numeric value between 0 to 100.
- vi. **Match(%):** Minimum percentage to be matched while comparing columns.
- vii. **Exclude:** In this field you should mention the word which should be excluded while comparison of record. **For Example:** "Mr.", "Dr.", "Er."
- viii. **Cross Match:** This parameter is selected when you want to cross check the source column record with more index list column. If primarily selected index column does not match with incoming column, then it is checked with cross match columns. Select the index column from the dropdown list.
- d. **Configure Metrics:** Configure metrics define the condition to pass the records. Use slider to set the value. **Maximum allowed value is 100.** Metrics parameters are explained as below:
 - i. **Total Min Match:** Set Minimum percentage to be matched based on the weightage. **Formula to calculate total minimum match:** $[(\text{Field weightage} / \text{Total weightage}) * 100]$. **For example:** Weightage for column: name, address and job is 25, 35, and 15 respectively. Now suppose only name and job is matched with record. Total Minimum Match: $[(25+15)/75]*100$, result is 53.33. You set condition as 60, now match will fail. In output for match percentage column value will be 0. If threshold was 50 then the same column will have value 53.33.

- ii. **Exclude Word of Length:** This metrics allows you to set the word length to be exclude while comparing the columns. **For example:** Record has prefix "**Mr.**", "**Dr.**", "**Er.**". You set value as 3 to exclude them.
 - iii. **Restrict Participated Record:** This metrics allows you to restrict the similar entries. Here **individual incoming record is matched with all the index records. For example:** Suppose 100 index records are matched with 1 incoming/source record, then records are stored as per set restricted participated records. If restricted participated record is set as 20, then out of 100 records only 20 records will be displayed.
 - iv. **Minimum Matched Records:** This metrics allows you to set the minimum number of records to be matched to pass the condition. **For example:** There are 93 records. You set value as 63, then match is considered pass when minimum of 63 records are matched.
 - v. **Select Synonym Table:** This metric is selected when there is similar table is present in data catalog. If selected, below fields will be populated:
 - **Select Synonym Column:** Select the synonym column from dropdown list, if any.
 - **Select Abbreviation Column:** Select the abbreviation column from dropdown list, if any.
 - **For example:** If incoming column has "**ABC Pvt**" and you to change it as "**ABC Private**" then make use of synonym table that has synonym column and synonym abbreviation.
 - e. Click **Delete Icon** when you want to delete the row.
6. Click **Save** to keep the configuration.
7. Configuration example shown as below:

Data Writer

Data Writer

DSV Sink

[Last updated on May 12, 2023](#)

The output from any Transformation operator in the pipeline can be provided to DSV Sink. It allows you to store the output in local machine based on the configured directory.

To configure DSV Sink:

1. In the Canvas, go to **Operators grid > Data Writer**.
2. Drag and drop the DSV Sink into the canvas.
3. Connect the output of transformation operator as input to **DSV Sink**.

4. Click vertical ellipses of **DSV Sink** > click **Edit**. The **Configure DSV Sink** window is displayed.
5. Click the expandable configuration **General Details**.

Configure the following:

- **Data Writer Name:** Enter the name for DSV Sink.
 - **Description:** Allows you to enter the description for the DSV sink.
 - **Add/Search category:** Select the category to which the DSV Sink belong to. You can also create a new category to add the sinks. For more information on creating category, see [Create New Category](#).
 - **Directory:** Enter the directory where the file must be stored.
 - **Delimiter:** Enter the delimiter values. The data stored in output will have delimiter separated values. **Example:** if comma is configured, the output is stored with comma separated values.
 - **Is Header Present:** If enabled, the file without header is also processed.
6. Click **Next** or click expandable configuration **Configuration**.
 7. Configure the following parameter: **Compression Type:** Select the file compression type. That file that is being stored or collected will be compressed with the selected format.

8. Click **Save**.

Create New Category

To create a category in the DSV Sink configuration grid:

1. Click on **Add/Search Category** configuration element. The configuration element is expanded.
2. Enter the category name in **Add/Search** category.
3. Click **.** A category is added.

Database Sink

[Last updated on May 12, 2023](#)

The **Data Writer- Database** allows you to store the data in tabular format, in the connected database. The output of any transformation operator must be connected to the DB Sink and store the data.

To configure Database Sink:

1. In the Canvas, go to **Operators** grid > **Data Writer**.
2. Drag and drop the Database Sink into the canvas.
3. Connect the output of transformation operator as input to **Database Sink**.
4. Click vertical ellipses of **Database Sink** > click **Edit**. The **Configure Database Sink** window is displayed.
5. Click the expandable configuration **General Details**.

Configure the following:

- **Connection Name:** Enter the name for DB Sink.
- **Description:** Allows you to enter the description for the DB sink.
- **Select category:** Select the category to which the DB Sink belong to. You can also create a new category to add the sinks. For more information on creating category, see [Create New Category](#).
- **Tag:** Enter the tag for database connection, that helps in identifying the purpose of data writer(in future analysis).
- **Merge Strategy:** Select the merge strategy to store the data in the tables.
- **Append:** Appends the data to the existing rows in the data table.
- **TruncateAndLoad:** Truncates (deletes) the existing data and loads the new rows.
- **Upsert:** This operation allows you to compare the input data with existing data and insert a new record or update existing data into a table based on the Primary key.
 1. If the selected Primary key column value in the input data matches with the existing table column value, then it will update the data in the selected columns.
 2. If the selected Primary key column value in the input data is not matching with the existing table column value then it will insert the data. To perform **Upsert** function:
 - Select the **Upsert** option from the **Merge Strategy** dropdown list.
 - Enter the criteria in search field and click search. Then select Primary key column and Up-

date column checkbox.

Or

- Select required **Primary key column** and **Update column** checkbox manually .

- **Table Name:** Enter the output table name.

6. Click **Next** or click expandable configuration **Configuration**.

7. Configure the connection details to connect to the selected database type.

Example: If the selected database type is MySQL, then enter the connection details (**User Name**, **Password**, **IP/Host**, **Port**, **Database Name**) associated with MySQL database.

Supported database types are Oracle, MySQL and Postgres.

8. Click **Test Connection**. If the connection is successful, Click **Save**.

Create New Category

To create a category in the DSV Sink configuration grid:

1. Click on **Select Category** configuration element. The configuration element is expanded.
2. Enter the category name in **Select** category.
3. Click . A category is added.

Kafka Sink

[Last updated on May 12, 2023](#)

Kafka Sink is a data sink, which allows you to store the data in the form of kafka topics into the configured Kafka tables.

To configure Kafka Sink:

1. In the Canvas, go to **Operators** grid > **Data Writer**.
2. Drag and drop the **Kafka Sink** to **Canvas** > **Configuration** pane.
3. Connect the output of **Transform Operator** as input to **Kafka Sink**.
4. Click the **vertical ellipses** on **Kafka Sink** > **Edit**.

5. The **Configure Kafka Sink** window is displayed.

6. Click the expandable configuration **General Details**.

Configure the following properties:

- **Kafka Sink:** Enter the name of kafka sink.
- **Description:** Enter the brief description on the Kafka sink.
- **Add/Search category:** Select the category to which the S3 Sink belong to. You can also create a new category to add the sinks. For more information on creating category, see [Create New Category](#).
- **Tags:** Allows you to enter the tags that depicts the purpose of S3 sink in a pipeline.

7. Click **Next** or expandable configuration **Configuration**:

Configure the following:

- **URL:** Enter the Broker URL(with machine IP and port) for Kafka sink connection.
- **Topic:** Enter the kafka topic name.
- **Partition Strategy:** Indicates on the data partitions to be made while it is stored on to the kafka file system. You can also select **Round - Robin** or **Random** strategy type.
- **Compression Type:** Select the compression type to compress the files after the data is loaded to Kafka Consumer.
- **Acknowledgement:** Select the type of acknowledgement required.
 - 0 - No acknowledgement required.
 - 1 - Acknowledgement is required.
 - All - Acknowledgement is required with a detailed information in a message.
- **Additional Properties:** Allows you to configure the additional properties in the **key - value** pairs. For more information, see [Additional Properties configuration](#).

8. Click **Next** or **Data Format** expandable.


Configure the following:

- **Data Format:** Select the file format to be stored in Kafka Sink. CSV and JSON are supported.

9. Click **Save**.

Kafka - Additional Configurations

To configure the additional key - value pairs:

1. In **Additional Configurations** pane, click . A new row is added.

Configure the following:

- **Key:** Indicates how to deserialize the message keys to string.
- **Value:** Indicates how to deserialize the message values to string.

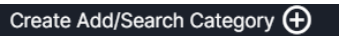
2. Click **Save**.

Note :

- The created Kafka Sink is available in the Operators > Data Writers grid > specific category for future usage.
- After all the configurations are done, save the Pipeline and run the Pipeline.

Create New Category

To create a category in the Kafka Sink configuration grid:

1. Click on **Add/Search Category** configuration element. The configuration element is expanded.
2. Enter the category name in **Add/Search** category.
3. Click . A category is added.

S3 Sink

[Last updated on May 12, 2023](#)

S3 Sink is an object storage suite capable of handling structured and unstructured data including log files, artifacts, backups, container images, photos and videos. You can either use Amazon S3 or MinIO as your S3 Sink storage.

To configure S3 sink:

1. In the Canvas, go to **Operators** grid > **Data Writer**.
2. Drag and drop the S3 Sink into the canvas.

3. Connect the output of transformation operator as input to **S3 Sink**.
4. Click vertical ellipses of **S3 Sink** > click **Edit**. The **Configure S3 Sink** window is displayed.
5. Click the expandable configuration **General Details**.

Configure the following:

- **Data Writer Name:** Enter the name for S3 Sink.
 - **Description:** Allows you to enter the description for the S3 sink.
 - **Add/Search category:** Select the category to which the S3 Sink belong to. You can also create a new category to add the sinks. For more information on creating category, see [Create New Category](#).
 - **Tags:** Enter the tags that depicts the purpose of S3 sink in a pipeline.
6. Click **Next** or click **Configuration** expandable.
 7. Configure the following:
 - **IP/Host:** Enter the IP or host details (URL) for S3, that you want to connect with. **Example:** `http://10.1yy.1xx.2:9aaa/` indicates the URL of S3 server with the machine ip: 10.1yy.1xx.2 and port: 9aaa.
 - **Access Key:** Enter the access key associated with the configured S3 server.
 - **Secret Key:** Enter the secret key associated with the configured S3 server.
 - **Directory:** Enter the directory in the S3 storage, to store the output data.
 - **Compression type:** **Example:** If the selected format is gzip, the output data is compressed and stored in gzip format.
 - **File Format:** Select the file format to store the output data in S3. **CSV** and **JSON** formats are supported. If you select the file format as **CSV**, then the below two additional fields are enabled.
 - **Delimiter:** Enter the delimiter to separate the values that are being stored in S3.
 - **Is Header Present:** If enabled, the file without header is also stored.
 8. Click **Save**.

Note :

- The created S3 Sink is available in the Operators > Data Writers grid > specific category for future usage.
- After all the configurations are done, save and run the Pipeline.

Create New Category

To create a category in the S3 Sink configuration grid:

1. Click on **Add/Search Category** configuration element. The configuration element is expanded.
2. Enter the category name in **Add/Search** category.
3. Click . A category is added.

GCS Sink

Last updated on May 12, 2023

GCS sink is a Google Cloud Storage sink with restfull online file storage web service for storing and accessing the data on Google Cloud Platform infrastructure.

To configure GCS sink:

1. In the Canvas, go to **Operators** grid > **Data Writer**.
2. Drag and drop the GCS Sink into the canvas.
3. Connect the output of transformation operator as input to **GCS Sink**.
4. Click vertical ellipses of **GCS Sink** > click **Edit**. The **Configure GCS Sink** window is displayed.
5. Click the expandable configuration **General Details**.

Configure the following:

- **Data Writer Name:** Enter the name for Sink.
- **Description:** Allows you to enter the description for the GCS sink.

- **Select category:** Select the category to which the GCS Sink belong to. You can also create a new category to add the sinks. For more information on creating category, see [Create New Category](#).
 - **Tags:** Enter the tags that depicts the purpose of GCS sink in a pipeline.
6. Click **Next** or click **Configuration** expandable.
7. Configure the following:
- **Upload File :** Allows you to upload a json file to authenticate the GCS connection.
 - **KeyJson:** Displays the preview of the JSON file.
 - **Directory:** Enter the directory in the GCS storage, to store the output data.
 - **Compression type:** Example: If the selected format is gzip, the output data is compressed and stored in gzip format.
 - **File Format:** Select the file format to store the output data in GCS . **CSV** and **JSON** formats are supported. If you select the file format as CSV, then the below two additional fields are enabled.
 - **Delimiter:** Enter the delimiter to separate the values that are being stored in GCS.
 - **Is Header Present:** If enabled, the file without header is also stored.
8. Click **Save**.

Note :

- The created GCS Sink is available in the Operators > Data Writers grid > specific category for future usage.
- After all the configurations are done, save and run the Pipeline.

Create New Category

To create a category in the GCS Sink configuration grid:

1. Click on **Select Category** configuration element. The configuration element is expanded.
2. Enter the category name in **Search Category**.
3. Click . A category is added.

S3 External - Table Sink

[Last updated on May 12, 2023](#)

S3 or Amazon storage service, which is a service offered by Amazon Web Services that provides **data storage in table** format through a web service interface.

To configure S3 External - Table Sink:

1. In the Canvas, go to **Operators** grid > **Data Writer**.
2. Drag and drop the **S3 External - Table Sink** into the canvas.
3. Connect the output of transformation operator as input to **S3 External - Table Sink**.
4. Click vertical ellipses of **S3 External - Table Sink** > click **Edit**. The **Configure S3 External - Table Sink** window is displayed.
5. Click the expandable configuration **General Details**.

Configure the following:

- **Data Writer Name:** Enter the name for Sink.
 - **Description:** Allows you to enter the description for the S3 External - Table sink.
 - **Select category:** Select the category to which the S3 External - Table Sink belong to. You can also create a new category to add the sinks. For more information on creating category, see [Create New Category](#).
 - **Tags:** Enter the tags that depicts the purpose of S3 External - Table sink in a pipeline.
6. Click **Next** or click **Configuration**.
 7. Configure the following:
 - **IP/Host:** Enter the IP or host address.
 - **Access Key:** Enter the access key.
 - **Secret Key:** Enter the secret key.
 - **Hive Thrift URL:** Enter the hive host link.
 - **Warehouse Directory:** Enter the warehouse directory path.
 - **Schema Name:** Enter the schema details.

- **Table Name:** Enter the table name.
 - **S3-Presto url:** Enter the link for the S3 Presto.
 - **S3-Presto Username:** Enter the username.
 - **S3-Presto Password:** Enter the password.
-
- Click , a row is added.
 - All the **incoming columns** are available as drop down in the **Enter Partition Name field**. Select the required column based on which the data must be partitioned.
 - To remove the column click **Delete** icon.
-
- **File Format:** Select the file format to store the output data in S3 External - Table.

8. Click **Save**.

Note :

- The created S3 External - Table Sink is available in the Operators > Data Writers grid > specific category for future usage.
- After all the configurations are done, save and run the Pipeline.
- We can get the output in as a table which we will be available in Data catalog.

Here is a configured example:

Create New Category

To create a category in the S3 External - Table Sink configuration grid:

1. Click on **Select Category** configuration element. The configuration element is expanded.
2. Enter the category name in **Search Category**.
3. Click . A category is added in the drop-down list.

GCS External - Table Sink

Last updated on May 12, 2023

GCS sink is a Google Cloud Storage sink with restfull online file storage web service for storing and accessing the table data on Google Cloud Platform infrastructure.

To configure GCS External - Table Sink:

1. In the Canvas, go to **Operators** grid > **Data Writer**.
2. Drag and drop the **GCS External - Table Sink** into the canvas.
3. Connect the output of transformation operator as input to **GCS External - Table Sink**.
4. Click vertical ellipses of **GCS External - Table Sink** > click **Edit**. The **Configure GCS External - Table Sink** window is displayed.
5. Click the expandable configuration **General Details**.

Configure the following:

- **Data Writer Name:** Enter the name for Sink.
 - **Description:** Allows you to enter the description for the GCS External - Table sink.
 - **Select category:** Select the category to which the GCS External - Table Sink belong to. You can also create a new category to add the sinks. For more information on creating category, see [Create New Category](#).
 - **Tags:** Enter the tags that depicts the purpose of GCS External - Table sink in a pipeline.
6. Click **Next** or click **Configuration**.
 7. Configure the following:
 - **Upload File :** Allows you to upload a json file to authenticate the GCS External - Table connection.
 - **KeyJson:** Displays the preview of the JSON file.
 - **Hive Thrift URL:** Enter the hive host link.
 - **Warehouse Directory:** Enter the warehouse directory path.
 - **Schema Name:** Provide the schema details.
 - **S3-Presto url:** Enter the link for the S3 Presto.
 - **S3-Presto Username:** Enter the username.
 - **S3-Presto Password:** Enter the password.
 - **Table Name:** Enter the table name.

- **Partition Columns:** It allows you to add partition table.
 - Click , a row is added.
 - All the **incoming columns** are available as drop down in the **Select Partition Name** field. Select the required column based on which the data must be partitioned.
 - To remove the column click **Delete** icon.
- **File Format:** Select the file format to store the output data in GCS External - Table.

8. Click **Save**.

Note :

- The created GCS External - Table Sink is available in the Operators > Data Writers grid > specific category for future usage.
- After all the configurations are done, save and run the Pipeline.

Here is a configured example:

Create New Category

To create a category in the GCS External - Table Sink configuration grid:

1. Click on **Select Category** configuration element. The configuration element is expanded.
2. Enter the category name in **Search Category**.
3. Click . A category is added.

Azure Blob External-table Sink

Last updated on May 12, 2023

Azure Blob External - Table Sink used to dump processed data into azure storage in parquet format. The table schema and data will be visible in data catalogue.

To configure Azure Blob External - Table Sink:

1. In the Canvas, go to **Operators** grid > **Data Writer**.
2. Drag and drop the **Azure Blob External - Table Sink** into the canvas.
3. Connect the output of transformation operator as input to **Azure Blob External - Table Sink**.
4. Click vertical ellipses of **Azure Blob External - Table Sink** > click **Edit**. The application displays **Configure Azure Blob External - Table Sink** window.

5. Click the expandable configuration **General Details**.

Configure the following:

- **Data Writer Name:** Enter the name for Sink.
- **Description:** Allows you to enter the description for the Azure Blob External - Table Sink.
- **Select Category:** Select the category to which the Azure Blob External - Table Sink belong to. You can also create a new category to add the sinks. For more information on creating category, see [Create New Category](#).
- **Tags:** Enter the tags that depicts the purpose of Azure Blob External - Table Sink in a pipeline.

6. Click **Next** or click **Configuration**.

7. Configure the following:

- **Account Name:** Enter the account name.
 - **Account Key:** Enter the account key.
 - **Container Name:** Enter the container name.
 - **Hive Thrift URL:** Enter the hive host link.
 - **Warehouse Directory:** Enter the warehouse directory path.
 - **Schema Name:** Enter the schema details.
 - **Table Name:** Enter the table name.
 - **Trino URI:** Enter the trino URI.
 - **Trino Username:** Enter the trino username.
 - **Trino Password:** Enter the trino password.
-
- Click , a row is added.
 - All the **incoming columns** are available as drop down in the **Enter Partition Name** field. Select the required column based on which the data must be partitioned.
 - To remove the column click **Delete** icon.
-
- **File Format:** Select the file format to store the output data in Azure Blob External - Table Sink.

8. Click **Save**.

Note:

- **After all the configurations are done, save and run the Pipeline.**

Here is a configured example:

Create New Category

To create a category in the Azure Blob External - Table Sink configuration grid:

1. Click on **Select Category** configuration element. The configuration element is expanded.
2. Enter the category name in **Search Category**.
3. Click **+**. A category is added in the drop-down list.

Notification Sink

Last updated on October 4, 2023

Notification Sink is a type of data sink in HyperSense that configures the email notifications and SMS notifications in order to get **Email** and **SMS** notifications to the configured users after completion of pipeline execution.

Note:

The **Email Server Information** has to be pre-configured in **Email Server** tab under **common settings configuration page** in order to get the email notifications to the configured users. Email notification will not work if you do not configure the email server information. For more information, see *Email Server*.

Email Notification

To configure the Email Notification Sink:

1. In the Canvas, go to **Operators** grid > **Data Writer**.
2. Drag and drop the **Notification Sink** into the canvas.

3. Connect the output of **Transform / Data Reader** operator as input to **Notification Sink**.
4. Click vertical ellipses of **Notification Sink** > click **Edit**. The **Configure Notification Sink** window is displayed.
5. Configure the following:
 - a. **Enter Notification Name**: Enter the name of notification.
 - b. **Internal** tab: Follow the below steps
 - i. **To, Cc and Bcc**: Enter or add the recipient Email ID.
Note: The recipients names will be reflected in the To, Cc, and Bcc, if the person details or data is configured in the System Admin.
 - ii. **Subject**: Enter the subject.
 - iii. **Body text**: In the **Body text** field write the content using the **Formatting text options**. Located above the Body text field area.
Note: When you drag and drop the data from the **Field** pane, the data will be displayed in the "Delimiter Enclosed Column/Field Name" format. The notification get fails in case of typo error in the format.
Example: "##string_text##".
 - iv. **Drag and drop the desired option** from the **Fields** pane, located left side of the page.
 - c. **Field**: This pane shows the created input columns.
 - i. **Search Field**: Enter the search criteria in the search field and click **Search** to find the required option from the list.
6. Click **Save**.

Note :

- After all the configurations are done, save and run the Pipeline.
- Upon running the pipeline, the mail will be sent to the mail ID's of TO, CC, and BCC Recipients.
- If some value is selected, based on the input value the number of mails get generates. If there are 10 records in the input then the application generates 10 mails to the user.

SMS Notification

Note:

The SMS Host information has to be pre-configured in SMS Host tab under common settings configuration page in order to get the SMS notifications to the configured users. SMS notification will not work if you do not configure the SMS Host. For more information, see *SMS Host*.

To configure the SMS Notification Sink:

1. From the canvas, go to **Operators** grid > **Data Writer**.
2. Drag and drop the **Notification Sink** into the canvas.
3. Connect the output of **Data Reader/Transform** operator as input to **Notification Sink**. Click vertical ellipses of **Notification Sink** > click **Edit**.
4. The **Configure Notification Sink** window is displayed.
5. Click the **SMS** tab. The SMS configuration page is displayed.
6. Configure the following fields:
 - a. **Available Fields**: The columns that are available in selected **output of previous Node or Operator** are listed here.
 - b. **Search Fields**: To search any field, enter the required column name in **Search Fields** and hit enter.
 - c. **Enter Notification Name**: Enter a name for notification in **Enter Notification Name** field.
 - d. **Send To**: Place the cursor in **Send To** field and select the required phone number, required user to whom the notification has to be sent. You can also click upload a csv file with the list of phone numbers in bulk.
 - e. **Enter your message**: Enter the message body that has to be sent through SMS notification. You can drag and drop any field from the fields available under Available Fields to tag any field in message body.
7. Click **Save** to save the configuration.

Note :

- If you want to configure Email notification and SMS notification together, you have to provide same notification name for both Email and SMS in **Enter Notification Name** field.
- After all the pipeline configuration is done, save and run the Pipeline. Upon running the pipeline, an SMS will be sent to the configured recipient phone numbers/user.
- If some value is selected, based on the input value the number of SMS gets generated. If there are 10 records in the input, then the application generates 10 SMS to the user.

Case Operator Sink

Last updated on February 11, 2025

Case Operator : Generating cases in PAS from streaming pipelines based on the thresholds defined in the input operators. Cases will be grouped based on one or more dimension variables selected.

To configure Case Operator Sink:

1. In the Canvas, go to **Operators** grid > **Data Writer**.
2. Drag and drop the **Case Operator Sink** into the canvas.
3. Connect the output of transformation operator as input to **Case Operator Sink**.
4. Click vertical ellipses of **Case Operator Sink** > click **Edit**. The **Case Operator Sink** window is displayed.
5. Configure the following:
 - a. **Name:** Enter the name. Maximum 120 characters are allowed.

Limitation:

While importing and cloning the pipeline, the system will add 9 and 6 characters, respectively, in the name. It is advised to use maximum of 110 characters to avoid failing the pipeline import and clone action.

- b. **Case Template:** Select the required case template from the drop-down list.

Note:

The source data in the case template is reflected from the PAS.

- c. **Measure:** Select the required measure from the drop-down list.
- d. **Values:** Select the required values from the drop-down list.
- e. **Participating Record Enabled:** This feature is used to generate the participation record. Select the **Participating Record Enabled** checkbox to generate the PR. To configure PR, refer to [Participation Record](#).

Note:

- If **Participating Record Enabled** checkbox is disabled, PR is not generated.
- If selected, note is displayed "**Select window start and end time in dynamic variables**". Window variables displayed based on the window interval configured in *SummarizeX*. If there are 2 window configuration, then 2 set of start time and end time is displayed. First window is labeled as window1_startTime and window1_endTime and then subsequent window labeling follows.
- **As per requirement, select window start time and end time to generate Participating Record.** Selected start and end time should be from the same window.

- f. **Dimension Variables:** Select the required dimension variable from the pane list. To select, click the checkbox of the desired option.

Note:

To avoid the re-occurrence of output with the same data from other data fields, user can use the **Case Operator** checkbox from the **Dimension Variables** pane. Which allows the user to group the required data and generate perfect output.

- g. **Dynamic Variables:** Select the single or multiple options from the dynamic variable pane. To select, click the checkbox of the desired option.

Note:

The number of columns returned as output by the last **Transform operators** will be available as an input to the case operator. The data will be reflected in the **Measure, Values, Dimension Variables**(except decimal data types) and **Dynamic Variables**.

6. Click **Save**.

Configured Example:

Note :

- **After all the configurations are done, save and run the Pipeline.** The output result displays in the Process Automation Studio > Cases tab > Grouped Cases tab.
- **To Import / Export using the Case Operator can be done as below:**
 - In case of Case Operator import, user needs to first export the case template linked to the Case Operator from PAS followed by the Case Operator pipeline.
 - Then import the case template in PAS followed by Case Operator pipeline in the other machine.

Go back to [workflow configuration](#) page to know about **Manual Case Creation** from Process Automation.

Case Operator Batch Sink

Last updated on December 17, 2024

Case Operator Batch : Generating cases in PAS for offline or batch processing pipelines based on selected dimension variables.

To configure Case Operator Batch Sink:

1. In the Canvas, go to **Operators** grid > **Data Writer**.
2. Drag and drop the **Case Operator Batch Sink** into the canvas.
3. Connect the output of transformation operator as input to **Case Operator Batch Sink**.
4. Click vertical ellipses of **Case Operator Batch Sink** > click **Edit**.
5. The **Case Operator Batch Sink** window is displayed. Configure the following:

- a. **Name:** Enter the name.
- b. **Case Template:** Select the required case template from the drop-down list.

Note:

The source data in the case template is reflected from the PAS.

- c. **Measure:** Select the required measure from the drop-down list.
- d. **Values:** Select the required values from the drop-down list.
- e. **QM Table Sources:** Select the input QM table to generate the PR. To know how to configure PR in Query Measure, see [PR Configuration](#).
- f. **Dimension Variables:** Select the required dimension variable from the pane list. To select, click the checkbox of the desired option.

Note:

To avoid the re-occurrence of output with the same data from other data fields, user can use the **Case Operator** checkbox from the **Dimension Variables** pane. Which allows the user to group the required data and generate perfect output.

- g. **Dynamic Variables:** Select the single or multiple options from the dynamic variable pane. To select, click the checkbox of the desired option.

Note:

The number of columns returned as output by the last **Transform operators** will be available as an input to the case operator. The data will be reflected in the **Measure, Values, Dimension Variables**(except decimal data types) and **Dynamic Variables**.

6. Click **Save**.

Configured Example:

Note :

- After all the configurations are done, save and run the Pipeline. The case will get generated in Process Automation Studio.
- To Import / Export using the Case Operator can be done as below:
 - In case of Case Operator import, user needs to first export the case template linked to the Case Operator from PAS followed by the Case Operator pipeline.
 - Then import the case template in PAS followed by Case Operator pipeline in the other machine.

Go back to [workflow configuration](#) page to know about **Manual Case Creation from Process Automation**.

Azure Blob Sink

Last updated on May 12, 2023

Azure Blob Sink is used to dump processed data into azure storage in csv or json format.

To configure Azure Blob Sink:

1. In the Canvas, go to **Operators** grid > **Data Writer**.
2. Drag and drop the **Azure Blob Sink** into the canvas.
3. Connect the output of transformation operator as input to **Azure Blob Sink**.
4. Click vertical ellipses of **Azure Blob Sink** > click **Edit**. The application displays **Configure Azure Blob Sink** window.
5. Click the expandable configuration **General Details**.
6. Configure the following:
 - **Data Writer Name**: Enter the name for Sink.
 - **Description**: Allows you to enter the description for the Azure Blob Sink.
 - **Select Category > Add/Search category**: Select the category to which the Azure Blob Sink belong to. You can also create a new category to add the sinks. For more information on creating category, see [Create New Category](#).
 - **Tags**: Enter the tags that depicts the purpose of Azure Blob Sink in a pipeline.
7. Click **Next** or click the expandable of the **Configuration** section.

8. Configure the following:

- **Account Name:** Enter the account name.
- **Account Key:** Enter the account key.
- **Container Name:** Enter the container name.
- **Directory:** Enter the directory in the azure storage, to store the output data.
- **Compression type:** Select the compression type from the dropdown list. For example: If the selected format is gzip, the output data is compressed and stored in gzip format.
- **File Extension Pattern:** Enter the File Extension. For example: *.csv.
- **Backup Strategy:** Select the backup strategy from the dropdown list.
- **File Format:** Select the file format to store the output data in Azure. **CSV** and **JSON** formats are supported. If you select the file format as **CSV**, then the below two additional fields are enabled.
 - **Delimiter:** Enter the delimiter to separate the values that are being stored in Azure.
 - **Is Header Present:** If the checkbox is checked or enabled, the output will be stored along with the header. If the checkbox is unchecked or disabled, the output will be stored without the header.

9. Click **Save**.

Note :

- The created Azure Blob Sink is available in the Operators > Data Writers grid > specific category for future usage.
- After all the configurations are done, save and run the Pipeline.

Here is a configured example:

Create New Category

To create a category in the Azure Blob Sink configuration grid:

1. Click on **Add/Search Category** configuration element. The configuration element is expanded.

2. Enter the category name in **Search Category**.
3. Click . A category is added in the drop-down list.

API Sink

Last Updated on September 23, 2024

API Sink data writer allows you to call the external API(s). It allows you to select the API(s) registered within a centralized API framework.

Note:

- API Sink Data Writer is available for Streaming as well as Batch Pipelines.
- You can configure multiple API(s).

Follow the below steps to configure the API Sink data writer:

1. In the Canvas, go to **Operators** grid > **Data Writer**.
2. Drag and drop the **API Sink** data writer into the canvas.
3. Provide an input link to the **API Sink**.
4. Click the vertical ellipses of **API Sink** > **Edit**. The **Configure API Sink** window is displayed.
5. Configure below details:
 - a. **Configure API**: This tab allows you to select the API(s) registered within a centralized API Framework. Select the API(s) from the dropdown list. Upon selection, API is added to the **Selected API's** section. API is not configured and status is displayed as "Not Configured". Expand the API to configure.

Note:

Click Delete icon to delete the Selected API's.

Complete the below details:

- i. **Request Mapping**: It allows you to map the data from the SOAP/REST API(s). Select the input map column from the dropdown list of the selected API that will request the data. Click **Next** after input columns selection, you are navigated to Response Mapping tab.
- ii. **Response Mapping**: It allows you to store the data from the selected SOAP/REST API(s). Enter the mapper column name where the output data should be stored.
 - **Previous**: Click to navigate to back to **Request Mapping** tab.
 - **Validate**: Click to validate the configuration.

- When API is validated, Config Status is displayed as **Configured**.
 - If Output Column is not mapped, below snackbar message is displayed.
- iii. **Click the Submit button** to save the **API Sink** configuration. Upon save, snackbar message is displayed "**API Sink Configured Successfully**".
- b. **Results Preview:** This tab allows you to preview the data table output as per the Request and Response Mapping configuration. In this tab, data table for configured API(s) is displayed. **Select the configured API(s) from the Select API dropdown list.**

Note:

- To view the **Result Preview**, make sure pipeline run is completed.
- You can copy the API response. Select the response > Right Click > Click Copy.
- You can retry the API responses. Select the response > Right Click > Click Retry.

6. Output table of **API Sink** is viewed in Data Catalog.