PRIVITAR

User Guide

Privitar Data Privacy Platform, version 4.4.1

Publication date September 6, 2023

Privitar Data Privacy Platform, version 4.4.1

Table of Contents

1. Introduction

The aim of this guide is to describe how to de-identify an input dataset using the Privitar platform.

1.1. Get Started: De-identifying Data in Four Steps

Start with De-Identifying Data in Four Steps for an introduction to the key concepts used by the Privitar platform. The guide describes setting up input data, creating a Policy, and producing de-identified output data.

1.2. 'How to' Guides

'How To Guides' are short sections that describe how to perform various tasks on the Privitar Platform, such as setting up and managing processing jobs, system administration, and advanced data format support.

2. How-To Guides

To get started, it is helpful to begin with an end-to-end overview of how to de-identify data using Privitar. For more information, see De-Identifying Data in Four Steps. These concepts are central to understanding how to use Privitar.

After familiarizing yourself with the concepts and terms used in Privitar, these sections will help you to configure the software to de-identify data:

- How to describe input data with a Schema. See What is a Schema?
- How to specify de-identification using a Policy. See What is a Policy?
- How to manage safe releases of data with Protected Data Domains. See What is a PDD?
- How to use Jobs to execute a de-identification process on the data. See What is a Job?
- Watermarking and investigating datasets. See Watermarking a Dataset.
- How to look up the original value of a token after running a Job. See Unmasking De-identified Data.

Once a configuration has been created, records of processing runs are stored in Privitar. These FAQs will help when working with existing Batch Jobs. For more information about any of these tasks, see Working with Existing Jobs:

- How to find the output data for a previously run Batch Job.
- How to re-run a Batch Job after new data becomes available or after a Policy change.
- How to view more detail about the steps within a Batch Job invocation.
- How to view diagnostic information about a failed Batch Job.

After reading the introduction, these guides may be relevant to your ongoing use of Privitar:

- What is a Token Vault and how is it used by Privitar? See What is a Token Vault?
- How to use Avro, CSV, and Parquet files with Privitar. See Batch data sources and Supported Types.
- How to specify Environments, Token Vault databases, and Hadoop Clusters to use with Privitar. See Environments Administration.
- How to work with partitioned data in Hadoop. See Partitioned Data and Wildcards (HDFS).
- How to automate Privitar configuration and processing with the Privitar Automation APIs. See APIs.
- How to read and write data in Amazon S3 Buckets or Azure Blob Storage. See Specifying Input Data Locations (HDFS and Cloud Storage).
- How to import and export Privitar objects, such as Schemas, Policies, and Jobs. See Importing and Exporting Configurations.
- How to configure Privitar SecureLink. See About Privitar SecureLink.

These sections are available on system administration topics:

- How to add an Environment. See Managing Environments.

- How to edit Users and Roles. See Managing Users, Managing API Users and Managing Roles.
- How to access the Audit log. See Managing the Audit log.
- How to check the health of the system. See Managing Service Health.
- How to monitor system usage. See Managing Service Usage.

3. De-Identifying Data in Four Steps

The four key concepts to understand when de-identifying any dataset using the Privitar Platform are:

3.1. Overview of Schemas, Policies, Protected Data Domains, and Jobs

- A **Schema** is a description of the tables and columns of the input data, likely a Hive database, Avro, Parquet, or CSV file. All data input to Privitar must conform to a Schema.
- A **Policy** describes a transformation of input into an output with privacy-preserving changes applied. Policies are the primary way in which privacy transformations are represented, so it is important that the Policy is created in line with an organisation's privacy requirements.
- A **Protected Data Domain (PDD)** is a set of de-identified datasets intended for the same use. PDDs also capture metadata information such as restrictions, names of stakeholders, and data location. Datasets published to the same PDD can be de-identified *consistently* so that they are linkable; datasets published to different PDDs are not linkable. De-identified data can be traced back to its PDD using its embedded watermark, which is added to data during the de-identification process.
- A **Job** represents the execution of a Policy on some input data, for a destination PDD. Batch Jobs are launched as a Spark Job on a Hadoop cluster from Privitar, and can also be used to re-run specific processing if new data is received or if there is a change to a Policy. Data Flow Jobs are used with external data flow pipelines, such as Apache NiFi, Kafka Connect/Confluent or Streamsets. Privitar On Demand Jobs can be used to enable de-identification in applications via a HTTP API.

3.2. Getting started

The procedure to de-identify a dataset using the Privitar platform is:

1. **Define a Schema for the new input data**. Each table in the Schema contains column definitions, specified in terms of Privitar data types.
   - If there are multiple tables in the data, for example if the data is an extract from a relational database, the Schema will contain multiple, corresponding table definitions.
2. **Create a Policy representing the de-identification operations required**. The Schema provides the basis for how the Policy is structured.
3. **Define the Protected Data Domain** to be used as the destination for all data related to the use case in question.
4. Based on the Policy, **create and execute a Job**. This is done by selecting a Policy and a destination Protected Data Domain.
   - Batch Jobs will be executed from Privitar.
   - Data Flow Jobs will be referenced from external processing pipelines.
   - Privitar On Demand Jobs are available via an API from the Privitar On Demand server.

For more information about the different job types that available from Privitar, see What is a Job?

4. Describing Input Data

This section describes how input data needs to be defined ready for processing by Privitar.

4.1. What is a Schema?

A Privitar Schema is a description of the tables and columns of the input data, likely a Hive database, Avro, Parquet or CSV file. Before data can be de-identified with Privitar, a corresponding Schema must be created in Privitar.

A Schema represents the structure of the input data. It contains:

- A list of named **tables**. Each table must have a name.
- Inside each named table are named **columns** of a particular data type.

Once the Schema exists, Policies can be created based on it. This means that any data that conforms to the Schema can be processed by that Policy.

It is only necessary to create a single Schema for a known set of tables, and then reuse that Schema for as many Policies as required.

4.2. Creating a Schema

A Schema can be created in various ways.

The most common ways to create a Schema are:

- Reading the definition directly from a Hive database. See, Creating a Schema from a Hive database.
- Reading the definition directly from AWS Glue Data Catalog. See, Creating a Schema from an AWS Glue Data Catalog.
- Reading the table definitions from data files in HDFS. See, Creating a Schema from HDFS.
- Specifying the Schema directly in the Privitar user interface. See, Specifying a Schema Directly.

Schemas can also be created using other methods:

- Importing a Privitar JSON configuration file of a previously exported Privitar Schema. See, Creating a Schema from JSON.
- Importing an exported Schema directly into Privitar. See, Importing an Exported Schema.]
- Importing a local CSV file directly into Privitar. See, Creating a Schema from CSV.
- Creating the Schema through the Privitar Automation API. See, Automation API (v3).

4.2.1. Creating a Schema from a Hive database

A Schema can be created by importing table definitions from a Hive database . You can import multiple tables from Hive to build a Schema and then edit the tables and columns using Privitar to finalize the Schema definition.

The process of importing from Hive requires details for the Hive database, because Privitar must connect to that database to read the table definitions.

There are two ways to specify the Hive database. The choices available depend on the Privitar configuration and may not be available on all Privitar installations:

- Connecting to an Environment's Hive configuration.
- Connecting to a manually specified Hive database.

For more information about setting up Hive in Privitar, see Hadoop Cluster Environment Configuration.

Once a Schema has been imported, tables and columns can also be added or modified if required to finalize the Schema definition. For more information, see Adding Tables and Columns to a Schema.

Connecting to an Environment's Hive configuration

To import a Schema from an Environment's Hive configuration, follow these steps:

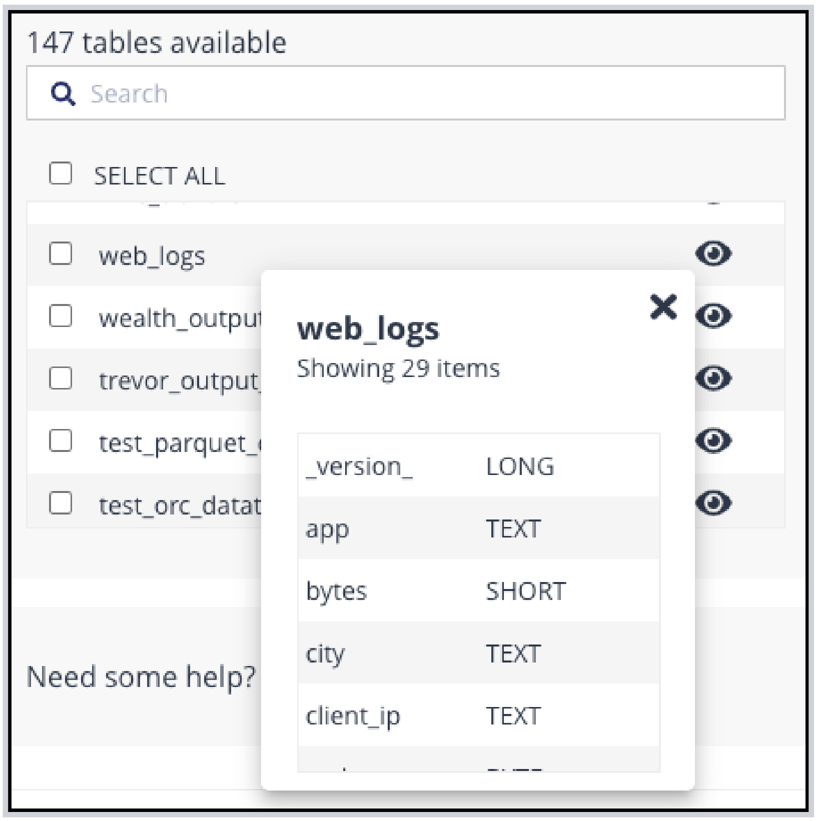
1. Select **Schemas** from the Navigation sidebar. The **Schemas** page is displayed.
2. Click on **Create New Schema**. The **New Schema** window is displayed.
3. Enter a name for the new Schema in the **Name** field.
4. Select **Import from Hive** from the **Import tables** list box. The **Import from Hive** window is displayed.
5. Select an Environment with the desired Hive configuration from the **Environment** list box.
6. If the tables that you are importing contain Date or Timestamp fields, Privitar will import these tables according to the following default Date and Timestamp formats:
   - **Date Format**: `yyyy-MM-dd`
   - **Timestamp Format**: `yyyy-MM-dd'T'HH:mm:ss`

   If you want to change this format, click on the *Cog* icon. Update the formats and click on **OK**.

   (For more information about the Date and Timestamp formats supported by Privitar, see Date and Timestamp formats.)
7. Click on **Fetch Tables**.

   The table details are fetched from the location specified and displayed in the left-hand pane.

   Use the *Eye* icon to the right of a table to inspect the contents of a table. For example:


147 tables available

Search

SELECT ALL

web_logs

wealth_output

trevor_output

test_parquet_

test_orc_datat

Need some help?

web_logs

Showing 29 items

version	LONG
app	TEXT
bytes	SHORT
city	TEXT
client_ip	TEXT

8. Pick the tables to import by selecting the checkbox alongside each table. The selected table is added to the list of tables to import in the right-hand pane.

   To quickly select all tables, use the **Select All** checkbox. You can also use *Shift-click* to select a number of tables.

   To de-select a table from the list of tables to import, click on the *Trash* icon.
9. Click on **Import tables** to add the selected tables to the Schema.

   The tables are imported and the **New Schema** window is updated with the tables that have been imported.
10. Click on a table in the left-hand pane to preview its definition on the right, and to make any edits to any of the columns included in the table. For more information about the editing actions and how to finalize the Schema definition, see Adding Tables and Columns to a Schema.
11. Click on **Save** to save the new Schema.

    The new Schema is added to the list of Schemas on the **Schemas** page.

Connecting to a manually specified Hive database

To import a Schema from a manually specified Hive database, follow these steps:

1. Select **Schemas** from the Navigation sidebar. The **Schemas** page is displayed.
2. Click on **Create New Schema**. The **New Schema** window is displayed.
3. Enter a name for the new Schema in the **Name** field.
4. Select **Import from Hive** from the **Import tables** list box. The **Import from Hive** window is displayed.
5. Select an Environment with the desired Hive configuration from the **Environment** list box

6. Click on **Specify Hive Connection...** . The **Connect to Hive** dialog box is displayed.
7. Select the Cluster type you wish to use from the **Hadoop Cluster Type** list box.
8. Select the authentication method to be used by the Hive connection from the **JDBC Connection Template** list box.

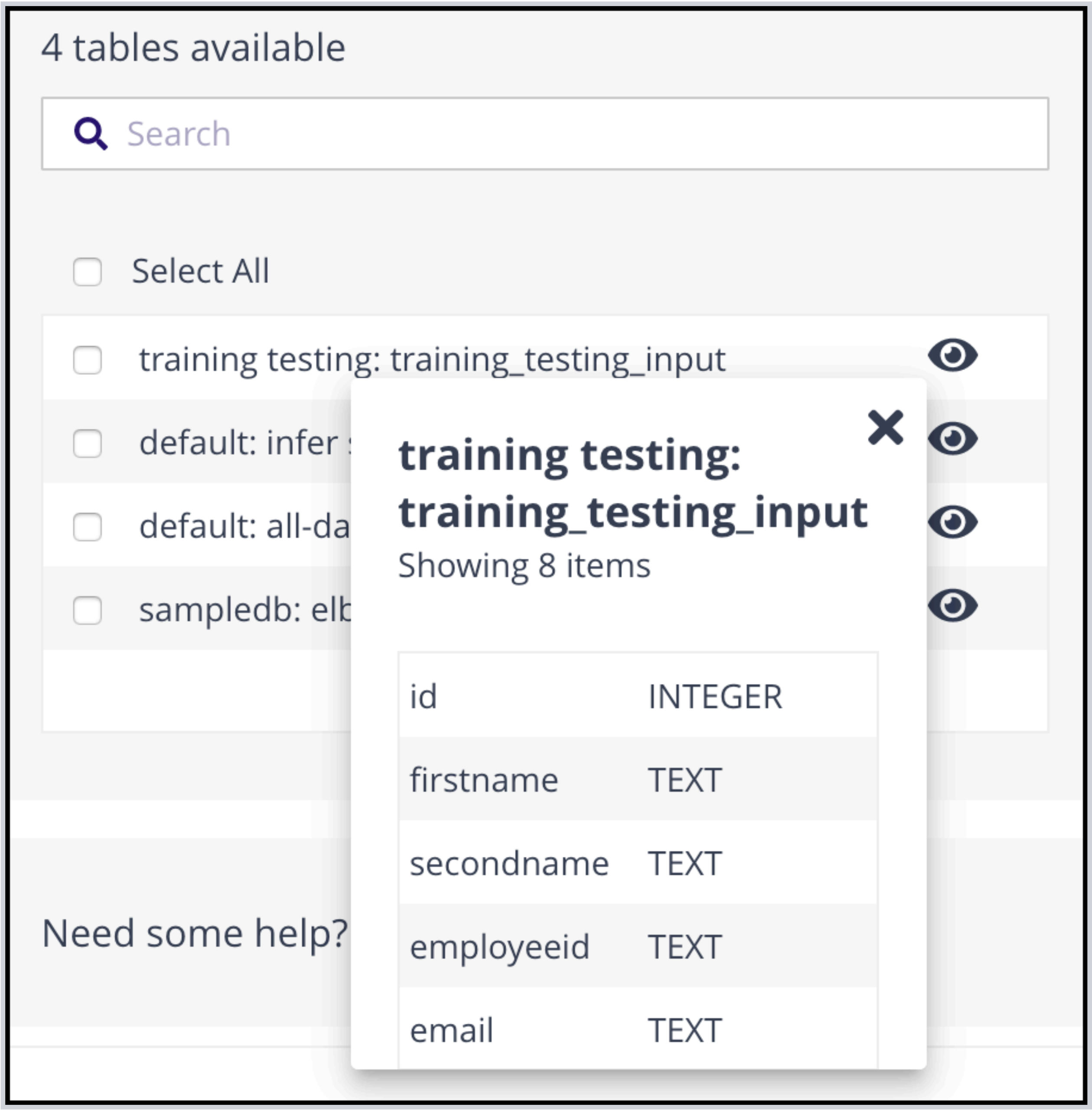
   The following table describes the options:

Method	Description
Hive Basic	Requires a connection URL, a user name and password.
Hive Knox	Requires a connection URL, a user name and password.
Hive Kerberos	Requires a connection URL, Keytab Path, Principal, Key Distribution Center (KDC) URL, and Realm.

   Please contact your system administrator for information on completing these fields.
9. Click on **OK**.

   The table details are fetched from the location specified and displayed in the left-hand pane.

   Use the *Eye* icon to the right of a table to inspect the contents of a table. For example:



training testing: training_testing_input
Showing 8 items

id	INTEGER
firstname	TEXT
secondname	TEXT
employeeid	TEXT
email	TEXT

10. Pick the tables to import by selecting the checkbox alongside each table. The selected table is added to the list of tables to import in the right-hand pane.
11. Click on **Import tables** to add the selected tables to the Schema.

    The tables are imported and the **New Schema** window is updated with the tables that have been imported.

    Click on a table in the left-hand pane to preview its definition on the right, and to make any edits to any of the columns included in the table. For more information about the editing actions and how to finalise the Schema definition, see Adding Tables and Columns to a Schema.
12. Click on **Save** to save the new Schema.

    The new Schema is added to the list of Schemas on the **Schemas** page.

4.2.2. Creating a Schema from an AWS Glue Data Catalog

A Schema can be created by importing table definitions from an AWS Glue Data Catalog. You can import multiple tables from the AWS Glue Data Catalog to build a Schema and then edit the tables and columns using Privitar to finalize the Schema definition.

The process of importing from an AWS Glue Data Catalog requires the details of an associated AWS Glue environment which Privitar must connect to in order to read table definitions. This environment depends on the Privitar configuration and may not be available on all Privitar installations.

For more information about setting up AWS Glue in Privitar, see AWS Glue Environment Configuration.

Once a Schema has been imported, tables and columns can also be added or modified if required to finalize the Schema definition. For more information, see Adding Tables and Columns to a Schema.

Connecting to an Environment's AWS Glue Data Catalog configuration

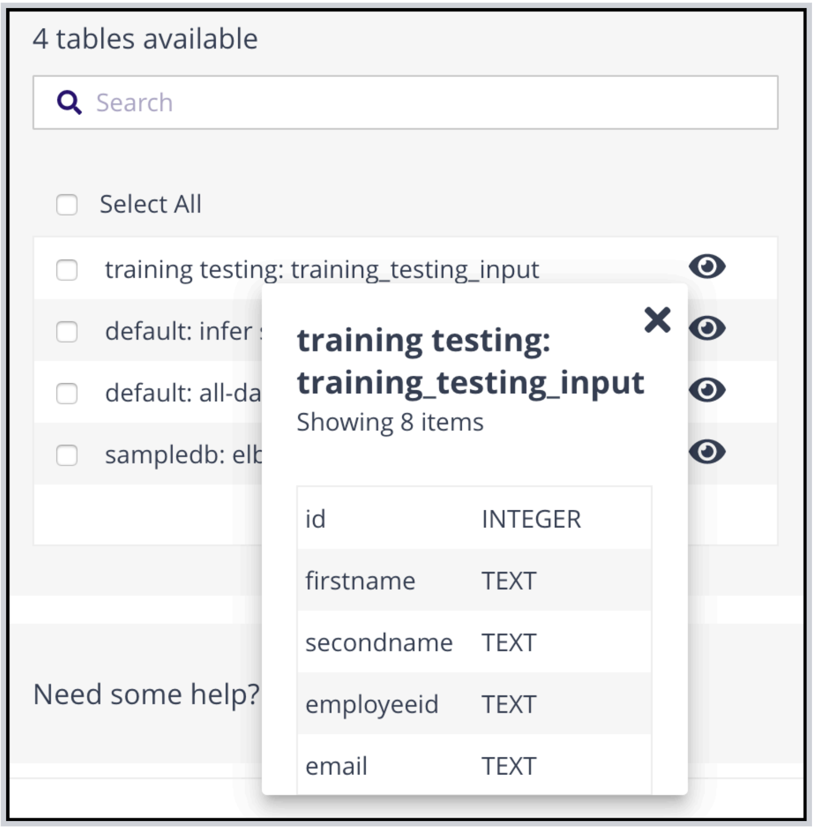
To import a Schema from an Environment's AWS Glue Data Catalog configuration, follow these steps:

1. Select **Schemas** from the Navigation sidebar. The **Schemas** page is displayed.
2. Click on **Create New Schema**. The **New Schema** window is displayed.
3. Enter a name for the new Schema in the **Name** field.
4. Select **Import from AWS Glue Data Catalog** from the **Import tables** list box. The **Import from AWS Glue Data Catalog** window is displayed.
5. Select an Environment with the desired AWS Glue Data Catalog configuration from the **Environment** list box.
6. If the tables that you are importing contain Date or Timestamp fields, Privitar will import these tables according to the following default Date and Timestamp formats:
   - **Date Format**: `yyyy-MM-dd`
   - **Timestamp Format**: `yyyy-MM-dd'T'HH:mm:ss`

   If you want to change this format, click on the *Cog* icon. Update the formats and click on **OK**.
7. Click on **Fetch Tables**.

   The table details are fetched from the location specified and displayed in the left-hand pane.

   Use the *Eye* icon to the right of a table to inspect the contents of a table. For example:


4 tables available

Search

Select All

training testing: training_testing_input

default: infer

default: all-da

sampledb: elb

Need some help?

training testing: training_testing_input

Showing 8 items

id	INTEGER
firstname	TEXT
secondname	TEXT
employeeid	TEXT
email	TEXT

8. Pick the tables to import by selecting the checkbox alongside each table. The selected table is added to the list of tables to import in the right-hand pane.

   To quickly select all tables, use the **Select All** checkbox. You can also use *Shift-click* to select a number of tables.

   To de-select a table from the list of tables to import, click on the *Trash* icon.
9. Click on **Import tables** to add the selected tables to the Schema.

   The tables are imported and the **New Schema** window is updated with the tables that have been imported.
10. Click on a table in the left-hand pane to preview its definition on the right, and to make any edits to any of the columns included in the table. For more information about the editing actions and how to finalize the Schema definition, see Adding Tables and Columns to a Schema.
11. Click on **Save** to save the new Schema.

    The new Schema is added to the list of Schemas on the **Schemas** page.

4.2.3. Creating a Schema from HDFS

A Schema can be created by importing table definitions from data files in HDFS. The supported HDFS file types are:

- Avro (`.avro`)
- Avro Schema (`.avsc`)
- CSV (`.csv`)
- Parquet (`.parquet`)

The process of importing from HDFS requires an Environment to be specified that has had a Hadoop Cluster added to it. Privitar must be able connect to a Hadoop Cluster to read the data files. (For more information about adding a Hadoop Cluster to an Environment, see Hadoop Cluster Environment Configuration.)

To create a Schema from HDFS, follow these steps:

1. Select **Schemas** from the Navigation sidebar. The **Schemas** page is displayed.
2. Click on **Create New Schema**. The **New Schema** window is displayed.
3. Enter a name for the new Schema in the **Name** field.
4. Select **Import from File (HDFS)** from the **Import tables** list box. The **Import from File (HDFS)** window is displayed.
5. Select an HDFS Environment that represents the cluster containing the data files to import, from the **Environment** list box.
6. Enter a path in the **Path** box, referencing either a single data file containing the table definition to import, or a directory containing multiple such data files.
7. If the tables that you are importing contain Date or Timestamp fields, Privitar will import these tables according to the following default Date and Timestamp formats:

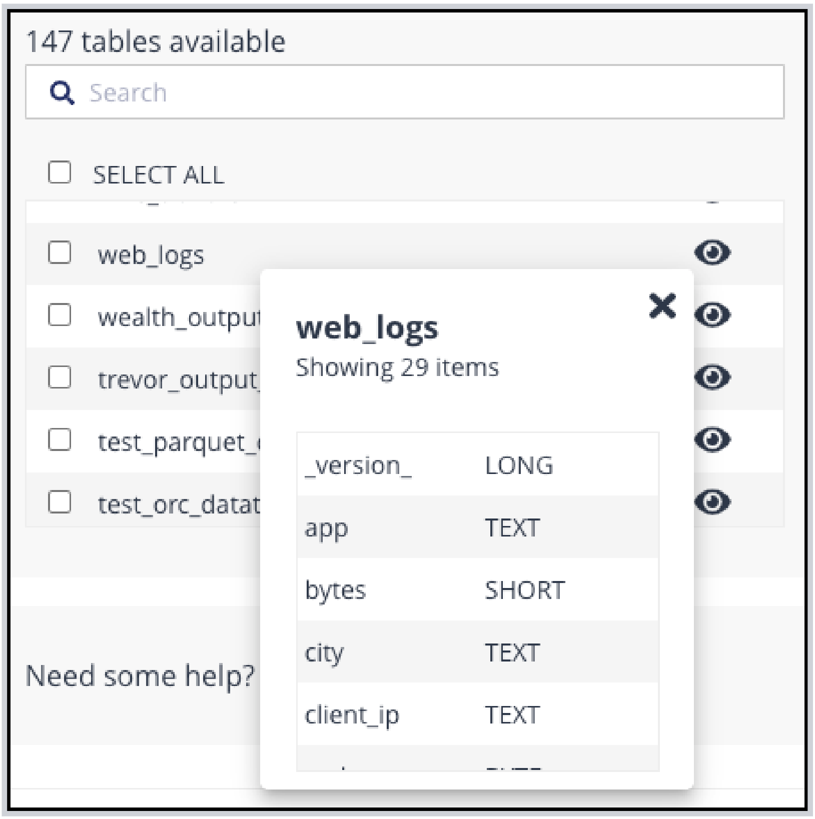
    - **Date Format**: `yyyy-MM-dd`
    - **Timestamp Format**: `yyyy-MM-dd'T'HH:mm:ss`

    If you want to change this format, click on the *Cog* icon. Update the formats and click on **OK**.

    (For more information about the Date and Timestamp formats supported by Privitar, see Date and Timestamp formats.)
8. Click on **Fetch Tables**.

    The table details are fetched from the location specified and displayed in the left-hand pane.

    Use the *Eye* icon to the right of a table to inspect the contents of a table. For example:



web_logs
Showing 29 items

version	LONG
app	TEXT
bytes	SHORT
city	TEXT
client_ip	TEXT

9. Pick the tables to import by selecting the checkbox alongside each table. The selected table is added to the list of tables to import in the right-hand pane.

   To quickly select all tables, use the **Select All** checkbox. You can also use *Shift-click* to select a number of tables.

   To de-select a table from the list of tables to import, click on the *Trash* icon.
10. Click on **Import tables** to add the selected tables to the Schema.

    The tables are imported and the **New Schema** window is updated with the tables that have been imported.
11. Click on a table in the left-hand pane to preview its definition on the right, and to make any edits to any of the columns included in the table. For more information about the editing actions and how to finalize the Schema definition, see Adding Tables and Columns to a Schema.
12. Click on **Save** to save the new Schema.

    The new Schema is added to the list of Schemas on the **Schemas** page.

4.2.4. Creating a Schema from a Hive database

A Schema can be created by importing table definitions from a Hive database . You can import multiple tables from Hive to build a Schema and then edit the tables and columns using Privitar to finalize the Schema definition.

The process of importing from Hive requires details for the Hive database, because Privitar must connect to that database to read the table definitions.

There are two ways to specify the Hive database. The choices available depend on the Privitar configuration and may not be available on all Privitar installations:

- Connecting to an Environment's Hive configuration.
- Connecting to a manually specified Hive database.

For more information about setting up Hive in Privitar, see Hadoop Cluster Environment Configuration.

Once a Schema has been imported, tables and columns can also be added or modified if required to finalize the Schema definition. For more information, see Adding Tables and Columns to a Schema.

Connecting to an Environment's Hive configuration

To import a Schema from an Environment's Hive configuration, follow these steps:

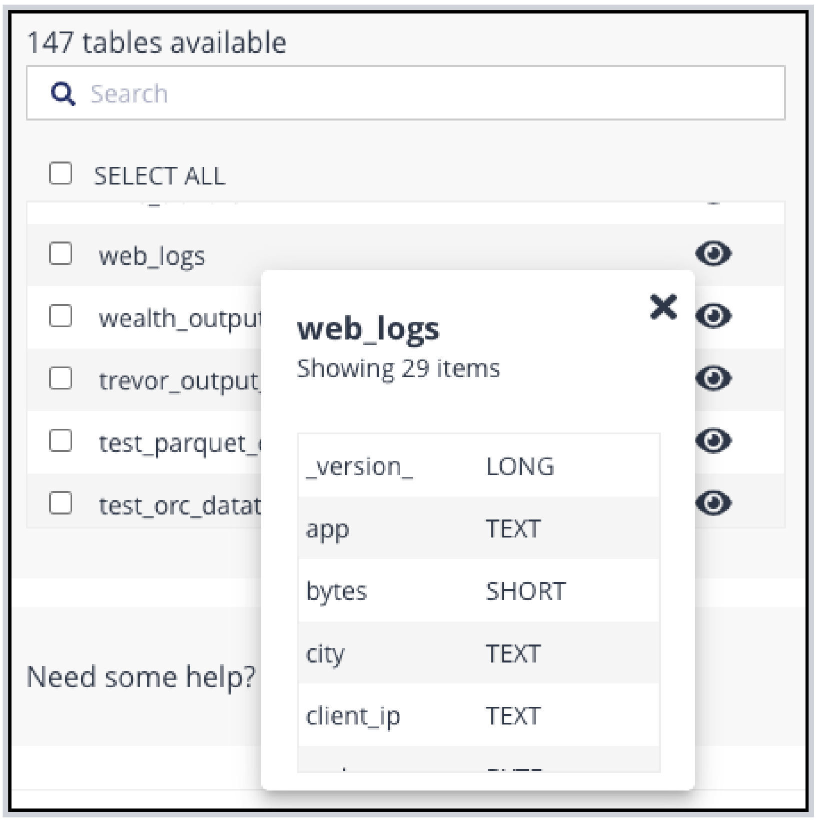
1. Select **Schemas** from the Navigation sidebar. The **Schemas** page is displayed.
2. Click on **Create New Schema**. The **New Schema** window is displayed.
3. Enter a name for the new Schema in the **Name** field.
4. Select **Import from Hive** from the **Import tables** list box. The **Import from Hive** window is displayed.
5. Select an Environment with the desired Hive configuration from the **Environment** list box.
6. If the tables that you are importing contain Date or Timestamp fields, Privitar will import these tables according to the following default Date and Timestamp formats:
    - **Date Format**: `yyyy-MM-dd`
    - **Timestamp Format**: `yyyy-MM-dd'T'HH:mm:ss`

    If you want to change this format, click on the *Cog* icon. Update the formats and click on **OK**.

    (For more information about the Date and Timestamp formats supported by Privitar, see Date and Timestamp formats.)
7. Click on **Fetch Tables**.

    The table details are fetched from the location specified and displayed in the left-hand pane.

    Use the *Eye* icon to the right of a table to inspect the contents of a table. For example:



web_logs
Showing 29 items

version	LONG
app	TEXT
bytes	SHORT
city	TEXT
client_ip	TEXT

8. Pick the tables to import by selecting the checkbox alongside each table. The selected table is added to the list of tables to import in the right-hand pane.

   To quickly select all tables, use the **Select All** checkbox. You can also use *Shift-click* to select a number of tables.

   To de-select a table from the list of tables to import, click on the *Trash* icon.

9. Click on **Import tables** to add the selected tables to the Schema.

   The tables are imported and the **New Schema** window is updated with the tables that have been imported.

10. Click on a table in the left-hand pane to preview its definition on the right, and to make any edits to any of the columns included in the table. For more information about the editing actions and how to finalize the Schema definition, see Adding Tables and Columns to a Schema.

11. Click on **Save** to save the new Schema.

    The new Schema is added to the list of Schemas on the **Schemas** page.

Connecting to a manually specified Hive database

To import a Schema from a manually specified Hive database, follow these steps:

1. Select **Schemas** from the Navigation sidebar. The **Schemas** page is displayed.
2. Click on **Create New Schema**. The **New Schema** window is displayed.
3. Enter a name for the new Schema in the **Name** field.
4. Select **Import from Hive** from the **Import tables** list box. The **Import from Hive** window is displayed.
5. Select an Environment with the desired Hive configuration from the **Environment** list box

6. Click on **Specify Hive Connection...** . The **Connect to Hive** dialog box is displayed.
7. Select the Cluster type you wish to use from the **Hadoop Cluster Type** list box.
8. Select the authentication method to be used by the Hive connection from the **JDBC Connection Template** list box.

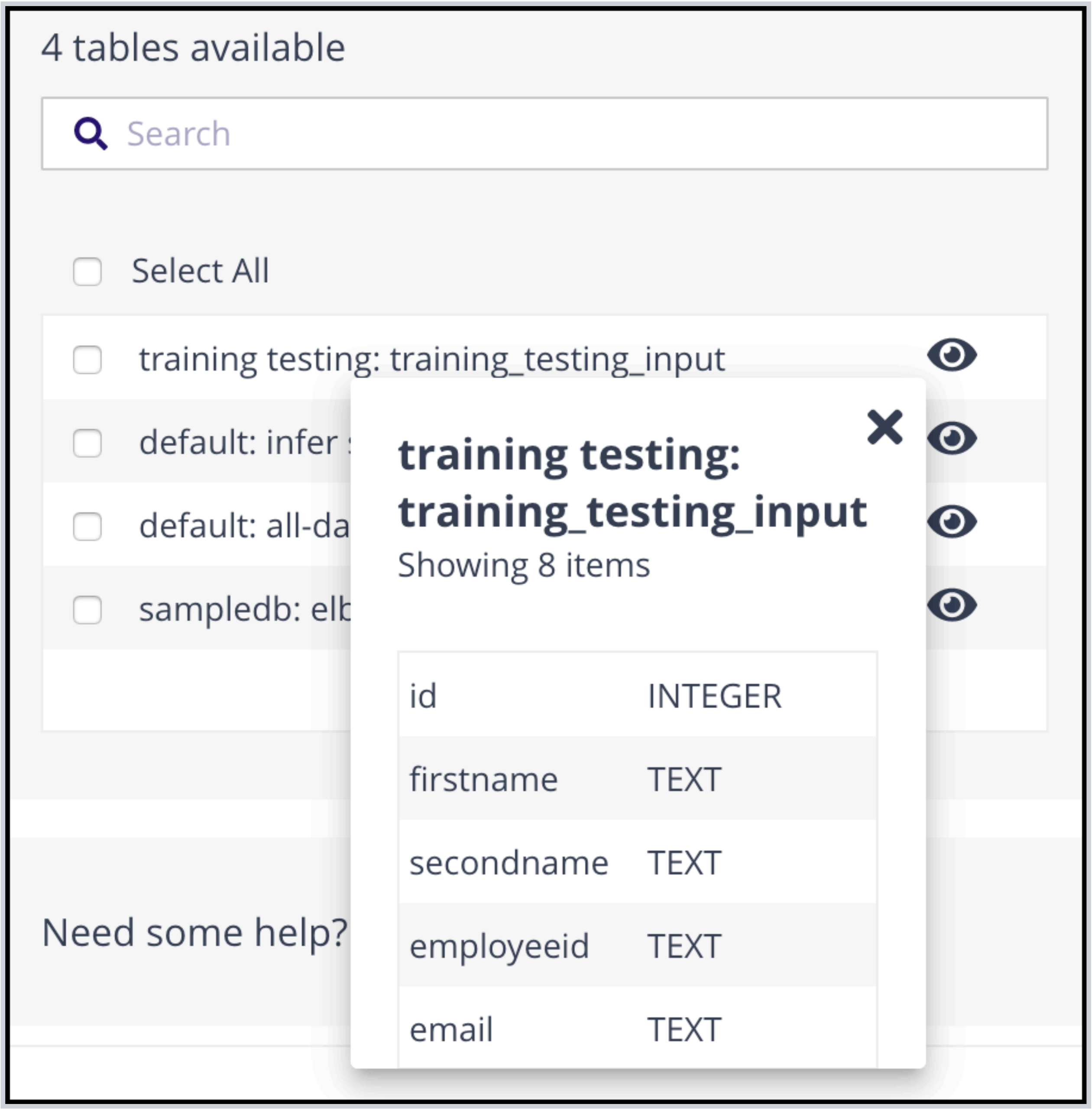
   The following table describes the options:

Method	Description
Hive Basic	Requires a connection URL, a user name and password.
Hive Knox	Requires a connection URL, a user name and password.
Hive Kerberos	Requires a connection URL, Keytab Path, Principal, Key Distribution Center (KDC) URL, and Realm.

   Please contact your system administrator for information on completing these fields.
9. Click on **OK**.

   The table details are fetched from the location specified and displayed in the left-hand pane.

   Use the *Eye* icon to the right of a table to inspect the contents of a table. For example:



training testing: training_testing_input
Showing 8 items

id	INTEGER
firstname	TEXT
secondname	TEXT
employeeid	TEXT
email	TEXT

10. Pick the tables to import by selecting the checkbox alongside each table. The selected table is added to the list of tables to import in the right-hand pane.
11. Click on **Import tables** to add the selected tables to the Schema.

    The tables are imported and the **New Schema** window is updated with the tables that have been imported.

    Click on a table in the left-hand pane to preview its definition on the right, and to make any edits to any of the columns included in the table. For more information about the editing actions and how to finalise the Schema definition, see Adding Tables and Columns to a Schema.
12. Click on **Save** to save the new Schema.

    The new Schema is added to the list of Schemas on the **Schemas** page.

4.2.5. Creating a Schema from JSON

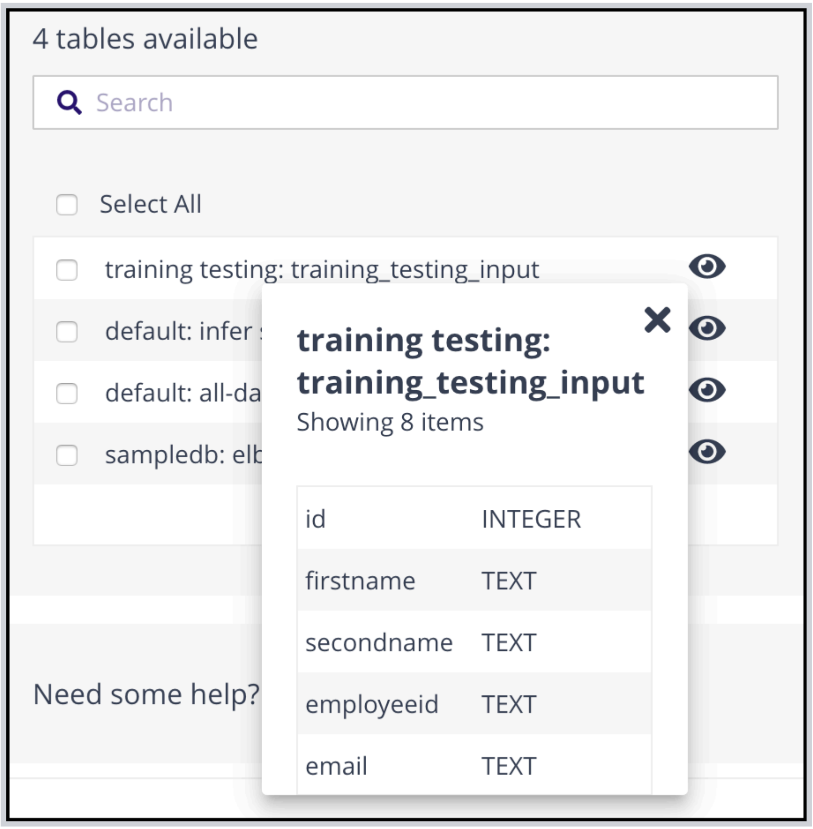
You can import and create Schemas from other Schema, Policy, or Job JSON configuration files that were exported from the Privitar Platform. Once the Schema is imported, you can review the Schema and make any changes to the tables and columns by editing the existing tables or adding new tables. To import a Schema directly into Privitar without making any changes to the Schema, see Importing an Exported Schema.

To create a Schema from JSON, follow these steps:

1. Select **Schemas** from the Navigation sidebar. The **Schemas** page is displayed.
2. Click on **Create New Schema**. The **New Schema** window is displayed.
3. Enter a name for the new Schema in the **Name** field.
4. Select **Import from JSON** from the **Import tables** list box. The **Import from JSON** window is displayed.
5. Click on **Browse** to select a JSON file to upload, or drag and drop a JSON file directly on to the **Import Schema File** box.

   The JSON file is read and the **Import from JSON** window is updated with the tables that are included in the imported Schema. The tables are shown in the left-hand pane.

   Use the *Eye* icon to the right of a table to inspect the contents of a table. For example:



id	INTEGER
firstname	TEXT
secondname	TEXT
employeeid	TEXT
email	TEXT

6. Pick the tables to import by selecting the checkbox alongside each table. The selected table is added to the list of tables to import in the right-hand pane.
7. Click on **Import Tables** to add the selected table definitions to the Schema.

   The tables are imported and the **New Schema** window is updated with the tables that have been imported.

8. Click on a table in the left-hand pane to preview its definition on the right, and to make any edits to any of the columns included in the table. For more information about the editing actions and how to finalize the Schema definition, see Adding Tables and Columns to a Schema.
9. Click on **Save** to save the new Schema.

   The new Schema is added to the list of Schemas on the **Schemas** page.

Note

The imported JSON file has to conform to the specific Privitar configuration format. It is not possible to import files from previous versions of the Privitar Platform.

4.2.6. Creating a Schema from CSV

A Schema can be created by importing definitions from a CSV file stored locally. This option is particularly useful when you want to create a Schema very quickly to try out the functionality of Privitar with local test data.

Only one CSV file can be read to create a Schema. The single CSV file will create a Schema with a single table. To create a Schema with multiple tables from multiple CSV files, you need to create a Schema using HDFS. Using HDFS, you can specify a directory that contains multiple CSV files. These files will be read into a Schema with multiple tables. For more information, see Creating a Schema from HDFS.

To create a Schema with a single table from a locally stored CSV file:

1. Select **Schemas** from the Navigation sidebar. The **Schemas** page is displayed.
2. Click on **Create New Schema**. The **New Schema** window is displayed.
3. Enter a name for the new Schema in the **Name** field.
4. Select **Import from CSV (Local)** from the **Import tables** list box. The **Import from CSV (Local)** window is displayed.
5. If required, edit the CSV settings to match the format of the CSV file that will be read to create the Schema.

   The following table describes the CSV settings:

Setting	Description
CSV Delimiter	The default character that is used to separate the fields in a row.
CSV Escape Char	The character that starts an escape sequence. For example, you can escape a quote (using `\"` or `\'`) within a quoted string to include these characters literally without being interpreted as a quote. For example, `'It\'s OK'` is interpreted as `It's OK`. If you use an escape character outside of a quoted context, then it is not interpreted as an escape value but instead appears verbatim.
CSV Quote Char	The character that is used to delimit text strings in the input.

Setting	Description
CSV Timestamp Format	The timestamp format used by Privitar for processing columns containing Date or Timestamp fields. If the tables that you are importing contain Date or Timestamp fields, Privitar will import these tables according to the following default Date and Timestamp formats: • **Date Format**: `yyyy-MM-dd` • **Timestamp Format**: `yyyy-MM-dd'T'HH:mm:ss` Click in the field to change the format. (For more information about the Date and Timestamp formats supported by Privitar, see Date and Timestamp formats.)
Contains Header Row	Check this box if the first row of the CSV file contains a header row. The column names in the header row will be read and used as the names for the columns in the Schema. If the box is not checked, the default column names will be assigned to the columns in the Schema. That is, `_c0.`, `_c1.` and so on.

6. Click on **Browse** link to select a CSV file to upload, or drag and drop a CSV file directly on to the **Source CSV File** field.

   The CSV file is read by Privitar and the **Import from CSV** window is updated with information about the CSV file:

   - One table is available to be imported.
   - The name of the table is derived from the name of the CSV file. For example, if the file is called `text.csv`, the name of the table to be imported will be `text`.
   - Click on the *Eye* icon to inspect the columns in the table.
   - The table is automatically added to the right column as the table to be imported.

7. Click on **Import tables** to add the selected table to the Schema.

   The table is imported and the **New Schema** window is updated with the tables that have been imported.

8. Click on a table in the left-hand pane to preview its definition on the right, and to make any edits to any of the columns included in the table. For more information about the editing actions and how to finalize the Schema definition, see Adding Tables and Columns to a Schema.

9. Click on **Save** to save the new Schema.

   The new Schema is added to the list of Schemas on the **Schemas** page.

4.2.7. Specifying a Schema Directly

A Schema can be specified directly in the Privitar user interface. This process involves listing the tables and columns in the dataset.

To creating a Schema directly, follow these steps:

1. Select **Schemas** from the Navigation sidebar. The **Schemas** page is displayed.
2. Click on **Create New Schema**. The **New Schema** window is displayed.

3. Enter a name for the new Schema in the **Name** field.
4. Add Tables and Columns to the Schema. For more information, see Adding Tables and Columns to a Schema.
5. Click on **Save** to save the new Schema.

   The new Schema is added to the list of Schemas on the **Schemas** page.

4.2.8. Creating Hierarchical Schemas

The Privitar Platform supports the processing of *complex types* using the Automation API. For example, data in Avro format or JSON format in Kafka messages. (Complex types are not supported in the user-interface.)

The following table lists the support levels for complex types for each Privitar Job type:

Job Type	Supported	Data Platforms
Batch	Not supported.	
Data Flow	Yes	NiFi, Kafka, Streamsets.
POD	Yes	(Also, Privitar SDK.)

To support these types requires the creation of a **Hierarchical Schema**. Once a Hierarchical Schema has been created, a Policy can be created to describe the required transformations to the primitive data values. The process of creating a Policy is identical to the process for a non-Hierarchical Schema.

The Automation API can be used to create Schemas where the fields are arranged in a hierarchy, which map onto complex types. When de-identifying such data using a Policy, Rules are applied in-place to the primitive components of the complex types, producing a de-identified output that retains the structure.

For example, consider the following Hierarchical Schema defined using JSON syntax:

```
customer: {
        name: "John Smith",
        addresses : [
            {
                address: "4 Cherry Tree Dr, London",
                postal-code: "W12 8PQ"
            },
            {
                address: "18B Willow Road, Liverpool",
                postal-code: "L7 7GH"
            }
        ]
    }
```

To process this input, the Automation API should be used to create a Hierarchical Schema containing the following fields and paths:

- `customer > name`
- `customer > addresses > address`

- `customer > addresses > postal-code`

A Policy can then be defined, using either the user-interface or the Automation API, to describe the required de-identification. For example:

- **(TOKENIZE)** `customer > name`
- **(DROP)** `customer > addresses > address`
- **(CLIP)** `customer > addresses > postal-code`

Finally, applying the Policy using Privitar produces a structured de-identified output:

```
customer: {
        name: "Xjdhgidkkidhg",
        addresses : [
            {
                postal-code: "W12"
            },
            {
                postal-code: "L7"
            }
        ]
    }
```

4.3. Adding Tables and Columns to a Schema

This section describes how to add Tables and Columns to a Schema from the **New Schema** window.

4.3.1. Adding a Table

Add a table to a schema using **Add Table**.

- Specify a name for the table. This is used to refer to the table in the Privitar user interface.

Once the table is created:

- The name of the table can be changed by clicking on the **Edit** icon in the list.
- The table can be deleted by clicking on the **Delete** icon in the list.

4.3.2. Adding a Column

To add columns to a table, follow these steps:

1. Select the table in the list and click on **Add Column**. The **Add Column** dialog box is displayed.
2. Enter a name for the column in the **Name** field.
3. Select a data type for the column from the **Data Type** list box.

   For more information about the data types that are supported by Privitar, see Privitar Data Types. Any data types that are not directly supported in Privitar can be classified as **Other**.

There are a few other considerations to be aware of when selecting the data types to be used:

- Privitar strictly validates data types as per the Schema. If the defined Privitar Schema does not match the Schema of the input file, errors will occur during processing.
- If the data type is a **Date** or a **Timestamp**, you will also have to add a **CSV Date/Timestamp Format** string to allow correct parsing of date strings. (For more information about Date and Timestamp formats supported by Privitar, see Date and Timestamp formats.) If the input file is not of CSV type (for example, an Avro or Parquet file), this setting will be ignored.

4. Select the **Optional** check box if you want the column to be optional. This means that this column is not present in all records that use this Schema.

   Schemas containing optional fields can only be used in Data Flow Jobs and Privitar On Demand Jobs, not in Batch Jobs.
5. Click on **Add** to add the new column. If you have more columns to add, click on **Add Another** instead. This adds the new column, but keeps the **Add Column** dialog box open so that you can add more columns.
6. Click on **Save** to save the Schema.

4.4. Importing an Exported Schema

You can import and create Schemas from other Schema, Policy, or Job configuration files that were exported from the Privitar Platform.

To import a Schema, follow these steps:

1. Select **Schemas** from the Navigation sidebar. The **Schemas** page is displayed.
2. Select **Import Schema** from the **Actions** list box. The **Import Schema** dialog box is displayed.
3. Click on the **browse...** link to select a file to upload, or drag and drop a file directly on to the dialog box.

   The **Import Schema** dialog box is updated to display the Schema that will be imported.
4. Click on **Import**, to import the Schema.

   The file is read and a new Schema is created in Privitar. The import date and time will be appended to the name of the Schema.

Note

The imported JSON file has to conform to the specific Privitar configuration format. It is not possible to import files from previous versions of the Privitar Platform.

5. Specifying Policies

This section describes how to specify Policies in Privitar.

5.1. What is a Policy?

A Privitar Policy represents a transformation of input conforming to a specified Schema into an output with privacy-preserving de-identification applied.

The Policy specifies, for each column in the input, the de-identification process (if any) that should be applied to that column. The output is new tables in a new location that consist of the transformed columns.

The techniques used by Privitar to de-identify data are:

- Masking
- Tokenization
- Generalization

Each of these techniques are treated as parts of a Privitar Policy.

After it has been defined, the execution of the Policy against its dataset is controlled by a Job.

5.1.1. Masking

Masking refers to a process where sensitive values are removed/redacted (for example, clipped or substituted) or obscured (for example, perturbed).

Masking can be applied with Batch (Hadoop), Data Flow and Privitar on Demand Jobs, but certain Rule types are only available in specific Job execution modes.

5.1.2. Tokenization

Tokenization refers to replacing raw values with generated tokens.

Privitar supports two types of Tokenization:

- **Random Tokenization**; raw values are replaced with randomly generated tokens.
- **Derived Tokenization**; raw values are replaced with a token derived from the encrypted value of the input.

The process of tokenization may be consistent, meaning that the same input value always results in the same token whenever it is processed by the same Rule within the same Protected Data Domain. This is known as **Preserve Data Consistency** and is an option that is available with rules that perform tokenization.

This is important when considering relationships in the data. Consistent tokenization is required if a keyed relationship is to be maintained, because the same value (that is, the same random token) must be present in several de-identified tables. For Rules that are configured to mask or tokenize consistently, you might optionally allow for unmasking of original values. That is, enable re-identication of data that was de-identified through that Rule.

If tokenization is not consistent, a new token will be created for each occurrence of a value. This means there will be multiple tokens for the same input value if it occurs more than once in the data.

Tokenization can be applied with Batch (Hadoop), Data Flow and Privitar on Demand Jobs, but certain Rule types are only available in specific Job execution modes.

5.1.3. Generalization

Generalization refers to the process of blurring sensitive values in input data to mitigate re-identification risks and defend against linkage attacks.

Privitar supports two types of generalization processing:

- Manual generalization
- Automatic generalization

Manual generalization without k-Anonymity constraints can be applied with Batch (Hadoop), Data Flow and Privitar on Demand Jobs. Manual Generalization with k-Anonymity enabled and Automatic generalization can only be applied with Batch Jobs, due to the requirement to check for the mandatory k-Anonymity constraints across all records.

5.2. Creating a Policy

To create a Policy:

1. Select **Policies** from the Navigation sidebar. The Policies page is displayed.
2. Click on **Create New Policy**. The **New Policy** page is displayed.
3. Enter a name for the Policy in the **Name** edit box. This name is used to refer to the Policy in the Privitar user interface.
4. Select an input Schema from the **Schema** list box. The columns contained in the Schema will be displayed in the **Masking** tab.
   - The chosen Schema determines the format of data that the Policy will accept.
   - The Schema cannot be changed once it has been selected. To create a Policy for a different Schema, a new Policy must be created.
   - If the Schema is edited such that it is incompatible with the Policy (for example, if a new column is added), then the Policy will revert to its Draft state (see below) and cannot be used until amended to be compatible with its Schema
5. In the **Masking** tab, set the masking rules to apply to each column in the Schema. The masking rules are used to specify how the Policy will transform the input data. For more information, see Setting a Masking Rule.
6. Use the **Manual Generalization** and **Automatic Generalization** tabs to determine how the Policy will blur input data to mitigate re-identification attacks. For more information, see Manual Generalization Strategy Types and Automatic Generalization Strategy Types.

7. Click **Save** to save the Policy.

   Note

   Until the Policy is completely configured, it will be in a Draft state and cannot be used in a Job. A Policy is only completely configured when all columns in the Schema have had a Rule applied to them.

5.2.1. Draft State

Each column of each of the tables in the chosen Schema must have a masking rule applied.

As long as there are columns that do not have a rule set, the Policy will be in the Draft state. The Policy cannot be used to transform data in this state.

To remove the Draft state, apply a masking rule to each column. When the Policy is saved it will automatically be removed from Draft state.

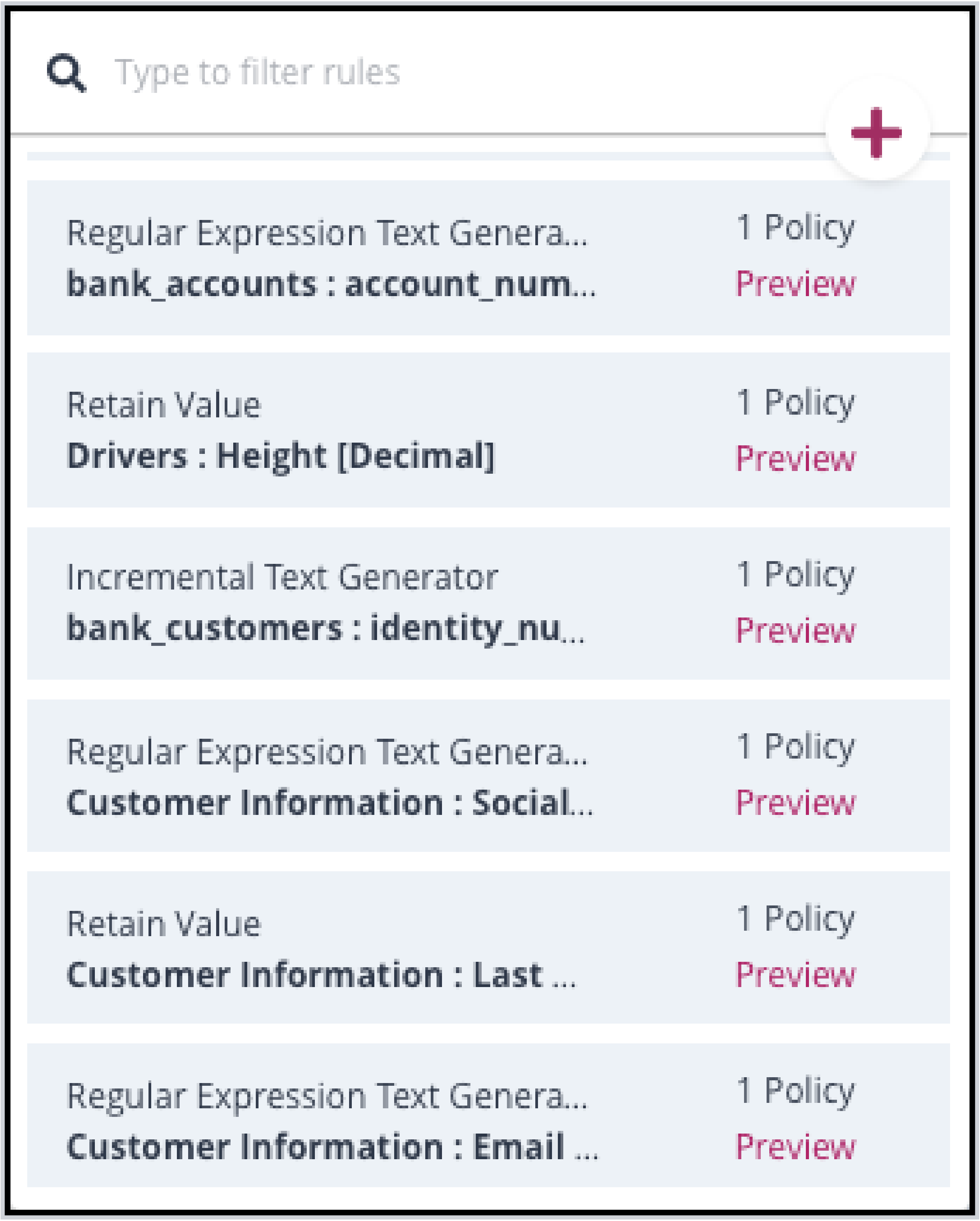
5.2.2. Importing an Exported Policy

A Policy can also be created by importing a previously exported file. See, Importing an Exported Policy.

5.3. Setting a Masking Rule

To set an already defined Rule from the Rules library to a Policy:

1. Select **Policies** from the Navigation sidebar. The **Policies** page is displayed showing all the Policies that have been created.
2. Select a Policy by clicking on the Policy name in the **Name** column. The **Edit Policy** page is displayed showing all the columns together with any rules that have been assigned to the columns.
3. Select **Assign Rule** to select a new masking Rule for a specific column.
   - You can also select multiple columns by clicking on the check boxes alongside each column. To set a masking rule for all the selected columns, click **Assign Rule to Selection**.
   - You can filter columns by entering a text string in the Search box.
     Any Column names that match the text string entered will be displayed. Delete the string from the Search box to view all the Columns again. For large datasets, use the Filter button on the right of the Search box to run filters on all the columns in the dataset. (See ???.)
4. For the selected column(s), the **Rules** dialog box is displayed showing all the rules that have been defined in your Privitar environment:



Type to filter rules

+

Regular Expression Text Genera...
bank_accounts : account_num...
1 Policy
Preview

Retain Value
Drivers : Height [Decimal]
1 Policy
Preview

Incremental Text Generator
bank_customers : identity_nu...
1 Policy
Preview

Regular Expression Text Genera...
Customer Information : Social...
1 Policy
Preview

Retain Value
Customer Information : Last ...
1 Policy
Preview

Regular Expression Text Genera...
Customer Information : Email ...
1 Policy
Preview

- You can filter rules by their rule name, by entering a text string in the search box.
- You can click on the **+** symbol to create a new rule. (For more information, see Using the Rules Library.
- An asterisk * symbol next to Policy indicates that the Rule is included in the current policy.

- You can click on **Preview** or on the rule name to see the details of the rule. If the rule is only used by the current policy you can edit the rule. If the rule is used by at least one other Policy, it is known as a **Shared Rule** and you are only able to view the rule. Shared Rules can only be edited via the Rules Library. (For more information, see Using the Rules Library.)

5. Click on a Rule from the list to apply the rule to the selected column(s).

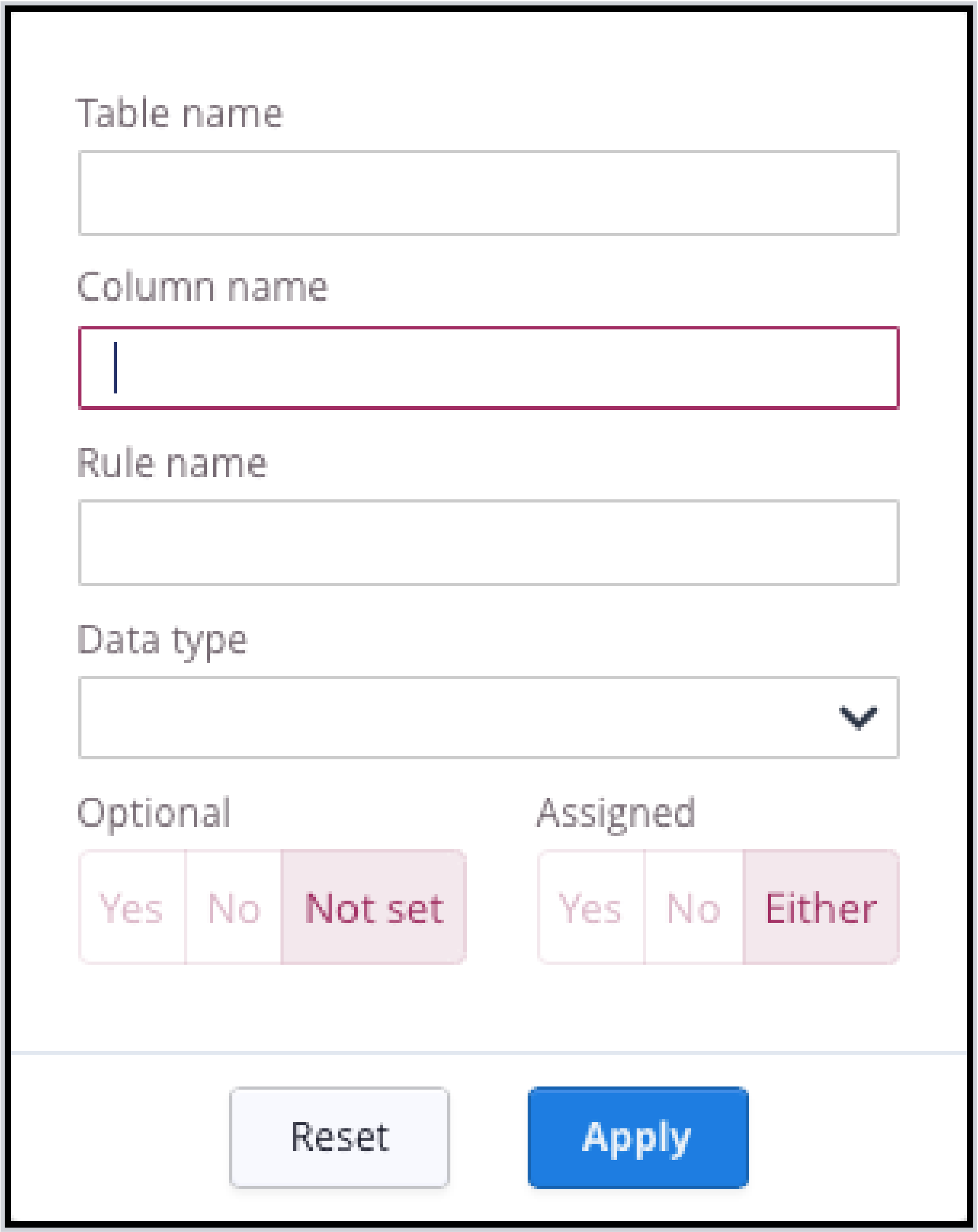
This process should be repeated until all columns are mapped to a rule.

Setting the Masking Rules is the only required step when creating a Policy.
Optionally, Manual generalization and Automatic generalization can also be specified. For more information, see Manual Generalization Strategy Types and Automatic Generalization Strategy Types.

5.3.1. Filtering Columns

For larger datasets with multiple columns, use the Filter dialog box. This dialog box provides many filtering options that you can use to select a subset of the dataset:

1. Click on the Filter icon to display the Filter dialog box:



You can apply a filter to the dataset using any of the fields or options shown in the dialog box.

2. Enter text strings to use as filters in any of the text fields, or choose a setting for **Optional** and **Assigned** columns.
3. Click **Apply** to apply the filter. The results of the filter are displayed in the Policy page.

5.4. What is k-anonymity?

In order to understand the risks of linkage attacks and re-identification, the following definitions are useful:

- A direct identifier in a dataset is a single column that contains an unambiguous reference to an individual. For example, a social security number.
- A quasi-identifier in a dataset is a set of columns that, taken together, can be used to reference individuals. In a dataset, given a value for each of the quasi-identifier columns, it may be possible to identify a single individual, which is undesirable in a privacy context.

The Privitar Platform can mitigate this by ensuring that, after processing, no fewer than a **Minimum Cluster Size** (also called *k*) of records have any particular combination of values for the quasi-identifier columns. If quasi-identifier values corresponding to a specific individual are known, it is no longer possible to identify which row in the cluster of records with those quasi-identifier values belongs to that individual; the most specific that can be achieved is a set of rows of at least the **Minimum Cluster Size**. This is known as *k-anonymity*.

Privitar provides two approaches to achieving k-anonymity:

- Manual Generalization
- Automatic Generalization

5.4.1. Manual Generalization

Manual Generalization is an approach to mitigating re-identification attacks by blurring data. This means that an exact value is replaced by a less precise value, either by binning values in the case of numbers and dates, or truncating values in the case of text. The Privitar Platform allows a binning or truncation strategy to be specified as part of the Policy.

Minimum Cluster Size

When the **Minimum Cluster Size** option is enabled, Privitar will drop clusters of rows that are smaller than the minimum size. Alternatively it is possible to use automatic generalization to automatically adjust the generalization parameters which avoid dropping rows.

5.4.2. Automatic Generalization

Automatic Generalization is a second generalization approach supported by the Privitar Platform to mitigate re-identification risks. Like Manual Generalization, it blurs a selected set of attributes in order to defend against linkage attacks and re-identification of individuals in data sets.

In this context, blurring refers to transformations that remove detail from values, such as binning, preserving only the most frequent values for each column, or combining granular records into broader hierarchical categories.

5.4.3. Comparison of Manual and Automatic Generalization

When Automatic Generalization is configured, an algorithm is used to determine a data transformation that achieves k-anonymity and preserves data utility without dropping any records. It dynamically determines an appropriate degree of blurring for each record depending on the input data and the required level of privacy protection.

In contrast, Manual Generalization acts in a strictly uniform way, blurring all input data to the same degree, and can drop rows as required to achieve k-anonymity.

Manual Generalization may be the right choice if, for example, a fixed blurring is required in the output data, such as a specific numeric binning or rounding a decimal value to a given precision.

5.4.4. Job types compatibility

Enforcing k-anonymity constraints requires Privitar to process the dataset as a whole unit. For this reason Policies with Automatic Generalization strategies can only be applied with Batch Jobs.

5.5. Specifying Manual Generalization

Before specifying Manual Generalization, note the following points:

- In order to add generalization for a column, it **must** first be added to the **Masking** tab. Generalization processes the output value of the masking step. Usually, a Retain masking Rule is applied.
- It is not required to have a generalization strategy for **every** column.

There are two areas that need to be specified for Manual Generalization:

- Generalization Strategy.
- Minimum Cluster Size. (This is an optional setting.)

To specify a Generalization Strategy:

1. Select **Policies** from the Navigation sidebar. The **Policies** page is displayed showing all the Policies that have been created.
2. Select a Policy by clicking on the Policy name in the **Name** column. The **Edit Policy** page is displayed showing all the columns together with any rules that have been assigned to the columns.
3. Select the **Manual Generalization** tab.
4. Click on **Add Generalization Strategy** to specify the generalization for a group of columns.
   - Select the columns to include in the strategy by clicking on the right chevron (>) to move the selected column to the window on the right of the dialog box. (You can use the **Select Columns** filter to display a subset of the columns.) All columns in the selection will be generalized in the same way.
   - Enter a name for the Generalization Strategy in the **Name** box. This is used to refer to the strategy in the Privitar user interface.

- Select a Generalization Strategy Type from the **Generalization Strategy Type** list box. (For more information, about the strategy types that are available, see Manual Generalization Strategy Types.) Privitar supports different strategy types based on the type of value being processed.
- Click **Save** to save the strategy.

5. Repeat the process as required to add generalization strategies for other columns.
6. Click Save to save the Policy, (or continue if you wish to specify a minimum cluster size).

To specify a Minimum Cluster Size:

1. Click the **ON** slider to enable the option.
2. Enter a value for **k** in the **Minimum Cluster Size** box.
3. Click **Select Columns** to add the columns to be included in the quasi-identifier analysis.
4. Click **Save** to save the Policy.

For more information about quasi-identifiers and minimum cluster sizes, see What is k-anonymity?.

5.5.1. Manual Generalization Strategy Types

The Strategy Types that are available for Manual Generalization are described in the following table:

Strategy Type	Description	Data Types
Discrete interval	The exact number is placed into a bin of configurable size, and replaced with either the midpoint of the interval or the interval range. The bin extents are discrete. The **Interval Alignment** setting determines the size of the bins. The **Output Midpoint Value/Output Range as Start:End** setting determines whether the output: • Retains the input data type and uses the midpoint of the bin for all values falling into it. • Is a text/string describing the range. When the **Output Range as Start:End** is chosen, the output is a formatted string such as: `[2..5]`.	Byte Short Integer Long

Strategy Type	Description	Data Types
Continuous interval	The exact number is placed into a bin of configurable size, and replaced with either the midpoint of the interval or the interval range. The bin extents are continuous. The **Interval Alignment** field determines the size of the bins. The **Output Midpoint Value/Output Range as Start:End** setting determines whether the output: • Retains the input data type and uses the midpoint of the bin for all values falling into it. • Is a text/string describing the range. When the **Output Range as Start:End** is chosen, the output is a formatted string such as: `[2.5..5.3]`.	Double Float
Reformat Date	The exact date or timestamp is binned and replaced with an output that reduces precision. For example, given an input format of `dd/MM/yyyy`, binning of dates to a monthly precision is achieved by setting **Output Date Format** to `MM/yyyy`. On CSV, the date or timestamp will be re-formatted and truncated as specified. On Avro, Hive and Parquet, the data type is retained after processing and all output values within a bin are set to the beginning of the corresponding time period. In the aforementioned example you might get `01/01/2017`, `01/02/2017`, `01/01/2017` and so on as output (in the natively encoded format). For more information about the date formats used for CSV files, see Date and Timestamp formats.	Date Timestamp
Clip Text	Detail is removed from the value by replacing it with a substring. For example, a string representing a card number can be truncated to its first four digits.	Text

5.6. Specifying Automatic Generalization

Before specifying Automatic Generalization, note the following points:

- In order to add generalization for a column, it **must** first be added to the **Masking** tab. Generalization processes the output value of the masking step. Usually, a Retain masking Rule is applied.
- It is not required to have a generalization strategy for **every** column.

To specify Automatic Generalization:

1. Select **Policies** from the Navigation sidebar. The **Policies** page is displayed showing all the Policies that have been created.

2. Select a Policy by clicking on the Policy name in the **Name** column. The **Edit Policy** page is displayed showing all the columns together with any rules that have been assigned to the columns.
3. Select the **Automatic Generalization** tab.
4. Click the **ON** slider to enable the option.
5. Enter a value for **k** in the **Minimum Cluster Size** box.
6. Select **Add Column** to add a column to be included in the Automatic Generalization strategy.

   For each column, configure the Automatic Generalization strategy for that column. The choice of strategy determines how the Automatic Generalization algorithm blurs values for that column. The extent to which the values in each column are blurred depends on the input data and will be automatically determined by the Automatic Generalization algorithm. There are different generalization strategies applicable for different Privitar data types.

   For a complete definition of the available strategies and which Privitar data types they can be applied to, see Automatic Generalization Strategy Types.

   A column can also be also be configured as **Do not generalize**. If you choose this option, you may also set this as a sensitive column by selecting the **Sensitive** check box. Specifying a column as Sensitive ensures that for each cluster of rows with the same quasi-identifier values, there is a diverse mix of values for the sensitive columns. For more information, see Automatic Generalization Advanced Settings.
7. Select the **Priority** check box if you want the column to be one of the two priority columns in the generalization strategy. All columns that are of interest for any intended analysis or have a particular structure that should be preserved can be selected as a priority column.

   Note

   A maximum of two priority columns can be selected at the same time.
8. Click **Save** to save the Policy.

For more information about quasi-identifiers and minimum cluster sizes, see What is k-anonymity?.

5.6.1. Automatic Generalization Strategy Types

The Strategy Types that are available for Automatic Generalization are given in the following table:

Strategy Type	Description	Data Types
Preserve All Values	This strategy will preserve all the values in the selected column, while still using it as a quasi-identifier in the calculation of k. For example, this rule can be used to preserve the granularity of a column which has already gone through a manual generalization step. Selecting this strategy will cause the generalization job to fail when it is not possible to preserve all the values while providing the k-Anonymity and l-diversity guarantees.	All (except `Other`)
Preserve Most Frequent Values	Returns either the original attribute value (if the frequency of the quasi-identifier value is greater than or equal to the defined minimum cluster size), or otherwise an **All** value to generalize the original value. Having **Partially Combine Small Groups** checked allows the algorithm to form groups of the minimum cluster size by combining smaller clusters together. For example, if the minimum cluster size is 100, it may combine three groups with different values of size 50, 40 and 30 into a composite cluster with size 120. The quasi-identifier value of this cluster will be, [value 1 or value 2 or value 3]. Null values can be either considered as a category on their own by selecting **Nulls are a distinct category**, or bucketed together with the infrequent values group by selecting **Group nulls with infrequent values**. The infrequent values group name can be customised by changing the **Infrequent value string**. The root group string can also be customised via the **Root node string** option. For more information about the options available, see Merged Null Category Handling.	`Text`

Strategy Type	Description	Data Types

Categorical Generalization	Generalizes values into more general categories according to a user-specified hierarchy. The hierarchy is uploaded in JSON format.
	
	The definition must contain a top-level 'any' category that, if necessary during generalization, may represent any possible value. The top-level category then contains nested children categories, which may themselves be further nested as deeply as required.
	
	The JSON syntax is:
	
	`{`
	`   "value": "`*`category-name`*`",`
	`   "children": [  `*`list-of-children`*` ],`
	`   "nodetype": "`*`node type`*`"`
	`}`
	
	Child categories are represented with the same syntax. Specifying a node type is optional (the default is TEXT), and can have the following values:
	
	• `TEXT`: a standard category.
	• `NULL`: this category will be used to group null values.
	• `CATCHALL`: this category will be used as a container for all unmatched values.
	
	It is only possible to specify one `NULL` and one `CATCHALL` node in each JSON file. For example:
	
	`{`
	`    "value":"any", "children":[`
	`        {`
	`            "value":"Scotland","children":[`
	`                {"value":"Edinburgh","children":`
	`[]},`
	`                {"value":"Glasgow","children":[]}`
	`            ]`
	`        },`
	`        {`
	`            "value":"Wales","children":[`
	`                {"value":"Cardiff","children":`
	`[]},`
	`                {"value":"Swansea","children":[]}`
	`            ]`
	`        },`
	`        {`
	`            "value":"England","children":[`
	`                {"value":"London","children":[]},`
	`                {"value":"Manchester","children":`
	`[]}`
	`            ]`
	`        }`
	`    ]`
	`}`
		`Text`

Strategy Type	Description	Data Types
	Note Selecting this strategy might cause the generalization job to fail if unexpected values are found in the input dataset, unless a `CATCHALL` node is explicitly defined in the JSON file.	

Discrete Interval Generalization	Bins the input values such that each bin corresponding to a value range comprises at least k records. The algorithm starts with a bin spanning the maximum range of column values, and iteratively subdivides to achieve a finer binning. The procedure is repeated until either the bins would contain fewer than k records or the range of the bin would be smaller than the specified **Interval Alignment**. The **Split Factor** can be adjusted to determine how many subdivisions are created for each bin at each iteration step
	
	The **Split on Median** setting changes the way each bin is split into smaller categories. If the strategy is set to split on median the algorithm calculates the percentiles of each bin and forms a subdivision of this bin into the resulting value ranges. The number and size of the percentiles depends on the **Split Factor**. A split factor of 3 would result in splitting each existing category at its 33rd and 66th percentiles, whereas a split factor of 2 would result in a single split at the 50th percentile.
	
	The **Output Mean** option controls the way this column is written to the output file. If checked, the mean of the values in a bin will be returned and the input data type retained. Otherwise, the range of the values is returned as a formatted string/text such as: `[1..4]`.
		`Byte`
		
		`Short`
		
		`Integer`
		
		`Long`

Strategy Type	Description	Data Types

Continuous Interval Generalization	Bins the input values such that each bin corresponding to a value range comprises at least k records. The algorithm starts with a bin spanning the maximum range of column values, and iteratively subdivides to achieve a finer binning. The procedure is repeated until either the bins would contain fewer than k records or the range of the bin would be smaller than the specified **Interval Alignment**. The **Split Factor** can be adjusted to determine how many subdivisions are created for each bin at each iteration step.
	
	The **Split on Median** setting changes the way each bin is split into smaller categories. If the strategy is set to split on median the algorithm calculates the percentiles of each bin and forms a subdivision of this bin into the resulting value ranges. The number and size of the percentiles depends on the **Split Factor**. A split factor of 3 would result in splitting each existing category at its 33rd and 66th percentiles, whereas a split factor of 2 would result in a single split at the 50th percentile.
	
	The **Output Mean** option controls the way this column is written to the output file. If checked, the mean of the values in a bin will be returned and the input data type retained. Otherwise, the range of the values is returned as a formatted string/text such as: `[3.1..23.0]`.
		`Double`
		
		`Float`

Date Generalization	Clusters dates or timestamps based on standard time intervals. The *Date Generalization* strategy will generalize the original dates up to the interval required in order to achieve k-anonymity, reducing the temporal resolution. The returned buckets will be either of size 10 years, 1 year, 1 month or 1 day depending on the range of dates spanned by the column.
	
	On CSV files, the Date Format determines the output format (for Avro, Parquet and Hive files the native encoding will be retained). It accepts a format string as defined in Date and Timestamp formats.
		`Date`
		
		`Timestamp`

Numeric Date Generalization	Treats dates and timestamps like numbers by converting them to the number of seconds since January 1st, 1970, 00:00:00 UTC. The resulting values are handled in the same way as numerical columns as described in Date Generalization.	`Date`
		
		`Timestamp`

5.6.2. Automatic Generalization Advanced Settings

Automatic Generalization with a minimum cluster size of k ensures that, for any combination of quasi-identifiers that exists in the data, the corresponding cluster of records will be at least k records in size. This characteristic of the output data gives a resistance to re-identification through linkage attacks. Informally, each individual is protected from being re-identified by being *hidden in the crowd*.

However, there remains a potential vulnerability in k-anonymous data: for columns which contain sensitive information, such as medical diagnoses, salaries or debts, there may not be sufficient diversity. This means that, given a cluster matching a combination of quasi-identifiers, there are very few—or only one—distinct value(s) in the sensitive column. This may constitute a leak of sensitive information because even if an adversary is prevented from knowing which exact record corresponds to an individual, he nonetheless learns the sensitive attribute. Hiding in the crowd is insufficient if the whole crowd has the same secret value.

In Privitar, specifying an **L-diversity** value of 2 or more mitigates this issue for the specified sensitive columns. For each of these columns the algorithm will ensure there is at least L different values in every cluster. In this way, the vulnerability of the k-anonymous data discussed above is eliminated.

When L-diversity is enabled, the process of generalization might introduce an unbalanced distribution of sensitive values. The **C-Ratio** setting ensures that each of the L values are reasonably balanced. It defines that the combined size of all but the largest L-1 categories must be larger than the size of the largest category divided by the C-Ratio. The effect of this is to ensure that as well as there being L categories, the largest category does not dominate. To understand why this balance is desirable, consider the possibility that a group has a sensitive Boolean column, and the data show one 'Yes' and 99,999 'No's. This very unbalanced ratio allows an adversary to deduce with 99.999% probability that an individual in that group has value "No".

The **Bins** setting affects the way numeric variables are processed when selected as sensitive columns. Numeric values are first binned into the specified number of bins, then L-diversity is applied as above, with each bin being treated as a sensitive category.

5.6.3. Merged Null Category Handling

Merged Null Category Handling controls how records with a NULL value are treated when the resulting cluster has fewer than the minimum cluster size of records and therefore does not satisfy k-anonymity. It is necessary to specify how these clusters are handled.

The general approach is to merge the NULL category with another category so that the resulting cluster has a size larger than the minimum cluster size. The options provided set the manner in which the NULL category is processed and what the output format of the merged cluster is.

The options provided are described in the table below:

Handling	Description
Output both null and non-null values in the category as the non-null category value.	If the null category is to be merged with a category X, output the value X for all records in the cluster regardless of whether the original value is X or NULL. If the null category is to be merged with categories X and Y, output the value X|Y for all records in the cluster regardless of whether the original value is X,Y or NULL.

Handling	Description
Output both null and non-null values in the category as null.	If the null category is to be merged with a category X, output NULL for all records in the cluster regardless of whether the original value is X or NULL. If the null category is to be merged with categories X and Y, output the value NULL for all records in the cluster regardless of whether the original value is X,Y or NULL.
Output both null and non-null values in the category as 'category value|null'.	If the null category is to be merged with a category X, output X| NULL for all records in the cluster regardless of whether the original value is X or NULL. If the null category is to be merged with categories X and Y, output the value X|Y|NULL for all records in the cluster regardless of whether the original value is X,Y or NULL.
Drop rows containing a null value for the variable from the output.	Rows containing NULL in the column are dropped. This means that there will be no category smaller than the minimum cluster size containing NULL.
Output null values as null even though this will violate the minimum cluster size.	Rows containing NULL are retained regardless of the minimum cluster size. This means that there may be a category smaller than the minimum cluster size that contains all records with NULL.

5.7. Importing an Exported Policy

The Privitar Platform can create Policies by importing previously exported Policy files. To import a Policy:

1. Select **Policies** from the Navigation sidebar. The Policies page is displayed.
2. Select **Import** from the **Actions** list box. The **Import Policy** dialog box is displayed.
3. Click on the **browse...** link to select a Policy file to upload, or drag and drop a file directly on to the dialog box.

   Policies can be imported by uploading a Policy or a Job file in JSON format.

   The **Import Policy** dialog box is updated with details about the imported Policy.
4. Review the name of the Schema and Policy to be imported. There are two options for imprting the Policy:
   - Select **Import the included schema** to import the Schema described in the file as a new Privitar Schema
   - Select **Link with existing schema** to re-use a Schema definition already in Privitar. (This option is only available if at least one compatible Schema is stored in Privitar.)
5. Click on **Import** to proceed.

   Privitar will look in the Rules Library for the Rules referenced in the file:
   - If a compatible match (that is, a Rule with the same name and definition) is found, the existing rule will be re-used by the imported policy.
   - If correspondingly named Rules exist, but their definitions are different from the definitions in the file, the conflicting rules will be listed in the **Import** dialog box. The following actions are available to resolve the conflicts:
     - Select **Resolve by creating new Rules** to import the Rules from the file using new names.

    - Select **Resolve by updating the Rules Library** to overwrite the definition of the existing Rules in the Rules Library. Note that this operation may produce unexpected results in existing active Protected Data Domains that are using the Rules, including those owned by other Teams.
    - Select **Resolve by using the current Rules from the Rules Library** to use existing Rules and ignore the changes in the file.
6. Click on **Import** to complete the import of the Policy.

Note

The imported JSON file has to conform to the specific Privitar configuration Policy format. It is not possible to import files from previous versions of Privitar. For more information about importing or exporting other Privitar objects, see Importing and Exporting Configurations.

5.8. Masking Rules

Masking Rules apply a de-identification transformation to each of the columns in the input Schema. The types of transformations supported, include:

- Passing through a column unchanged.
- Removing a column entirely.
- Generating artificial token values based on user-specified patterns.
- Masking data values by truncation (clipping), substitution or encryption.

To decide which behavior is most appropriate, consider the input Schema and determine:

- Which columns are direct identifiers.
- Which columns are quasi-identifiers or indirect identifiers that may allow linkage attacks when combined with data from other datasets.
- Which columns are sensitive (to your organization or the data subjects in the dataset).
- What are the use cases for the data after de-identification (for example, determining which columns should be consistently masked and potentially require unmasking later on).

This exercise will give a good indication as to where the masking should be applied.

In general it is desirable to remove all identifying information from a dataset. By replacing identifier columns with randomly generated tokens (for example using a Regular Expression Text Generator Rule), some privacy risk is eliminated.

Privitar preserves input data types during and after processing (with the exception of the Encrypt Rule which will always output Text, independently of the input data type), and supports Rule configurations to retain or closely resemble the original data format after masking (for example, credit card numbers, email addresses, dates, IDs).

5.8.1. Using the Rules Library

The Privitar Platform collects all Masking Rules in a shared Library. Unlike other Privitar objects (such as Schemas, Policies and Protected Data Domains), Rules are shared across Teams. That is, Rules can be viewed and used in Policies and Jobs by all Teams. This allows the definition of common Masking Rules that can be used by all users of the Privitar platform in an organization.

Note

Only the Team that originally created a Rule can change it.

For more information about the Rules that are available in Privitar, see Masking Rule Types.

To open the Rules Library, click on **Policies** from the Navigation sidebar, then click on the **Rules** tab. A list of all the Rules that have been created are displayed.

Creating a Rule

Rules can be created directly from the Rules Library, or while defining a Policy.

To create a Rule from the Rules library:

1. Click on **Create New Rule.**

   The **Add Masking Rule** window is displayed.
2. Enter a name for the rule in the **Name** field. This is the name that will be used to refer to the rule in Privitar, so enter a name that is meaningful.
3. Select the rule to apply from the **Mask Type** list box.

   For some of the rules you select, there will be additional options or fields to complete in one or both of the group boxes:

   - **Masking Behavior**
   - **Tokenization Behavior**

   For more information about the options associated with Rules, see Masking Rule Types.
4. Click **Save** to save the Rule.

   The new rule will be displayed in the Rule index listing.

Editing a Rule

Rules can be edited directly from the Rules Library.

Note

Editing or deleting a Rule that is applied in Policies, Protected Data Domains and Jobs, including those owned by other Teams, can have unexpected effects and might, for example, invalidate previously generated tokens or data de-identification pipelines using that Rule.

In some scenarios it might be advisable to create a new Rule instead of editing an existing rule.

To edit a rule from the Rules library:

1. Click on the Rule Name.

   The **Update masking rule** window is displayed.
2. Edit the Rule name or settings as required.

   For more information about the options associated with Rules, see Masking Rule Types.
3. Click **Save** to save the changes you have made to the Rule.

   If this Rule is used by Policies, Protected Data Domains or Jobs, carefully check what the impact of the Rule change will be. If the rule you have selected has been run at least once in a job to create a Protected Data Domain, a dialog box is displayed describing the dependency. You can download details by clicking on **Download Full Details**. Only if you are sure you want to proceed, click **Confirm**.

Viewing which Policies use a Rule

It is possible to view the Policies that are using each Rule from the **Used By** column in the Rules Library index.

Clicking on **Policy** in the **Used By** column for any rule listed in the Rules Library index will list the Policies in your Team that use the Rule. Policies can be inspected or edited from this screen by clicking on the Policy name in the **Name** column. To go back to the Rules Library, click the **Rules** tab.

Deleting a Rule

Editing or deleting a Rule that is applied in Policies, Protected Data Domains and Jobs, including those owned by other Teams, can have unexpected effects and might, for example, invalidate previously generated tokens or data de-identification pipelines using that Rule.

To delete a Rule from the Rules library:

1. Click on the Row Select button to select the Rule that you want to delete.
2. Click on **Delete** from the **Actions** list box.

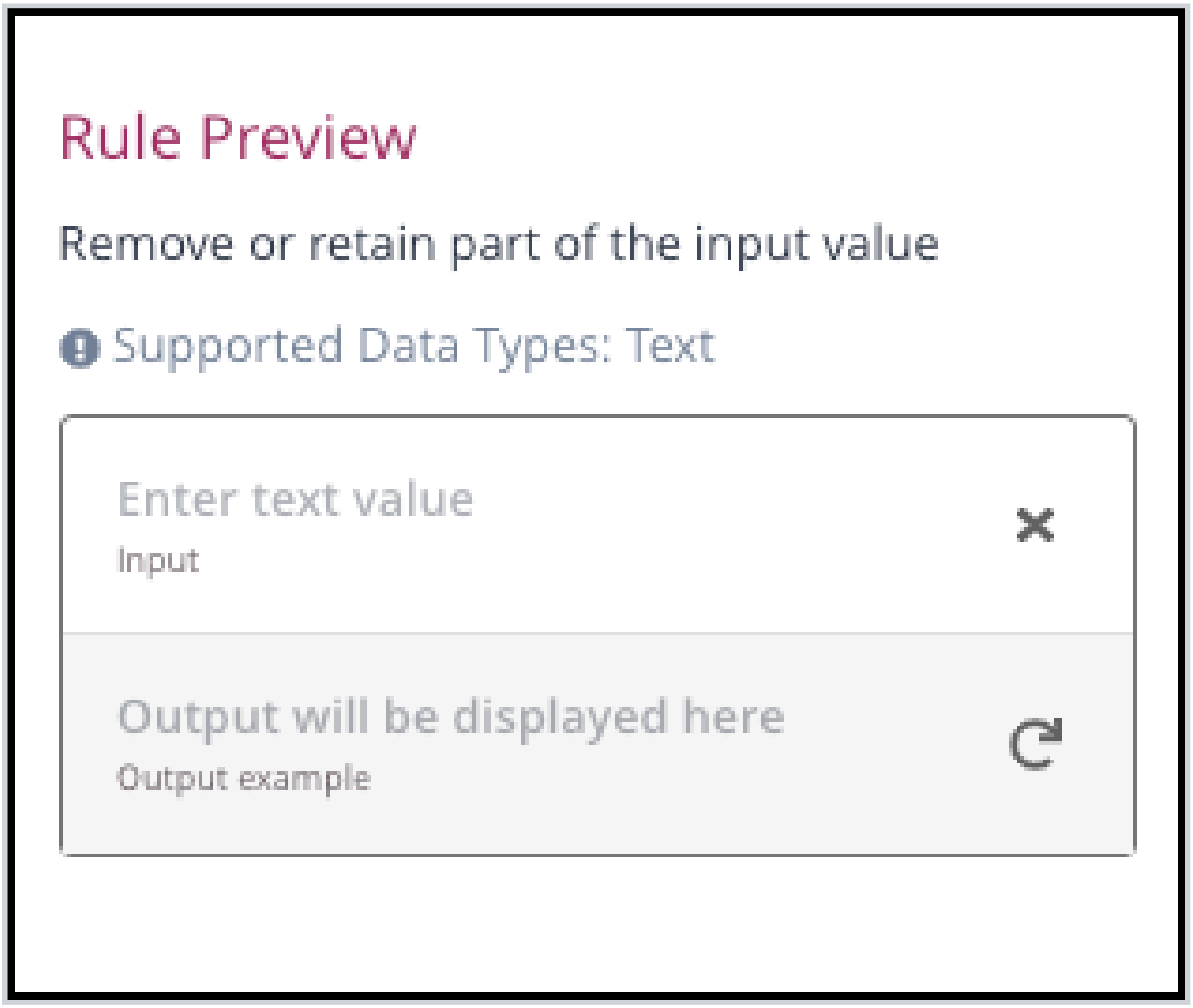
   If this Rule is used by Policies, Protected Data Domains or Jobs, carefully check what the impact of the Rule change will be. If the rule you have selected has been run at least once in a job to create a Protected Data Domain, a dialog box is displayed describing the dependency. You can download details by clicking on **Download Full Details**. Only if you are sure you want to proceed, click **Confirm**.

Deleting a rule is a non-reversible action.

5.8.2. Using Rule Preview

To help you to see how a Masking Rule works on input data, there are some Rules in Privitar that provide a **Rule Preview**. Using the Rule Preview, you can input some example data that is representative of your input dataset and examine what happens to that input data when a particular Rule is applied to that data.

The Rule Preview is provided in a side-panel of the **Add masking rule** window, the **Update masking rule** window, and the **View/Edit rule** window. For example:


Rule Preview

Remove or retain part of the input value

Supported Data Types: Text

Enter text value
Input

Output will be displayed here
Output example

To use the Rule Preview to preview a Rule:

1. Select a Rule from the **Mask Type** list box that supports Rule Preview. (For more information about the Rules that support Rule Preview, see Masking Rule Types Supporting Rule Preview.)
2. If relevant for the selected Rule, enter values for the Rule in the **Masking Behavior** area.
3. In the Rule Preview panel, enter an input value in the **Enter text value** field, ensuring that the value entered is one of the supported data types for the Rule.
4. Click on the *circular arrow* to generate an example of the output that would be created when the Rule is applied to the input value.

   The example output is displayed in the **Output** field.

   Suitable error messages will be displayed if the preview is unsuccessful.
5. If appropriate for the Rule type, you can click on the *circular arrow* again to generate another example output. (This only applies for Rules that generate constant values, such as the **Date Generator** rule.)

The Preview options that are available will differ depending on the Rule that is previewed. Some Rules will require an input value (such as **Clip Text**), while other Rules will not require an input value (such as **Date Generator**).

Some additional points about the Rule preview:

- The preview does not have any built-in values set for Rules that require options to define their behavior. You need to enter the options in the **Masking Behavior** area for the preview to provide an example output.
- The preview does **not** support Consistent Tokenization. That is, the preview will not provide a consistent output value if the same input value is entered multiple times.

5.8.3. Importing an Exported Rule

The Privitar Platform can create Rules by importing previously exported Rules.

To import a Rule:

1. Select **Policies** from the Navigation sidebar. The **Policies** page is displayed.
2. Select the **Rules** tab.
3. Select **Import** from the **Actions** list box. The **Import Rule** dialog box is displayed.
4. Click on the **browse...** link to select a Rule file to upload, or drag and drop a file directly on to the dialog box.

   Note

   The imported JSON file has to conform to the specific Privitar configuration format. It is not possible to import files from previous versions of the Privitar Platform.

5. Review the name of the Rule to be imported and click **Import**.

   If a Rule with the same name already exists, Privitar will provide you with an option to either create a new Rule with a new name (by adding the import date and time to the Rule's name) or to overwrite the existing Rule.

   It is usually strongly recommended to create a new Rule. Updating the existing Rule may produce unexpected results in existing active Protected Data Domains, including those owned by other Teams.

5.8.4. Masking Rule Types

The table below summarizes the types of masking operations available when creating a Policy.

Note

Rules are compatible with certain Privitar Schema Input data types and execution engines/Job types. Not all rules are available for use with every data type and not all rules are available in every execution engine/job type.

Rule Type	Description	Input data types	Job type compatibility
Clip Text	Form an output value by removing or retaining part of the input value. For more information, see Clip Text.	Text	Batch (All types) Data Flow On Demand
Constant Text Value	All values are replaced by the supplied constant value.	Text	Batch (All types) Data Flow On Demand
Generalize Date	The value is replaced by a generalized version of the date, or a constant date if the value is outside of a user-defined date range. For more information, see Generalize Date.	Date Timestamp	Batch (All types) Data Flow On Demand
Date Generator	The value is replaced by a randomly generated date within a user-defined range. For more information, see Date Generator.	Date	Batch (All types) Data Flow On Demand
Drop Column	The column is removed entirely and will not appear in the output.	All	Batch (All types) Data Flow On Demand
Encrypt	Encrypt values into ciphertext with the specified encryption key. The value is encrypted using AES Encryption and the key length used (128-bit, 192-bit or 256-bit) will be determined by the **Key name** that is supplied. (The supplied **Key Name** must refer to one of the Encryption keys that is configured in the Privitar Environment. For more information, see Key Management Environment Configuration).	All (except Other)	Batch (HDFS/Hive only. Not AWS Glue)
Hash Text	The original value is completely replaced by a generated SHA256 salted hash of the value and a pepper (secret salt). By default, the hash will be a base 64 string; however, you can provide a regular expression for the hash output. The output value will always be irreversible. For more information, see Hash Text	Text	Batch (HDFS/Hive only) Data Flow On Demand

Rule Type	Description	Input data types	Job type compatibility
Incremental Text Generator	Generate text values that contain incrementally increasing numbers. Specify the **Start Number** and the **Increment Size** to define how the series of integers progresses. Optionally specify **Field Length** to pad the number to a fixed size. For example with Field Length 5 you get: `00001`. Optionally specify **Field Prefix** and **Field Suffix** to add fixed string values before and after the numeric part. For example setting a prefix of `A-` and suffix of `-B` results in: `A-1-B`.	Text	Batch (All types) Data Flow On Demand
Lookup Text (Upload Lookup File)	The value is replaced by a randomly selected token from a Lookup file that is uploaded locally. For more information, see Lookup Text.	Text	Batch (All types), Data Flow, On Demand
Lookup Text (Lookup from File)	The value is replaced by a randomly selected token from a Lookup file that is stored in HDFS. For more information, see Lookup Text.	Text	Batch (HDFS/Hive only)

Rule Type	Description	Input data types	Job type compatibility
PAN Card Number Generator	The value is replaced by a randomly generated PAN (Permanent Account Number). Check Preserve First 6 and/or Preserve Last 4 to retain part of the input value in the generated output. Note that when using this option, the input values must be numeric of at least length 6, so that the first six and/or last four values can be extracted properly. Check Luhn Valid to ensure the generated value is consistent with the Luhn algorithm. When using the options to preserve first six or last four characters or produce Luhn-valid numbers, the row will be dropped if the input is not long enough to do this. The output formatting is based on the input. Non-digit characters are treated as a pattern into which the generated or preserved digits are inserted. For example, the inputs: • `1234-5678-9012-3456` will produce hyphen-separated output. • `1234 5678 9012 3456` will produce space-separated output. • `1234567890123456` will produce output with no separator.	Text	Batch (All types) Data Flow On Demand
Perturb Currency	This rule perturbs the currency value by adding or subtracting random noise to the input data.	Float Double	Batch (All types) Data Flow On Demand

Rule Type	Description	Input data types	Job type compatibility
Perturb Date	The input date is changed by addition or subtraction of a number of whole days chosen uniformly at random based on a supplied maximum allowed change (**Amplitude**). The rule always changes the date by at least one day. For example, with an amplitude of 3 days, the input date 2017/01/01 will result in one of these outputs with equal probability: • 2016/12/29 • 2016/12/30 • 2016/12/31 • 2017/01/02 • 2017/01/03 • 2017/01/04	Date	Batch (All types) Data Flow On Demand
Perturb Timestamp	The input timestamp is changed by addition or subtraction of a number of milliseconds chosen uniformly at random based on a supplied maximum allowed change (**Amplitude**). The rule always changes the time by at least one millisecond.	Timestamp	Batch (All types) Data Flow On Demand

Rule Type	Description	Input data types	Job type compatibility
Random Number Generator	This rule replaces the input data with a random number between a specified **Minimum Value** and **Maximum Value**. The **Minimum Value** specified is **inclusive**. That is, the value is included in the range of random numbers that can be generated. The Maximum Value specified is **exclusive**. That is, the value is excluded from the range of random numbers that can be generated. **Decimal Places** is an optional configuration (for decimal/floating point type numbers) to specify how many decimal places should be considered in the generated numbers. The range of **Minimum Value** and **Maximum value** as well as the **Decimal Places** precision configurations have to be compatible with the data types that the Rule is applied to.	Byte Short Integer Long Float Double	Batch (All types) Data Flow On Demand
Regular Expression Number Generator	The value is replaced by a randomly generated number that would match the supplied regular expression. For more information, see Regular Expression Number Generator.	Byte Short Integer Long	Batch (All types) Data Flow On Demand
Regular Expression Text Generator	The value is partially or completely replaced by a randomly generated string that would match the supplied regular expression. For more information, see Regular Expression Text Generator	Text	Batch (All types) Data Flow On Demand
Retain Value	The value is passed through unchanged in the output. That is, the original value is retained.	All	Batch (All types) Data Flow On Demand

Rule Type	Description	Input data types	Job type compatibility
SecureLink Encryption	Converts a direct identifier into an encrypted form suitable for later use with the **SecureLink Tokenization** rule. The encryption keys used are referenced from the SecureLink destination Environment configured on the Job that runs this Policy. Choose **Produce reversible tokens** if required to enable re-identification of original values through the SecureLink Unveiler. Note that this option sets a maximum length on input values of 15 characters. Input longer than this limit is rejected when the Job is run. For more information about SecureLink encryption, see About Privitar SecureLink.	Text	On Demand
SecureLink Tokenization	Converts an encrypted identifier produced by the SecureLink Encryption rule into a randomised token described by a regular expression. Choose Permit re-identification of original values if required. Note that choosing this option also requires that the corresponding SecureLink Encryption Rule has its Produce reversible tokens option set. This will allow re-identification of original values through the SecureLink Unveiler. For more information on the use of regular expressions, see Regular Expression Text Generator. For more information about SecureLink encryption, see About Privitar SecureLink.	Text	On Demand
Substitute Text	The input value is replaced with a mapped substitute value as defined in the Rule configuration. For more information, see Substitute Text.	Text	Batch (All types) Data Flow On Demand

Rule Type	Description	Input data types	Job type compatibility
Timestamp Generator	The value is replaced by a randomly generated timestamp. For more information, see Timestamp Generator.	Timestamp	Batch (All types) Data Flow On Demand

5.8.5. Masking Rule Types Supporting Unmasking

The following table lists the Rule Types that are compatible with Unmasking.

Rule types that tokenize the input data in some way are compatible with Unmasking. For Rules that are compatible, the table also indicates the types of tokenization that can be applied to data that is processed by the Rule type.

For more information about Unmasking, see Unmasking De-identified Data.

Rule Type	Unmasking	Tokenization Support	
		Random	Derived
Clip Text	NO		
Constant Text Value	NO		
Date Generator	YES	YES	NO
Drop Column	NO		
Encrypt	YES	YES	NO
Generalize Date	NO	NO	NO
Hash Text	No		
Incremental Text Generator	YES	YES	NO
Lookup Text	YES	YES	NO
PAN Card Number Generator	YES	YES	NO
Perturb Currency	NO		
Perturb Date	NO		
Perturb Timestamp	NO		
Random Number Generator	YES	YES	YES
Regular Expression Text Generator	YES	YES	YES
Regular Expression Number Generator	YES	YES	YES
Retain Value	NO		
SecureLink Encryption	YES	YES	YES
SecureLink Tokenization	YES	YES	NO
Substitute Text	NO		
Timestamp Generator	NO		

Note

There are some usage restrictions for Rules that support Derived Tokenization. For more information, see Masking Rule Types Supporting Derived Tokenization.

5.8.6. Masking Rule Types Supporting Derived Tokenization

When tokens are generated by Rules with **Preserve Data Consistency** enabled, the produced tokens are stored in Token Vaults.

The default tokenization behavior is to use the Token Vault to generate a random token for the values tokenized by a specific Masking rule. This is known as **Random Tokenization**.

However, tokens can also be generated using **Derived Tokenization**. When this setting is enabled, tokens are generated from a value derived from the encrypted value of the input, rather than from the Token Vault. This increases the efficiency of tokenization and is particularly effective for processing high cardinality data.

Derived tokenization is only available for Batch Jobs using HDFS Token Vaults. It is only supported for tokenization using the following Rule types:

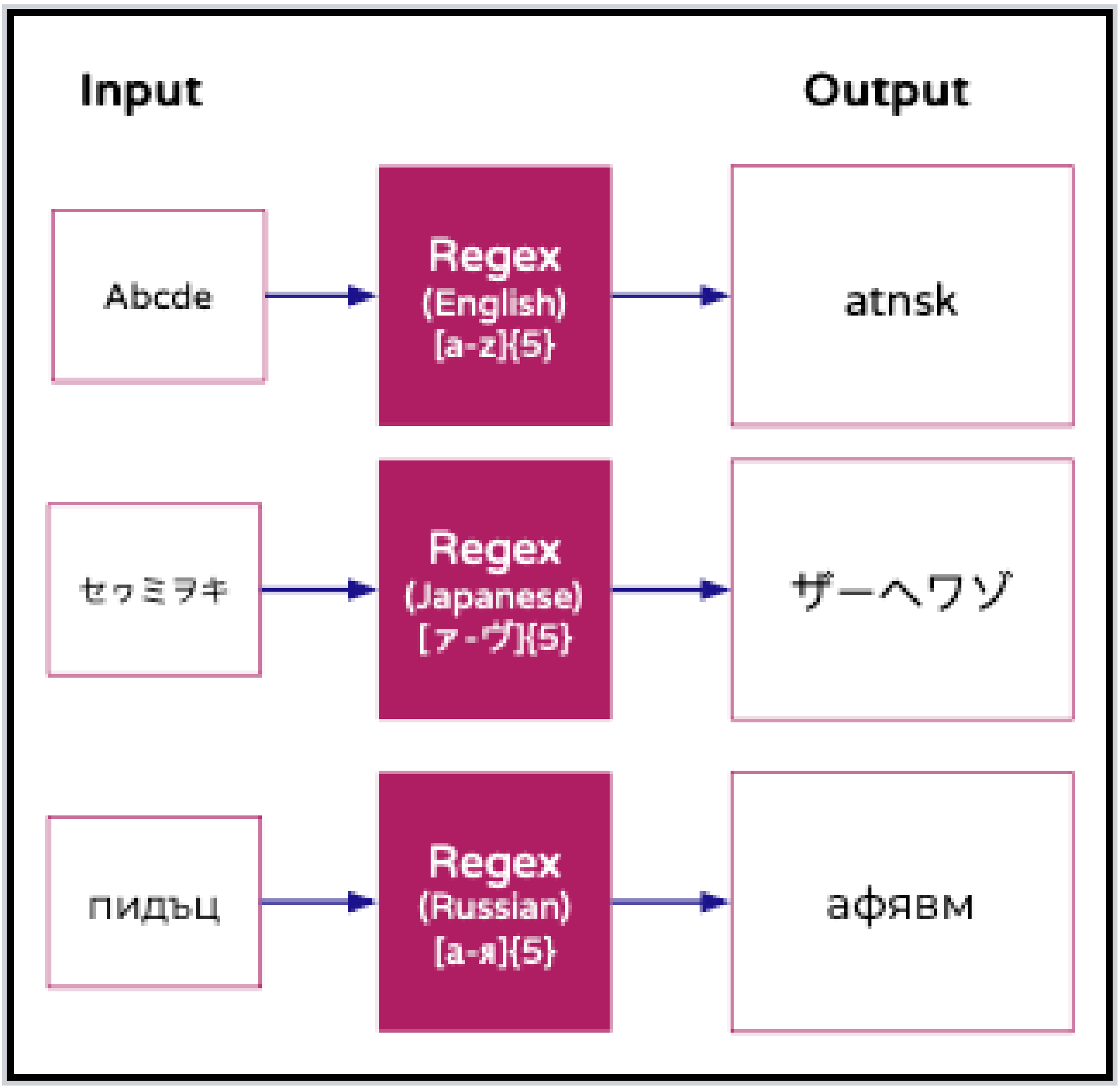
- Regular Expression Text Generator. At least one of the following tokenization settings must be used:
  - The entire input is replaced with a token. That is, the options for this Rule are set to the default values which will replace the entire input with a token:
    - **Start Index** = 0
    - **Length to Replace** = 0
  - A fixed-length regular expression is used. (For example, `[a-z]{8}`, not expressions such as `[a-z]{8,10}`)
- Regular Expression Number Generator.
- Random Number Generator. The input and output must be whole number types.
- Timestamp Generator.

Any other Rule will fall back to using Random tokenization.

For more information about Derived Tokenization and how to configure an HDFS Token Vault to support this feature, see Token Vault Environment Configuration.

5.8.7. Multi-Language Support for Masking Rules

The Privitar platform provides multi-language support for Masking and Unmasking of text in a dataset. For example, the following diagram illustrates how the Regular Expression Text Generator rule (Regex) can be used to mask inputs in three different languages:



For a full list of the languages that are supported, see Privitar Multi-Language Support.

The following sections describe some aspects to be aware of when dealing with datasets in multiple languages.

Consistent support across the data pipeline

It is important to be aware of language support in other technologies that are used as part of any data pipeline; for example, NiFi or Kafka. There may be cases where a language is supported by Privitar but not supported by a component of the data pipeline. This could lead to unexpected behavior when the data is processed by Privitar. It is advised to check the supported languages for all components in the data pipeline that are parsing data.

Consistent character encoding

Privitar treats characters as their Unicode representation. Some characters can be represented the same visually but use different Unicode encoding and as a result, will be treated as different characters. When tokenizing, the characters will be considered distinct which will impact consistency. This can be avoided by using consistent encoding in the raw data.

The following examples illustrate the requirement for consistent encoding:

- For example, `fi` can be represented in unicode as a single Unicode point (*ligature*) using `\uFB01` or as two separate unicode points using `\u0066 \u0069`. When consistently tokenizing these two forms of `fi`, they will be tokenized to different outputs.

  This discrepancy can be avoided by using consistent encoding in the raw data. One approach to achieve consistent encoding is to normalize the strings into one of the standard Unicode normal forms. Normalizers are available in most languages such as Java and Python.
- A single character could also be represented by multiple Unicode points which can impact Rules that use *substring* behavior such as the **Clip Text** Rule (or the **Regular Expression Text Generator** Rule when partially replacing the input).

  For example, applying a Clip rule with a length of 4 against `café` might produce `cafe` or `café` depending on whether the é was represented by one or two Unicode points.

Multi-language Support in the API

Multi-language support is also provided in the Privitar API. To represent characters from different languages, use the Unicode representation of the character.

For example, to specify the following Regex range for Japanese characters in the API:

```
[-]{5}
```

use the following Unicode:

```
[\u{30A1}-\u{30F4}]{5}
```

It is not possible to use Unicode notation in the user-interface. When specifying a Regex range, you must use the language characters.

For more information about specifying Regex ranges using Unicode representation, see Unicode range RegExp generator.

Further information

Please refer to the following external sources for more information about some of the terms and concepts used in this section:

- Representing characters in Unicode. See, Character Encoding: Essential concepts.
- Unicode Normal forms. See, Unicode Normalization Forms.
- Normalizer for Java. See, Oracle docs : Class Normalizer.
- Normalizer for Python. See, Unicode Character Database.

5.8.8. Masking Rule Types Supporting Rule Preview

The following table lists all the Masking Rules available in Privitar and whether they support a Rule Preview. For Rules that do not provide a preview, this is either because the functionality is not supported, or because the Rule behavior can't be previewed. For example, the **Drop Column Rule**.

Rule	Description
Clip text	Supported.
Constant Text Value	Supported.
Date Generator	Supported.
Drop Column	Not suitable for preview.
Encrypt	Not suitable for preview.
Generalize Date	Supported.
Hash Text	Supported
Incremental Text Generator	Not supported.
Lookup Text	*Upload Lookup file* is supported. *Lookup from file* is **not** supported.
PAN Card Number Generator	Supported.
Perturb Currency	Supported.
Perturb Date	Supported.
Perturb Timestamp	Supported.
Random Number Generator	Supported.
Regular Expression Number Generator	Supported.
Regular Expression Text Generator	Supported.
Retain Value	Supported.
SecureLink Encryption	Not suitable for preview.
SecureLink Tokenization	Not suitable for preview.
Substitute Text	Supported.
Timestamp Generator	Supported.

5.8.9. Clip Text

This section provides a comprehensive description of the **Clip Text** Rule.

For a summary of the rule and its compatibility with Privitar jobs and execution environments, see Masking Rule Types.

Data Type

The supported data types for this rule are:

- Text

Description

This rule forms an output value by removing or retaining part of the input value. The clipping can be applied starting from the left of the value (default) or from the right of the value. For example:

```
SE12 8PZ
```

can become:

`SE12`

The clipped output value is obtained by either **retaining** the specified number of selected characters (if **Preserve Selected Characters** is checked) whilst removing all other characters, or **removing** those selected characters whilst retaining the rest (if **Preserve Selected Characters** is not checked).

For example, if the number of characters to clip is three characters, starting from the left:

`SW12 7AH`

becomes:

- `SW1` – (if **Preserve Selected Characters** is checked)
- `2 7AH` – (if **Preserve Selected Characters** is not checked)

Masking Behavior

The options are described in the following table:

Option	Description
Preserve Selected Characters	The action to take with the characters that have been selected from the input to derive the final output value. The clipped output value is obtained by either retaining the specified number of selected characters (if **Preserve Selected Characters** is checked) whilst removing all other characters, or removing those selected characters whilst retaining the rest (if **Preserve Selected Characters** is not checked).
Start from the Left	Clip characters starting from the left. If this box is not checked, characters are clipped from the right.
Number of Characters to Clip	The number of characters to clip from the input text.

Examples

Here are some examples of the **Clip Text** rule:

Input	Output	Description
`SE12 8PZ`	`SE12`	Clip 4 characters, starting from the left. (**Preserve Selected Characters** is checked.)
`SE12 8PZ`	`8PZ`	Clip 5 characters, starting from the left. (**Preserve Selected Characters** is not checked.)

5.8.10. Generalize Date

This section contains a comprehensive description of the **Generalize Date** rule.

For a summary of the rule and its compatibility with Privitar jobs and execution environments, see Masking Rule Types.

Data Types

The supported data types for this rule are:

- Date
- Timestamp

Description

The input date is replaced by a generalized version of the date, or a constant date if the input date is outside of a user-defined date range.

The date format used for input and output of `Date` data types is `yyyy-MM-dd`. For example:

```
2018-12-23
```

would become `2018-01-01` if the date is generalized to only preserve the year, and set the month and day to `01`.

The date format used for the input and output of Timestamp data types is `dd-MM-yyyy HH:mm:ss.SSS`. The output time will always be generalized to midnight. For example:

```
2018-09-16T11:33:00.465
```

would become `2018-01-01T00:00:00.000` if the date is generalized to only preserve the year, and set the month and day to `01`. There are no options to use a specific time of day when setting the date range.

Alternatively, a date range can be specified and a constant date returned if the input date is outside of the range specified. For example, if you set the minimum date to `1900-01-01` and the maximum date to `1970-01-01`, then an input date that is before `1900-01-01` or after `1970-01-01` will be generalized. For example:

```
1897-05-06
```

would become `1899-12-31` as it is before the minimum date (`1900-01-01`). The date is generalized to return the constant date that has been specified in the rule for the **Set constant before a date** (in this case `1899-12-31`).

Similarly:

```
1978-03-04
```

would become `1969-12-31` as it is after the maximum date (`1970-01-01`). The date is generalized to return the constant date that has been specified in the rule for the **Set constant after a date** (in this case `1969-12-31`).

The date format is always preserved in the output.

Note

This behavior of this rule is designed such that it can satisfy the HIPAA (Health Insurance Portability and Accountability Act) requirement for the storage of ages and dates contained in patient healthcare data. For more information about HIPAA and how to use this rule to de-identify data for compliance with the HIPAA Privacy rule, see Setting HIPAA Privacy Rules.

Generalization Behavior

The following table describes the Generalization options that can be used to define the constant date to be returned to replace the input value:

Option	Description
Set the day to	This can be any number from `1` to `31`. This can also be set to return the original input value of the day; `Original value`.
Set the month to	This can be any number from `1` to `12`. This can also be set to return the original input value of the month; `Original value`.
Set the year to	This can be any number from `1900` to `2100`. This can also be set to return the original input value of the year; `Original value`.

Important considerations when setting the date

To ensure that a *valid date* is returned by the rule, note the following points about setting the date:

- If a generalize date is set as `-/-/31` (where `-` is `Original value`) the date will be generalized to return the last valid day of the month, for months that don't contain the specified number of days. For example, if the input date is `2013/09/16`, the date returned by the rule would be `2013/09/30`.
- The rule will also check if the year specified is a *Leap year* and make similar changes to the date to return the closest valid date.

Masking Behavior

Masking behavior can be used to specify a date range within which to apply the Rule.

The following table defines the Masking behavior:

Option	Sub-option	Description
Set constant date based on	Absolute date	Use this option to specify a *Static Date*. That is, a date that is **not** relative to the current date.
	Relative date	Use this option to specify a *Dynamic Date*. That is, a date (time period) that is relative to the current date.

Option	Sub-option	Description
	Disable Masking behavior	Select this button to disable Masking behavior.
Set constant before a date		Use this option to set a **Minimum** date. That is, the date constant that will be used to replace the input date. This date constant will be applied if the input date is calculated to be **earlier** than the Minimum date.
Set constant after a date		Use this option to set a **Maximum** date. That is, the date constant that will be used to replace the input date. This date constant will be applied if the input date is calculated to be **later** than the Maximum date.

Some important points to note when defining the Masking behavior:

- When setting a date constant, the Date picker will prevent the setting of an invalid date in terms of days of the month. For example, it is not possible to set a date of `2020/09/31`. But, note that it is possible to enter a date directly into the **set date to** field. It is recommended that you use the Date picker to enter the date ranges.
- If a Minimum date **is** specified, but a Maximum date **is not** specified, then all input dates that are later than the Minimum date will be generalized according to the setting in **Generalization Behavior**.
- If a Minimum date **is not** specified, but a Maximum date **is** specified, then all input dates that are earlier than the Maximum date will be generalized according to the setting in **Generalization Behavior**.
- If specifying both Minimum **and** Maximum Static dates, then the Maximum date must be **later** than the Minimum date. An error will be displayed when the rule is saved, if a clash of dates is detected.
- If specifying both Before **and** After Relative dates, there must not be an overlap between the two specified dates. An error will be displayed when the rule is saved, if an overlap of dates is detected.
- For a Relative date, the exact generalization behavior depends on the environment in which the job is run:
  - For Batch jobs, the current date used will be the same for all records, even if the Job runs into the following day.
  - For data flow jobs, the current date will update over time.

    The current date is updated when the data flow processor refreshes the Job definition. The frequency of the update is determined by the cache refresh setting in the `application.properties` file. By default, the refresh is set to 10 minutes.

    So, in the default case, the current date will be updated every 10 minutes. This setting should be sufficient for most use-cases that are applying this rule to a dataset, but contact your system administrator if you need a shorter refresh time period.
- For both Absolute and Relative Dates, the replacement date constant that is used for dates outside the date range will not be affected by a change in the current date. That is, the date constant will not be affected by a Job running into the following day.

5.8.11. Date Generator

This section provides a comprehensive description of the **Date Generator** Rule.

For a summary of the rule and its compatibility with Privitar jobs and execution environments, see Masking Rule Types.

Data Types

The supported data types for this rule are:

- Date

Description

The value is replaced by a randomly generated date within a user-defined date range, using a specified format. The default format is `yyyy-MM-dd`. For example:

2018-12-23

The **Start of date range (inclusive)** field is the earliest date that can be randomly selected within the specified range.

The **End of date range (exclusive)** field is the latest date that can be randomly selected within the specified range.

Masking Behavior

The options are described in the following table:

Option	Description
Date Format	The format of the output is specified by a Date Format string for CSV file processing. The format used is described Date and Timestamp formats. (On Avro, Parquet and Hive the natively encoded format will be retained.)
Start of date range (inclusive)	Start of date range that is used to generate the random date. The date is inclusive. That is, it includes the specified date in the date range. (That is, a start of date range of 2020/05/03 would include 2020/05/03.)
End of date range (exclusive)	End of date range that is used to generate the random date. The date is exclusive. That is, it excludes the specified date from the date range. (That is, a end of date range of 2020/05/03 would generate date values up to 2020/05/02.)

Tokenization Behavior

Tokenization Behavior contains various settings that determine how tokenization is performed when the rule is applied to a dataset.

Note

Some rules may not have all the settings described, as some of the settings are only appropriate for certain types of rules.

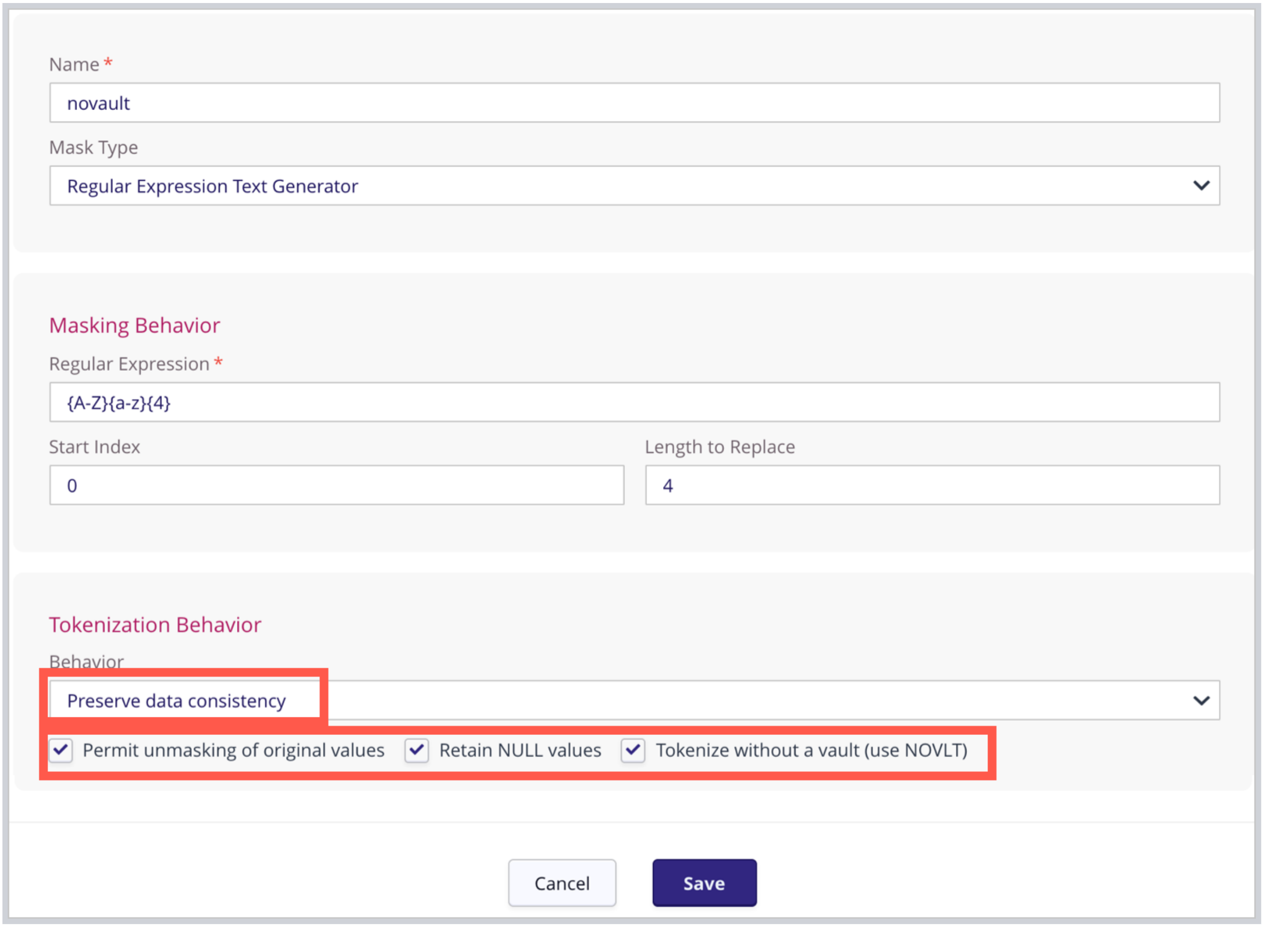
The **Behavior** setting determines whether a Token Vault is used to ensure consistency and to allow unmasking of the generated values. There are two options:

- **Preserve Data Consistency**
- **Do not preserve data consistency, token duplication allowed**.

Select **Preserve Data Consistency** to consistently tokenize values within the same Protected Data Domain (PDD). That is, re-use the same tokens for previously seen raw values to be able to preserve referential integrity for this field within the PDD (but not to other PDDs).

The other settings include:

- **Permit unmasking of original values**
- **Retain NULL values**
- **Tokenize without a vault (use NOVLT)**


Name *: novault

Mask Type: Regular Expression Text Generator

Masking Behavior

Regular Expression *: {A-Z}{a-z}{4}

Start Index: 0

Length to Replace: 4

Tokenization Behavior

Behavior: Preserve data consistency

✓ Permit unmasking of original values ✓ Retain NULL values ✓ Tokenize without a vault (use NOVLT)

When consistency is preserved, optionally some Rules allow you to enable unmasking for users with the corresponding permissions by checking **Permit unmasking of original values**.

If **Retain NULL values** is checked, NULL values in the input will not be replaced/tokenized and will be retained as NULL in the output.

5.8.12. Hash Text

This section provides a comprehensive description of the **Hash Text** Rule.

For a summary of the rule and its compatibility with Privitar jobs and execution environments, see Masking Rule Types.

Data Types

The supported data types for this rule are:

- Text

Description

The original value is completely replaced by a generated SHA256 salted hash of the value and a pepper (secret salt). By default, the hash will be a base 64 string; however, you can provide a regular expression for the hash output.

The rule does not utilize a Token Vault, so there is no support for Unmasking of any output value. As the input value is hashed, there is no way to return to the input value; the rule output is always irreversible.

Consistency in tokenization is achieved by the hashing function. Effectively, the function will always return the same output value for the same input value within a given PDD for a specific rule.

Note

This rule can only be used in a Job that has a KMS configured in the Privitar Environment. For more information, see Key Management Environment Configuration.

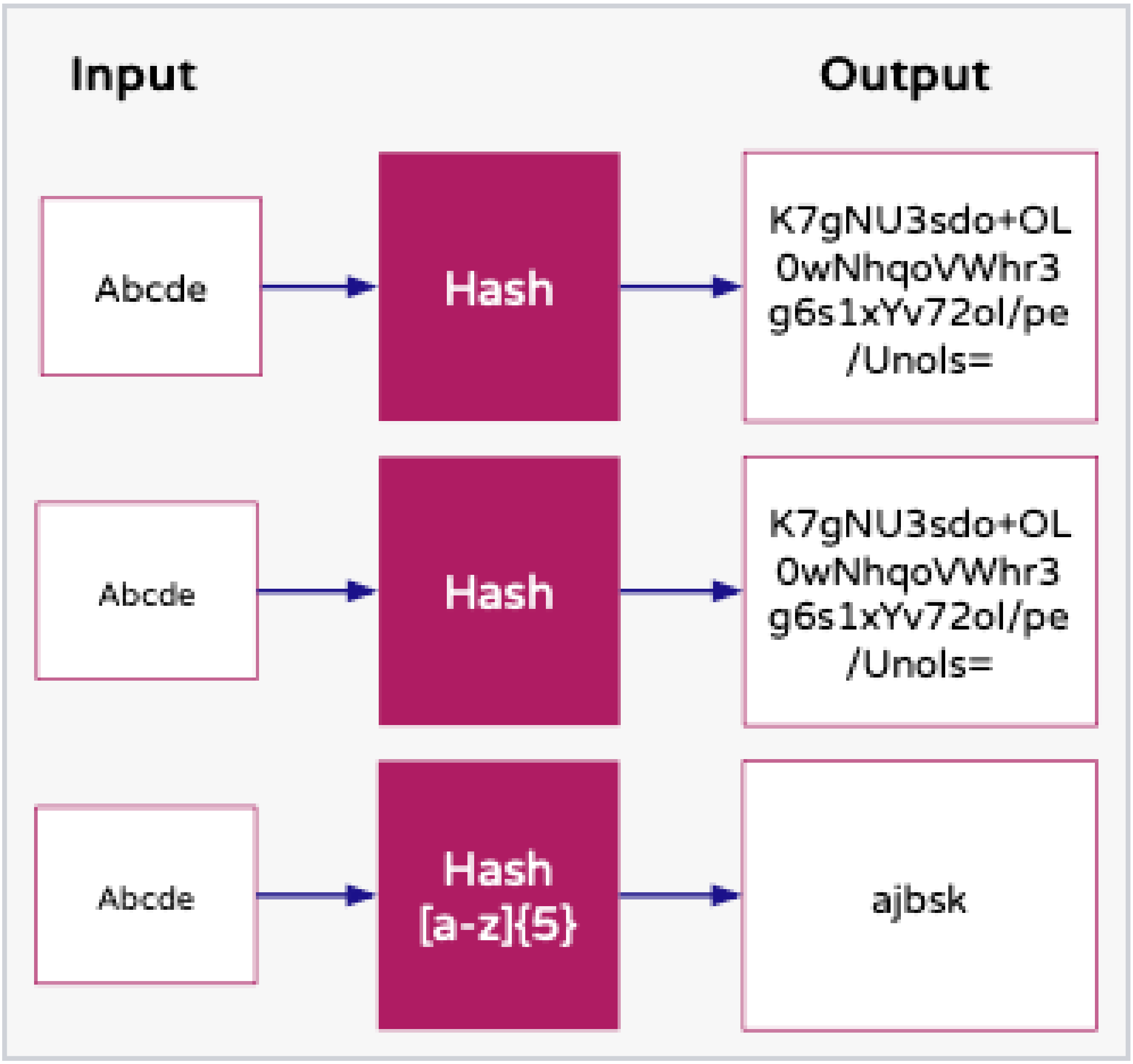
Masking Behavior

The options are described in the following table and assume that the original value is not null:

Option	Description
default	If you do not specify a regular expression, the default output is a (hashed) base 64 string.
Regular expression	The pattern that the generated text should match. Using a regular expression with the rule ensures that it is possible to add a Watermark to the dataset. It is not possible to add a watermark if the rule is used without a regular expression. For more information, see Watermarking a Dataset. For more information about the regular expression syntax supported in Privitar, see Regular Expression Syntax. (Click on the *RegExp* class in the *Class Summary* table.)

Examples

The following diagram illustrates the output behavior of the rule. The first two examples show how the same input value produces the same output value. The final example shows how the output value can be changed using a regular expression:


Input
Output
Abcde
Hash
K7gNU3sdo+OL OwNhqoVWhr3 g6s1xYv72ol/pe /Unols=
Abcde
Hash
K7gNU3sdo+OL OwNhqoVWhr3 g6s1xYv72ol/pe /Unols=
Abcde
Hash [a-z]{5}
ajbsk

Here are some other examples of regular expressions that could be used to match some example fields and formats:

Field	Format	Expression
Email address	xxxxxxx@xxxxx.com	[a-z]{7}\@[a-z]{5}\.com
Surname	xxxxxxxx	[a-z]{8}

Tokenization Behavior

Tokenization Behavior contains various settings that determine how tokenization is performed when the rule is applied to a dataset.

For the Hash text rule, the **Behavior** setting is fixed as:

Consistency enforced by hashing function but duplicate tokens are possible

This means that the hashing function ensures that the same input value will always return the same output value. But, there is the unlikely possibility of duplicate tokens being generated. (The collision resistance of the SHA256 hashing algorithm is discussed in many external publications.)

However, collisions are much more likely if a regular expression is specified with the rule. For example, if the regular expression defines an output that is smaller than the default hash output.

If **Retain NULL values** is checked, NULL values in the input will not be replaced or tokenized and will be retained as NULL in the output.

Hash Text Environment Requirements

The hash text rule can only be used in environments that have a KMS configured.

The rule does not require the user to specify any key details as the key will be created automatically on the first execution of any hash text rule in a given environment. The Privitar Platform does not support rotation or deletion of the key (even in case the environment is deleted or the KMS type is changed). In case the key is deleted or rotated by the user directly from the KMS this will result in any newly processed data to be inconsistent with previously tokenized data.

Required Permissions for using the Hash Text Rule with AWS Secrets Manager

For the Hash Text Rule all the communication with the KMS in this case AWS Secrets Manager is done on the on the execution engine (POD, Hadoop Batch Processor, SDK, other data flow processor) using the AWS SDK and will use the Region and Endpoint to connect to the AWS secrets manager.

We require both read and write access to AWS Secrets Manager for the Hash Text rule as we will create the secret if missing on first use. AWS has a managed policy for Secrets Manager that will grant the necessary permission, however customer might chose another policy or create its own in which case we will require that this grants the necessarily permissions for the following AWS Secrets Manager actions:

- DescribeSecret - see Minimum permissions required
- ListSecretVersionIds - see Minimum permissions required
- CreateSecret - see Minimum permissions required (we always require the secretsmanager:TagResource permission as we tag the secret with the key algorithm)
- GetSecretValue - see Minimum permissions required

In case of using customer manager CMK they also need to grant access to the AWS KMS for the following actions:

- GenerateDataKey - needed only if you use a customer-managed AWS KMS key to encrypt the secret. You do not need this permission to use the account default AWS managed CMK for Secrets Manager.

- Decrypt - needed only if you use a customer-managed AWS KMS key to encrypt the secret. You do not need this permission to use the account default AWS managed CMK for Secrets Manager.

Required Permissions for using the Hash Text Rule with AWS Secrets Manager

By default, the secret created by the Hash Text rule will be able to seen as decided by the existing IAM policies. We recommend that you restrict this to a specific role, specifically the role that you create while following these steps. You can also do this manually by attaching an AWS Resource Policy to the secret. To learn more: AWS Documentation: Attaching a resource-based policy to a secret

1. To have the platform create the secret, run a job using the environment that you created when following the steps in Configure an AWS Secret Manager as a KMS in Policy Manager.
2. In the Amazon Elastic Compute Cloud (EC2) console, navigate to AWS Secrets Manager > Secrets.
3. Search for your environment ID.
4. Open the secret created by the platform, which will have a name with a format similar to “privitar-hash-rule-[YOUR_ENVIRONMENT_ID].”
5. Edit resource permissions and add a restriction like that in this example, replacing the Amazon Resource Name (ARN) with the ARN of the role that you just created.

To restrict access to just SecretsManager_Limited in account 12345555555, the resource policy would have the form:

```
{
  "Version" : "2012-10-17",
  "Statement" : [ {
    "Effect" : "Deny",
    "Principal" : "*",
    "Action" : "secretsmanager:*",
    "Resource" : "*",
    "Condition" : {
      "ArnNotLike" : {
        "aws:PrincipalArn" : "arn:aws:iam::1234555555:role/
Secrets_manager_limited"
      }
    }
  } ]
}
```

The user updating the policy must assume the role to which access will be restricted, otherwise the user will be locked out. If they can’t assume this role, then either they should receive temporary access to be able to update the resource policy, or the resource policy should include the user, in a form similar to the following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "StringNotLike": {
                    "aws:PrincipalARN": [
                        "<ARN-OF-ALLOWED-ROLE>",
                        "<ARN-OF-ALLOWED-USER>"
                        ]
                }
            }
        }
    ]
}
```

Glue Environments

Glue deployments have permission to create secrets and the first time a hash text rule is used, it will create one for that rule.

The following resource policy is automatically attached to the secret created.

```
{
  "Version": "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Deny",
      "Principal": "*",
      "NotAction" : [
        "secretsmanager:DeleteSecret",
        "secretsmanager:RestoreSecret",
        "secretsmanager:GetResourcePolicy"
      ],
      "Resource" :"*",
      "Condition" :
      {
        "ArnNotLike" :
        {
         "aws:PrincipalArn" : <Glue job IAM role arn>
        }
      }
    }]
}
```

This denies access to all principals except the IAM role that the Glue job uses. To allow the secret to be deleted in the future, it doesn't deny DeleteSecret permissions. Note that the secret would have to be deleted via the AWS CLI command below, because deletion from the AWS console UI requires DescribeSecret permissions.

```
aws secretsmanager delete-secret --secret-id <secret name> --force-delete-without-recovery
```

5.8.13. Lookup Text

This section provides a comprehensive description of the **Lookup Text** Rule.

For a summary of the rule and its compatibility with Privitar jobs and execution environments, see Masking Rule Types.

Data Types

The supported data types for this rule are:

- Text

Description

The value is replaced by a randomly selected value read from from a Lookup file stored in HDFS or uploaded locally. For example, here is a typical Lookup file consisting of six names:

```
Gary
Joshua
Daniel
Thomas
Craig
Samuel
```

This file could be used by the rule to map the following input values to new output values:

Input value	Output value	Mapping
John	**Daniel**	John > Daniel
Bob	**Craig**	Bob > Craig
Christian	**Gary**	Christian > Gary
Louis	**Joshua**	Louis > Joshua

If using **Lookup From File**, specify the name of a file in **Lookup From File**. The full HDFS directory path and filename must be specified.

If using **Upload Lookup File**, use *Drag & drop a file* or **Browse** to upload a file stored locally.

The Lookup file must be a CSV file that contains a single column of newline-separated replacement values (no comma is needed to separate the values). The replacement value is drawn at random from this file. The size of the Lookup file can be up to 100k rows.

Tokenization will fail if there are fewer replacement values in the Lookup file than original values to be replaced.

Masking Behavior

The options are described in the following table:

Option	Description
Lookup From File	The full HDFS directory path including filename of the Lookup file.

Option	Description
Upload Lookup File	File to be uploaded.

Tokenization Behavior

Tokenization Behavior contains various settings that determine how tokenization is performed when the rule is applied to a dataset.

Note

Some rules may not have all the settings described, as some of the settings are only appropriate for certain types of rules.

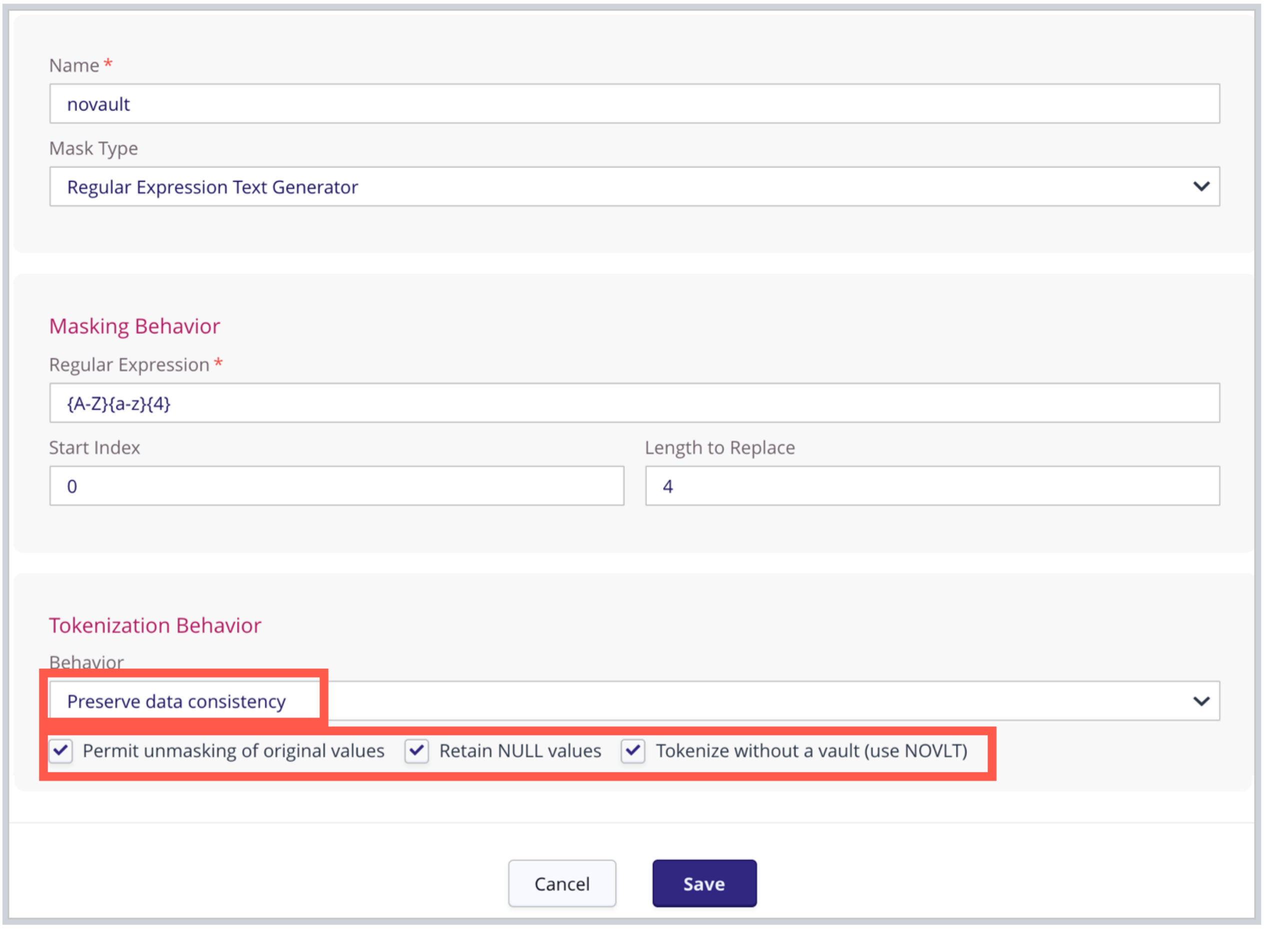
The **Behavior** setting determines whether a Token Vault is used to ensure consistency and to allow unmasking of the generated values. There are two options:

- **Preserve Data Consistency**
- **Do not preserve data consistency, token duplication allowed**.

Select **Preserve Data Consistency** to consistently tokenize values within the same Protected Data Domain (PDD). That is, re-use the same tokens for previously seen raw values to be able to preserve referential integrity for this field within the PDD (but not to other PDDs).

The other settings include:

- **Permit unmasking of original values**
- **Retain NULL values**
- **Tokenize without a vault (use NOVLT)**



Name *
novault
Mask Type
Regular Expression Text Generator

Masking Behavior
Regular Expression *
{A-Z}{a-z}{4}
Start Index
0
Length to Replace
4

Tokenization Behavior
Behavior
Preserve data consistency
[✓] Permit unmasking of original values [✓] Retain NULL values [✓] Tokenize without a vault (use NOVLT)

When consistency is preserved, optionally some Rules allow you to enable unmasking for users with the corresponding permissions by checking **Permit unmasking of original values**.

If **Retain NULL values** is checked, NULL values in the input will not be replaced/tokenized and will be retained as NULL in the output.

5.8.14. Regular Expression Number Generator

This section provides a comprehensive description of the **Regular Expression Number Generator** Rule.

For a summary of the rule and its compatibility with Privitar jobs and execution environments, see Masking Rule Types.

Data Types

The supported data types for this rule are:

- Byte, Short, Integer, Long

Description

The value is replaced by a randomly generated number that would match the supplied regular expression. For example, for an initial value:

```
1234
```

To replace this with a randomly generated 4-digit number, you could use the following regular expression:

```
[0-9]{4}
```

which could produce a value such as:

```
4059
```

A more complex example, would be:

```
[1-4]{1}[0-9]{3}
```

The first part of the expression (`[1-4]{1}`) generates one number between `1` and `4`. The second part of the expression (`[0-9]{3}`) generates three numbers between `0` and `9`.

The regular expression specifies a pattern that the generated number should match. Only expressions generating integers are accepted, and the range of potential minimum and maximum values that can be generated based on the expression has to be compatible with the data types the Rule is applied to.

Masking Behavior

The options are described in the following table:

Option	Description
Regular expression	The pattern that the generated number should match. For more information about the regular expression syntax supported in Privitar, see Regular Expression Syntax. (Click on the *RegExp* class in the *Class Summary* table.)

Examples

Here are examples of regular expressions that could be used to match some example fields and formats:

Field	Format	Expression
Employee ID (For example, 52-938486)	xx-xxxxxx	[0-9]{2}-[0-9]{6}

Tokenization Behavior

Tokenization Behavior contains various settings that determine how tokenization is performed when the rule is applied to a dataset.

Note

Some rules may not have all the settings described, as some of the settings are only appropriate for certain types of rules.

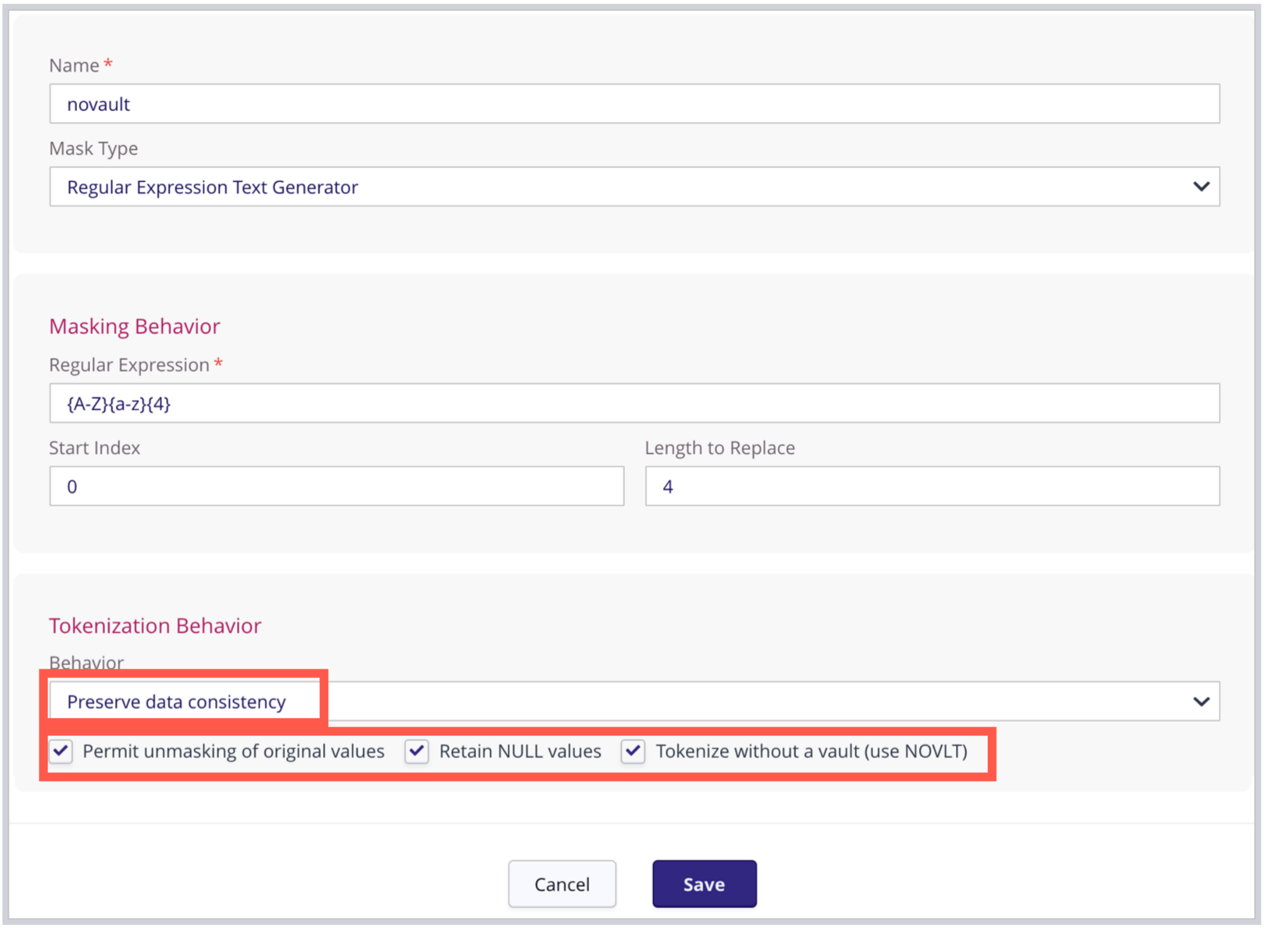
The **Behavior** setting determines whether a Token Vault is used to ensure consistency and to allow unmasking of the generated values. There are two options:

- **Preserve Data Consistency**
- **Do not preserve data consistency, token duplication allowed**.

Select **Preserve Data Consistency** to consistently tokenize values within the same Protected Data Domain (PDD). That is, re-use the same tokens for previously seen raw values to be able to preserve referential integrity for this field within the PDD (but not to other PDDs).

The other settings include:

- **Permit unmasking of original values**
- **Retain NULL values**
- **Tokenize without a vault (use NOVLT)**



Name *
novault
Mask Type
Regular Expression Text Generator

Masking Behavior
Regular Expression *
{A-Z}{a-z}{4}
Start Index
0
Length to Replace
4

Tokenization Behavior
Behavior
Preserve data consistency
[✓] Permit unmasking of original values [✓] Retain NULL values [✓] Tokenize without a vault (use NOVLT)

When consistency is preserved, optionally some Rules allow you to enable unmasking for users with the corresponding permissions by checking **Permit unmasking of original values**.

If **Retain NULL values** is checked, NULL values in the input will not be replaced/tokenized and will be retained as NULL in the output.

5.8.15. Regular Expression Text Generator

This section provides a comprehensive description of the **Regular Expression Text Generator** Rule.

For a summary of the rule and its compatibility with Privitar jobs and execution environments, see Masking Rule Types.

Data Types

The supported data types for this rule are:

- Text

Description

The value is partially or completely replaced by a randomly generated string that would match the supplied regular expression. For example, for an initial value:

```
abcdef
```

you could use the following regular expression:

```
[a-z]{6}
```

This would produce an output such as:

```
mvskyc
```

Optionally specify **Start Index** and **Length to Replace** to replace a part of the input value with the generated text. For example, with an input of `abcdef` and a pattern of `[0-9]{4}`, replacing four characters from position 1, an output might be `a4321f`.

Masking Behavior

The options are described in the following table:

Option	Description
Regular expression	The pattern that the generated text should match. For more information about the regular expression syntax supported in Privitar, see Regular Expression Syntax. (Click on the *RegExp* class in the *Class Summary* table.)
Start Index	Where to start replacing the string. This is required for partial replace. Start index 0 is the beginning of the string. Index 1 would be after the first character of the string.
Length to Replace	The length of the string to replace. This is required for partial replace. The value entered will be the number of characters from the string that will be replaced.

Examples

Here are examples of regular expressions that could be used to match some example fields and formats:

Field	Format	Expression
Email address	xxxxxxx@xxxxx.com	[a-z]{7}\@[a-z]{5}\.com
Surname	xxxxxxxx	[a-z]{8}

Tokenization Behavior

Tokenization Behavior contains various settings that determine how tokenization is performed when the rule is applied to a dataset.

Note

Some rules may not have all the settings described, as some of the settings are only appropriate for certain types of rules.

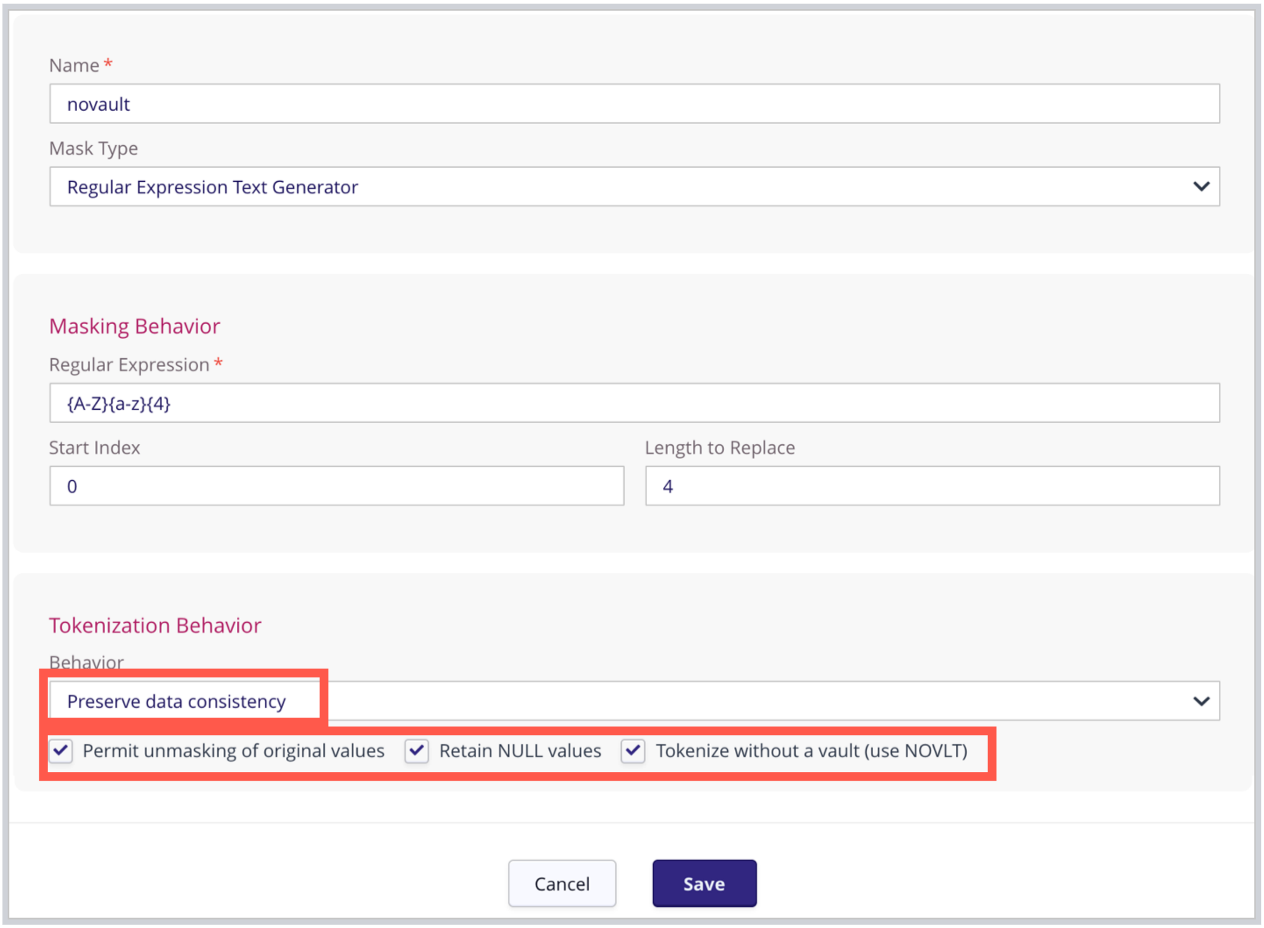
The **Behavior** setting determines whether a Token Vault is used to ensure consistency and to allow unmasking of the generated values. There are two options:

- **Preserve Data Consistency**
- **Do not preserve data consistency, token duplication allowed**.

Select **Preserve Data Consistency** to consistently tokenize values within the same Protected Data Domain (PDD). That is, re-use the same tokens for previously seen raw values to be able to preserve referential integrity for this field within the PDD (but not to other PDDs).

The other settings include:

- **Permit unmasking of original values**
- **Retain NULL values**
- **Tokenize without a vault (use NOVLT)**


Name *

novault

Mask Type

Regular Expression Text Generator

Masking Behavior

Regular Expression *

{A-Z}{a-z}{4}

Start Index

0

Length to Replace

4

Tokenization Behavior

Behavior

Preserve data consistency

✓ Permit unmasking of original values ✓ Retain NULL values ✓ Tokenize without a vault (use NOVLT)

When consistency is preserved, optionally some Rules allow you to enable unmasking for users with the corresponding permissions by checking **Permit unmasking of original values**.

If **Retain NULL values** is checked, NULL values in the input will not be replaced/tokenized and will be retained as NULL in the output.

5.8.16. Substitute Text

This section provides a comprehensive description of the **Substitute Text** Rule.

For a summary of the rule and its compatibility with Privitar jobs and execution environments, see Masking Rule Types.

Data Types

The supported data types for this rule are:

- Text

Description

The input value is replaced with a mapped substitute value from a Mappings file uploaded locally. For example, a typical file is:

```
John, Gary
Bob, Joshua
```

```
Christian, Daniel
Louis, Thomas
David, Craig
Samuel, Samuel
```

This file could be used by the rule to map the following input values to new output values:

Input value	Output value (mapping)
John	**Gary**
Samuel	**Samuel**
Christian	**Daniel**
David	**Craig**

Select **Mappings** to upload a file containing name-value pairs that determine how an input value is substituted. If there is no mapping for a particular input, then the value supplied in **Default Value** is used instead.

Note

The substitute value cannot be the string/text "NULL" (or any variation of this string, such as "null" or "NuLL".)

Masking Behavior

The options are described in the following table:

Option	Description

Mappings	The mappings file to upload. The file should be a CSV file. (The size of the file can be up to 100k rows.)
	
	The format of the file is:
	
	`Original value,Mapped value`
	
	For example:
	
	`Chicago,Area_1`

Default Value	A default mapping value to use if no mapping value is found in the mappings file for a particular input value.

Tokenization Behavior

Tokenization Behavior contains various settings that determine how tokenization is performed when the rule is applied to a dataset.

Note

Some rules may not have all the settings described, as some of the settings are only appropriate for certain types of rules.

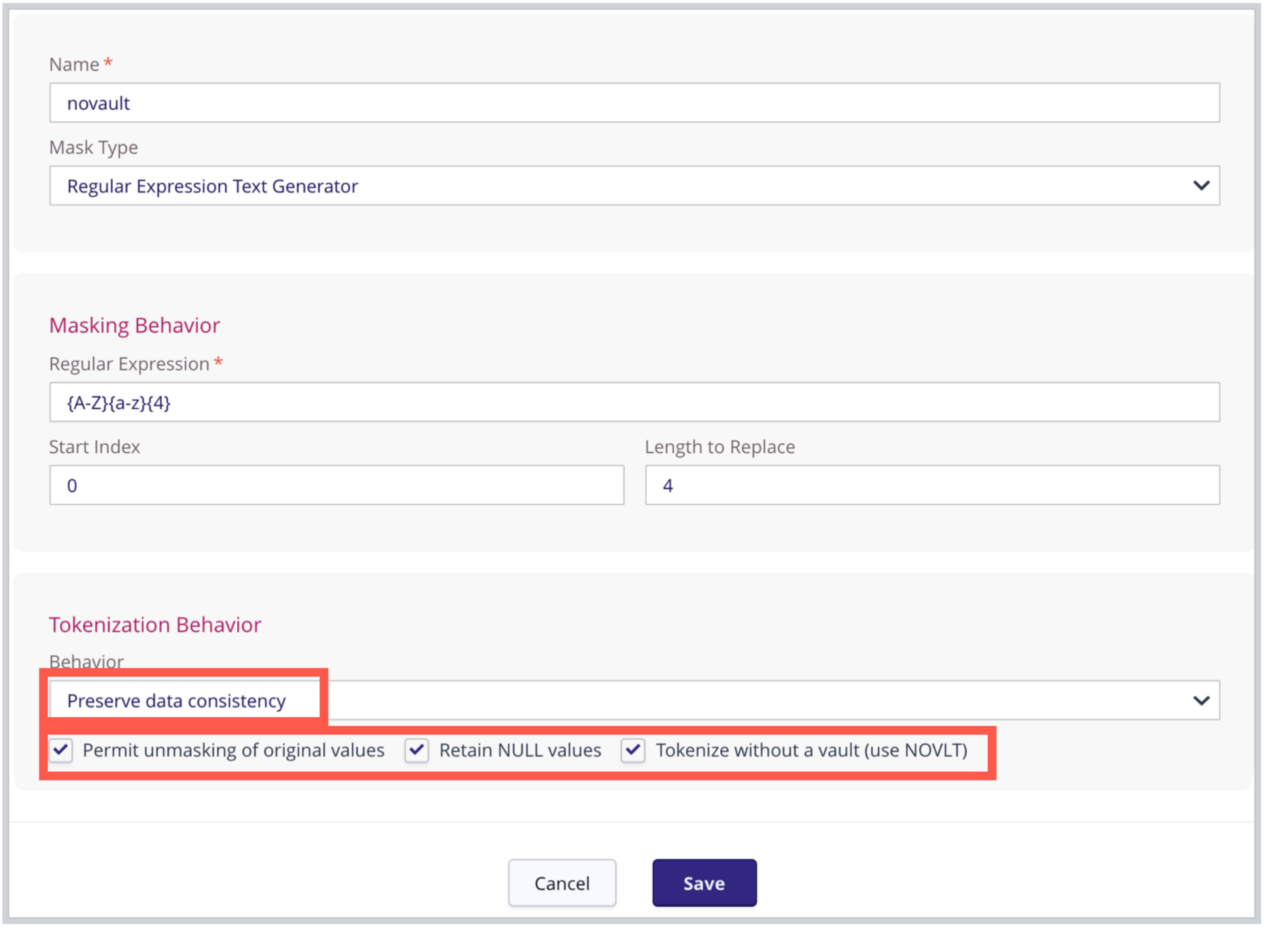
The **Behavior** setting determines whether a Token Vault is used to ensure consistency and to allow unmasking of the generated values. There are two options:

- **Preserve Data Consistency**
- **Do not preserve data consistency, token duplication allowed**.

Select **Preserve Data Consistency** to consistently tokenize values within the same Protected Data Domain (PDD). That is, re-use the same tokens for previously seen raw values to be able to preserve referential integrity for this field within the PDD (but not to other PDDs).

The other settings include:

- **Permit unmasking of original values**
- **Retain NULL values**
- **Tokenize without a vault (use NOVLT)**


Name *: novault

Mask Type: Regular Expression Text Generator

Masking Behavior

Regular Expression *: {A-Z}{a-z}{4}

Start Index: 0

Length to Replace: 4

Tokenization Behavior

Behavior: Preserve data consistency

✓ Permit unmasking of original values ✓ Retain NULL values ✓ Tokenize without a vault (use NOVLT)

When consistency is preserved, optionally some Rules allow you to enable unmasking for users with the corresponding permissions by checking **Permit unmasking of original values**.

If **Retain NULL values** is checked, NULL values in the input will not be replaced/tokenized and will be retained as NULL in the output.

5.8.17. Timestamp Generator

This section provides a comprehensive description of the **Timestamp Generator** Rule.

For a summary of the rule and its compatibility with Privitar jobs and execution environments, see Masking Rule Types.

Data Types

The supported data types for this rule are:

- Timestamp

Description

The value is replaced by a randomly generated timestamp, within a user-defined range, using a specified format. The default format is `yyyy-MM-dd'T'HH:mm:ss.SSS`. For example:

2019-10-21T21:34:21.254

The **Start of timestamp range (inclusive)** box is the earliest timestamp that can be randomly selected within the specified range.

The **End of timestamp range (exclusive)** box is the latest timestamp that can be randomly selected within the specified range.

Masking Behavior

The options are described in the following table:

Option	Description
Timestamp Format	The format of the output is specified by a Date Format string for CSV file processing. The format that can be used is described Date and Timestamp formats. (For Avro, Parquet and Hive files the natively encoded format will be retained.
Start of Timestamp range (inclusive)	Start of date range that is used to generate the random timestamp. The date is inclusive. That is, it includes the specified timestamp in the timestamp range.
End of Timestamp range (exclusive)	End of date range that is used to generate the random timestamp. The date is exclusive. That is, it excludes the specified timestamp from the timestamp range.

Tokenization Behavior

Tokenization Behavior contains various settings that determine how tokenization is performed when the rule is applied to a dataset.

Note

Some rules may not have all the settings described, as some of the settings are only appropriate for certain types of rules.

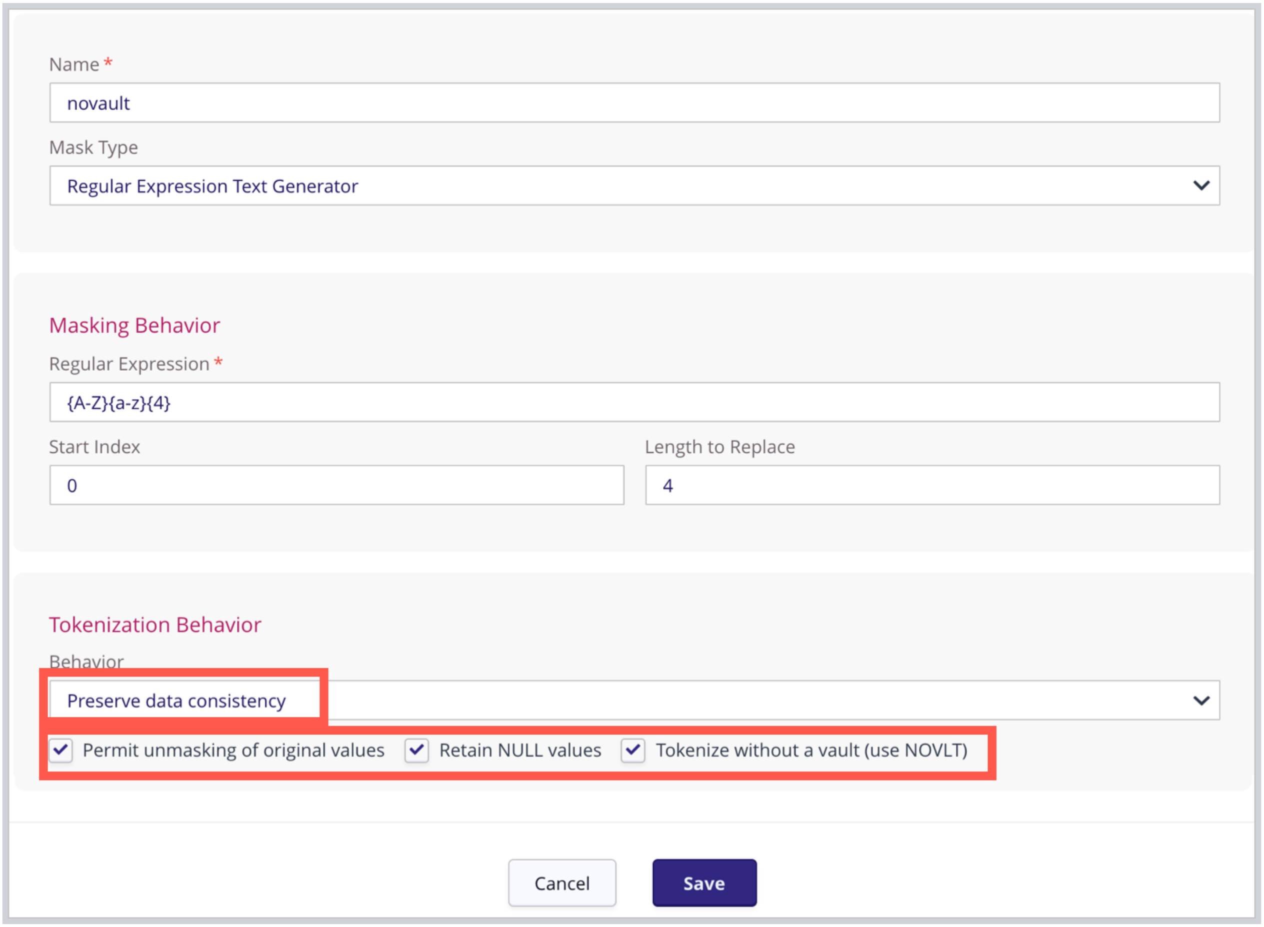
The **Behavior** setting determines whether a Token Vault is used to ensure consistency and to allow unmasking of the generated values. There are two options:

- **Preserve Data Consistency**
- **Do not preserve data consistency, token duplication allowed**.

Select **Preserve Data Consistency** to consistently tokenize values within the same Protected Data Domain (PDD). That is, re-use the same tokens for previously seen raw values to be able to preserve referential integrity for this field within the PDD (but not to other PDDs).

The other settings include:

- **Permit unmasking of original values**
- **Retain NULL values**
- **Tokenize without a vault (use NOVLT)**


Name *
novault
Mask Type
Regular Expression Text Generator

Masking Behavior
Regular Expression *
{A-Z}{a-z}{4}
Start Index
0
Length to Replace
4

Tokenization Behavior
Behavior
Preserve data consistency
✓ Permit unmasking of original values ✓ Retain NULL values ✓ Tokenize without a vault (use NOVLT)

When consistency is preserved, optionally some Rules allow you to enable unmasking for users with the corresponding permissions by checking **Permit unmasking of original values**.

If **Retain NULL values** is checked, NULL values in the input will not be replaced/tokenized and will be retained as NULL in the output.

6. Managing Protected Data Domains

This section describes how de-identified data for a specific business use is contained within a Protected Data Domain (PDD).

6.1. What is a PDD?

A Protected Data Domain (PDD) is a logical collection of datasets, brought together for a specific purpose, with strict linkage isolation from datasets in other PDDs.

A PDD should be created whenever datasets are required for a particular use, such as an analysis project, a development/testing activity, or sharing with an external party. The PDD definition remains in Privitar, allowing data to be added incrementally over time.

Processing in Privitar always produces output in a destination PDD. The PDD is selected when a de-identification Job is used.

In Privitar, PDDs contain the following information:

- Metadata about the specific intended usage of the de-identified data. For example, the recipient and purpose of the data release.
- The output location for de-identified data, if a Hadoop cluster is configured in the Environment. This may be a location in HDFS and/or a Hive database.
- Metadata on the published datasets, including the Policies and Jobs used to process them.
- Information about the Token Vaults used by masking rules.

6.1.1. Protected Data Domains and Data Consistency

Datasets published to the same PDD will have preserved data consistency when processed by Rules with the **Preserve Data Consistency** option enabled.

Datasets published to a specific PDD will always be inconsistent with datasets published to a different PDD, even if the same Policies and Jobs were are used to process them.

This isolation means that while data linkability is possible within a single PDD, recipients of different PDDs will be prevented from linking between those PDDs.

6.1.2. Protected Data Domains and Watermarking

Datasets published to the same PDD may optionally contain a unique signature embedded as a hidden **Watermark**. This allows a file to be traced back to its original PDD.

This is useful, for example, in situations where data that has been shared externally needs to be attributed back to a PDD following a data breach. The PDD contains useful metadata on the file, such as the name of the authorizer and its original recipient.

Warning

When running data flow or POD jobs, the PDD is still editable after you have processed data into that PDD. If watermarking was previously disabled, and you enable it after you processed data, this could make it harder to investigate the watermark when some of the data doesn't have a watermark.

For more information about this feature, see Watermarking a Dataset.

6.2. Creating a PDD

In addition to creating PDDs directly from the user interface, PDDs can also be created *inline* as part of the process for running a Batch Job or creating a Data Flow Job. PDDs may also be created using the Privitar automation APIs.

To create a PDD:

1. Select **Protected Data** from the Navigation sidebar. The **Protected Data** page is displayed, showing an index of all the PDDs that have been created.
2. Click on **Create New Protected Data Domain**. The **Create Protected Data Domain** screen is displayed.
3. Enter details about the new PDD:
   - A **Name** that will be used to refer to this specific PDD.
   - Whether to attempt to **Embed Watermarks** in the de-identified files.

   Note

   Enabling this option does not guarantee that every processed file will contain a watermark. Some small files may not be compatible, and the Policy used with the PDD must include at least one masked column. For more information, see Watermarking a Dataset.
4. Enter details about the output locations of the PDD in **Output Data Locations**. PDDs can contain output in various locations. Depending on the Environment configuration, HDFS, Hive and AWS Glue may be available. When using Data Flow or Privitar On Demand Jobs, no output location is required.
5. If a Hadoop cluster is configured in the Environment:
   - Enable **Batch Jobs**.
   - Enter the HDFS output path of the folder where output data will be written in the **Batch Output Path**.
6. If Hive is configured in the Environment:
   - Enable **Hive Batch Jobs**.
   - Enter the Hive **Database Name** into which output records will be inserted.
   - (Depending on configuration) Enter the HDFS Output path where Hive table data should be stored in the **Write Job output to** field. This option is presented only

if the value is not specified centrally by a system administrator in the PDD's Environment.

7. If AWS Glue is configured in the Environment:
    - Enter the Amazon S3 output path of the folder where output data will be written in the **Batch Output Path**. Please read these important details about Amazon S3 bucket permissions.
8. If metadata attribute mappings are configured and enabled (See Managing Metadata Attributes), then they will appear under **Properties**.
9. Complete the required **Metadata** fields. The fields shown in the dialog box depend on the Privitar configuration. (For more information, see Managing Metadata Attributes.)
10. Click **Save** to save the new PDD.

6.3. Viewing PDD details

The *Protected Data* page provides a summary of the PDDs that have been created on the Privitar platform.

The page contains a table with a separate entry for each PDD. The following information is provided for each PDD:

- **Name**: The name of the PDD.
- **Watermark**: If Watermarking has been enabled for the PDD.
- **Expiration Date**: the value of the expiration date of the PDD, if the associate metadata attribute is enabled and mapped.
- **Status**: the current status of the PDD (See What is the lifecycle of a PDD?for details)

To view more details about a PDD, click on the name of the PDD in the table. The PDD details page is displayed.

The PDD details page provides high-level information about the PDD, and gives access to its metadata, activity, and linked content. It also allows the creation of Unmasking Jobs.

The high-level information available is:

- **ID** for the PDD. This is the internal identifier of the PDD. It could be used with the Privitar automation APIs or data flow jobs, for example.
- **Status**: the current status of the PDD (See What is the lifecycle of a PDD?for details)
- **Watermark**. if the PDD has Watermarking enabled or disabled, this displays whether watermarks are embedded in data produced for this PDD.
- **Environment**. This displays the Privitar Environment in which the PDD is located.

In addition, the page allows access to various PDD functions:

- **Audit Log**. Use this link to see the history of changes to the PDD in the audit section of the Policy Manager.
- **Unmask value**. Use this link to reverse the tokenization of selected values for this PDD. This operation is subject to permission requirements.
- **Edit**. Use this link to rename the PDD or change its watermark status. Note that the watermark status cannot be changed once a Job has been run in the PDD.

- **Close**. Use this link to close the PDD. Once this is done, no further data can be linked to the PDD.

6.3.1. Summary tab

The **Summary tab** is separated into 4 areas, depending on your configuration to show the:

- **Description**, only display if mapped in the **Metadata Attribute** page (See Managing Metadata Attributes for details)
- **PDD Expiry Date**, only display if mapped in the **Metadata Attribute** page (See Managing Metadata Attributes for details)
- **Locations**, if HDFS, or Hive support, are defined in the PDD environment configuration. They may only be changed if no Job has yet been run for this PDD.
- **Metadata**, based on the **Metadata Attribute** page. The mapped attributes are not displayed on this list.

To edit the metadata attributes values, click on the associated cog icon.

6.3.2. Activity tab

This section shows a list of the currently running Jobs and past Jobs run for this PDD.

The different Jobs will show the following attributes:

- Job name
- Job submitter
- Policy used in the Job
- Job status
- Timestamp for the last update

It also provides a list of Jobs run previously and completes the details of the running Jobs with:

- Type
- Output Location
- Input Location
- Bad records
- Size of the input
- Number of rows of the input

6.3.3. Content tab

This section provides a summary of the output of the different Batch Jobs associated with the current PDD. It will also show any bad records, along with an error message. You may need to fix the problem before executing the job again.

Note

This view shows the last run's result of the output files. Specifically, if the same job was run multiple times in this PDD, every batch job run overrides the previous run

6.3.4. Unmasking tab

Use this tab for any tasks related to unmasking jobs for the current PDD.

To create unmasking jobs, please follow instructions from Creating and Running Data Flow/POD Unmasking Jobs or Creating Batch Unmasking Jobs.

6.4. Closing and Deleting PDDs

If the need for a specific PDD has ended, it is good practice to *close* the PDD.

6.4.1. Closing a PDD

A PDD should be closed when it is no longer needed. Closing a PDD has the following effects:

1. The PDD will become *read only;* no new data may be added. However, existing de-identified data associated with the PDD is unchanged.
2. The Token Vault used to ensure token consistency in this PDD is discarded; therefore, closing a PDD also prevents any unmasking of tokens associated with the PDD.

PDD metadata and the history of processing associated with the PDD are not removed. It is possible for a watermark investigation to identify a dataset as originating from a PDD that has subsequently been closed.

A PDD might be closed for data minimization reasons, or to control storage volume:

- If de-identified output needs to be made irreversible, the PDD can be closed to prevent unmasking.
- If the need for a PDD is eliminated, or if it is only appropriate for it to contain a finite amount of data, closing the PDD will stop new data from being published into it.
- PDDs require on-disk storage for Token Vaults, so it may be necessary to close unused PDDs to recover space.

A closed PDD will display a lock next to its name on the Protected Data index.

To close a PDD:

1. Select **Protected Data** from the Navigation sidebar. The **Protected Data** page is displayed, showing an index of all the PDDs that have been created.
2. Click on the row select button alongside the PDD to select the PDD.
3. Select **Close Protected Data Domain** from the **Actions** list box.

   The **Close Protected Data Domain** dialog box is displayed asking you to confirm that you want to close the PDD.

4. Click on **Close Protected Data Domain**.

   It may take a few minutes to close the PDD. Click on **Run in Background** to hide the dialog box.

Alternatively, a PDD can be closed from the details page of an individual PDD by clicking **Close Protected Data Domain** at the top of the page, or it can also be closed programmatically via the Privitar Automation API (v3).

Note

Closing a PDD is an asynchronous process. You can monitor the status on the UI in the **Protected Data Domain** page or via the Privitar Automation API (v3). Depending on which type of Token Vault the Protected Data Domain uses, different execution modes are used for the closing process. For example, HDFS/cloud storage and HBase Token Vaults will be closed via Spark Batch/Hadoop, whereas JDBC and DynamoDB vaults are closed via the Privitar application or by Privitar On Demand (POD).

6.4.2. Deleting a PDD

PDDs may be completely deleted. A PDD must have been previously closed before it can be deleted. Once this is done, the following steps will delete a PDD from Privitar's configuration.

Note

Most of the time, closing a PDD is enough.

Typical reasons why you may delete a PDD include:

- If a PDD was created as a mistake.
- If you would like to purposely disallow watermarking investigations.
- If you need to clean up deprecated, or unused, PDDs

To delete a PDD:

1. Select **Protected Data** from the Navigation sidebar. The **Protected Data** page is displayed, showing an index of all the PDDs that have been created.
2. Click on the Row select button alongside the PDD to select the PDD.
3. Select **Delete Protected Data Domain** from the **Actions** list box.

   The **Delete Protected Data Domain** dialog box is displayed asking you to confirm that you want to delete the PDD.
4. Click on **Delete Protected Data Domain**.

Note

Deleting a PDD only deletes the PDD from Privitar's configuration. De-identified data written to HDFS/Hive or processed as part of a data flow pipeline or via a Privitar On Demand API will not be deleted.

Note

Deleting a PDD should be done with caution as it is not possible for deleted PDDs to be identified as the result of watermarking investigations.

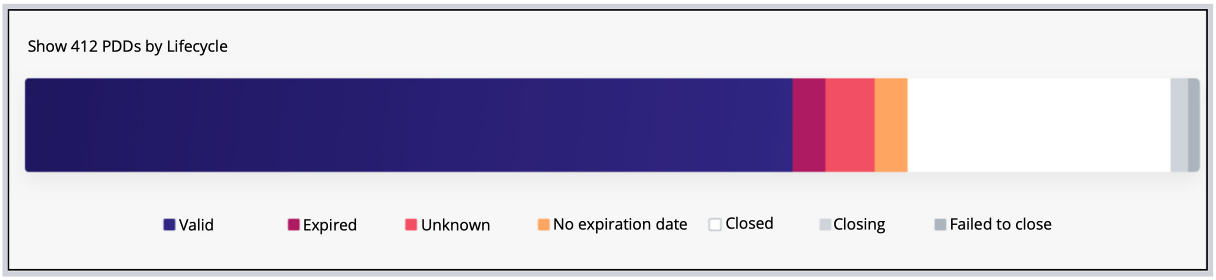
6.5. What is the lifecycle of a PDD?

On the **Protected Data Domains** page, the Privitar platform summarizes the current status of all PDDs that have been created. This summary includes whether the PDD is Open or Closed, and (optionally) whether the PDD is expired.

Using an overview of the status of PDDs is important for the following reasons:

- To control and manage the datasets that are being used by your organization.
- To provide summary statistics to support any business compliance processes in your organization.

The status of all the PDDs that have been created on the Privitar platform is shown graphically at the top of the **Protected Data Domains** page.



There are two key attributes reported for each PDD:

- Expiration date
- Open/closed status

For a typical PDD lifecycle, it would be expected that there would be a number of PDDs that are **Valid**: open and not expired. There might also be a number of PDDs that are no longer in use and have been **Closed**.

The report will also show PDDs that are not in one of these two states and may require action. PDDs that are in any of the following states would be deemed as *unmanaged* and could pose a risk to your organization. The status report can be used to help you to quickly identify these PDDs:

- In terms of Expiration date, there may be Open PDDs that have **Expired**, or have **No Expiration date** set.
- In terms of Closure status, there may be PDDs that have **Failed to close**.
- There is also a PDD status of **Unknown** that may require investigative action. **Unknown** PDDs may occur due to configuration problems with metadata system integrations.

The information that is provided by PDDs on this page is provided by examining the metadata that has been added to PDDs. For more information about how to configure PDD metadata, see Managing PDD Lifecycle.

6.6. Managing PDD Lifecycle

Protected Data Domains have an optional *expiration date*, which determines the *lifecycle* of the PDD. When the Privitar Platform is configured with an appropriate metadata attribute to contain the expiration date, features related to the PDD's lifecycle are available.

Note

This feature is only available when a metadata attribute is mapped to *expiration date* (See ??? for the procedure).

PDDs have a lifecycle *status* that corresponds to acceptable use of the data in a PDD with respect to its open/closed state and its expiration date. The Privitar Platform provides visualization and API support to locate PDDs that an organization may deem to be risky.

Three important statuses are summarised here. The full list is given in the PDD Status Reference section below.

PDD status	PDD status summary	Suggested action
Expired	PDDs that are open and are past their expiration date.	The data is past its approved lifetime, so data consumers should stop using this data and the PDD should be closed.
No expiry date	PDDs that are open and have no expiration date set.	The data is being used in an unlimited way, which may be a source of increased risk over time. Data owners should either close the PDD or configure an appropriate time limit on its use.
Unknown	PDDs that are open and have an erroneous expiration date set.	There was a problem interpreting a specified expiration date value. This may happen if there are issues with an API integration with the Privitar Platform. System administrators should investigate.

Note that organizations should use their own policies to determine the actual risk of PDDs with these statuses.

6.6.1. Configure a metadata attribute for the expiration date

To configure a metadata attribute to contain the expiration date:

1. Select **Metadata Attributes** from the navigation bar.
2. If you would like to use an existing metadata attribute, click **Edit** on the relevant row. To create a new metadata attribute for this purpose, click on **Add New Metadata Attribute**. See Managing Metadata Attributes for further details regarding the creation of new metadata attributes.
3. On the **Details** screen, under the **Mapping** section, select *Expiration Date*.

   Note

   To be a valid *expiration date* , the metadata attribute should have a data type of Date.

4. Click on **Save**.

6.6.2. Using the PDD lifecycle visualization

To view a summary of all PDDs organized by lifecycle status, the Privitar Platform provides a visualization that supports a drill-down capability to find PDDs with various statuses.

To use the PDD lifecycle chart drill-down feature, use the following steps:

1. Select **Protected Data** from the navigation bar.
2. Click on the chart to filter the list of PDDs to just those with the selected status.

To obtain a quick overview of a specific status, use the following steps:

1. Select **Protected Data** from the navigation bar
2. Move the move over a section of the chart to retrieve its summary.

6.6.3. PDD Status Reference

PDDs may have the following statuses. Along with their descriptions are suggested actions.

Status	Description
Valid	The PDD is open, and the expiration date is set to a date in the future. The PDD data can be safely used.
Expired	The expiration date of a PDD is passed, and the PDD remains open. The user should close the PDD and discontinue use of the data. In some circumstances it may be appropriate to extend the expiration date.
Unknown	The current status of a PDD cannot be retrieved. This status is typically due to a technical configuration issue and will rarely appear. If it is shown, check the metadata attribute mapped to Expiration Date in the **PDD Metadata** configuration.
No expiration date	The PDD is open, but there is no expiration date set. The PDD data can be used without lifetime restrictions. The user should consider configuring an expiration date.
Closing	Closure of a PDD has been requested and is in progress. This is a temporary status that will lead to either *Closed* or *Failed to close*. The PDD is not available for de-identification processing anymore.
Closed	The PDD is not available for de-identification tasks anymore.

Status	Description
Failed to close	A PDD closure was attempted, and a problem was encountered. This is likely due to a technical configuration issue. If it is shown, contact your system administrator.
Deleted	The PDD has been removed and is not available anymore.

6.7. What is a Token Vault?

A Token Vault is a secure storage mechanism for tokens generated during de-identification. Each Token Vault is associated with a single Protected Data Domain.

When a Job requiring *consistent* tokenization is run, the PDD's Token Vault is populated with a mapping from the original source values to the tokens produced as replacements. Then, when multiple columns across different tables are masked with Rules with the **Preserve data consistency** option enabled, the same Token Vault - and hence the same mapping - is used to transform values across all the tables.

This is important because it means that the same tokenized value will be produced in both output tables for the same input value. In this way joining relationships based on values in those columns are preserved.

Consider the following dataset of users containing three fields; `ID`, `Name` and `Country`:

ID	Name	Country
12345	John Doe	US
73847	Jane Smith	UK
82375	Sally Jones	US

The columns in the dataset are tokenized using the **Regular Expression Text Generator** rule with the **Preserve data consistency** option enabled. The following table defines the regular expression rule that is applied to each column:

ID	Name	Country
`[0-9]{5}`	`[a-zA-Z]{7-9}`	`[a-zA-Z]{4}`

After applying the rules to the dataset, the de-identified dataset (in the Protected Data Domain) could be transformed as shown in the following table:

ID	Name	Country
23421	akAsdLn	YjqD
53627	yTbqnKas	QwRg
93464	obAsOias	YjqD

Note

As consistent tokenization has been used, the two entries for US have been tokenized with the same value (`YjqD`).

The same Token Vault is reused across successive runs of any Job targeting the same PDD. This means that if new data or different datasets are published to the PDD, any values that have already been tokenized in previous runs will be mapped to the existing tokens. This allows consistency to be maintained over time.

After a Token Vault has been populated by running a Job, it is possible to retrieve the original value for a token, provided the token itself is known and certain security requirements are met.

In order to be able to reveal (unmask) the mappings between the de-identified dataset and original dataset, the relationships are maintained by storing the mappings in the Token Vault. The Token Vault has a table, per rule, to store these mappings.

In the example above - which uses three rules - there are three tables in the token vault: **ID**, **Name** and **Country**.

ID - Token Vault	
Value	**Token**
12345	23421
73847	53627
82375	93464

Name - Token Vault	
Value	**Token**
John Doe	akAsdLn
Jane Smith	yTbqnKas
Sally Jones	obAsOias

Country - Token Vault	
Value	**Token**
US	YjqD
UK	QwRg

For the **ID** values, the de-identification flow from original values to the de-identified data is shown in the following diagram:

Original Data			**ID - Token Vault**			**Deidentified Data**	
ID	**Other Columns...**		**Value**	**Token**		**ID**	**Other Columns...**
12345	...	→	12345	23421	→	23421	...
73847	...	→	73847	53627	→	53627	...
82375	...	→	82375	93464	→	93464	...

For more information about finding the original values of de-identified data, see Unmasking De-identified Data.

6.8. Privitar NOVLT

Privitar NOVLT:

A feature of the Privitar Data Provisioning Platform that applies consistent tokenization without a token vault. NOVLT allows for data linkability across regions. NOVLT also offers faster throughput and less latency than most vaulted solutions.

6.8.1. When to Use a Token Vault or Privitar NOVLT

Use this table to determine when you should use a token vault or Privitar NOVLT.

Token Vault	NOVLT
structured or unstructured inputs	structured inputs
data is in a single region	data is in a single region or is spread across regions

Privitar NOVLT is best suited for inputs with a well-defined structure, such as IDs, Social Security numbers, account numbers, and so on. Avoid using NOVLT for less-structured inputs, such as names, postal addresses, and email addresses.

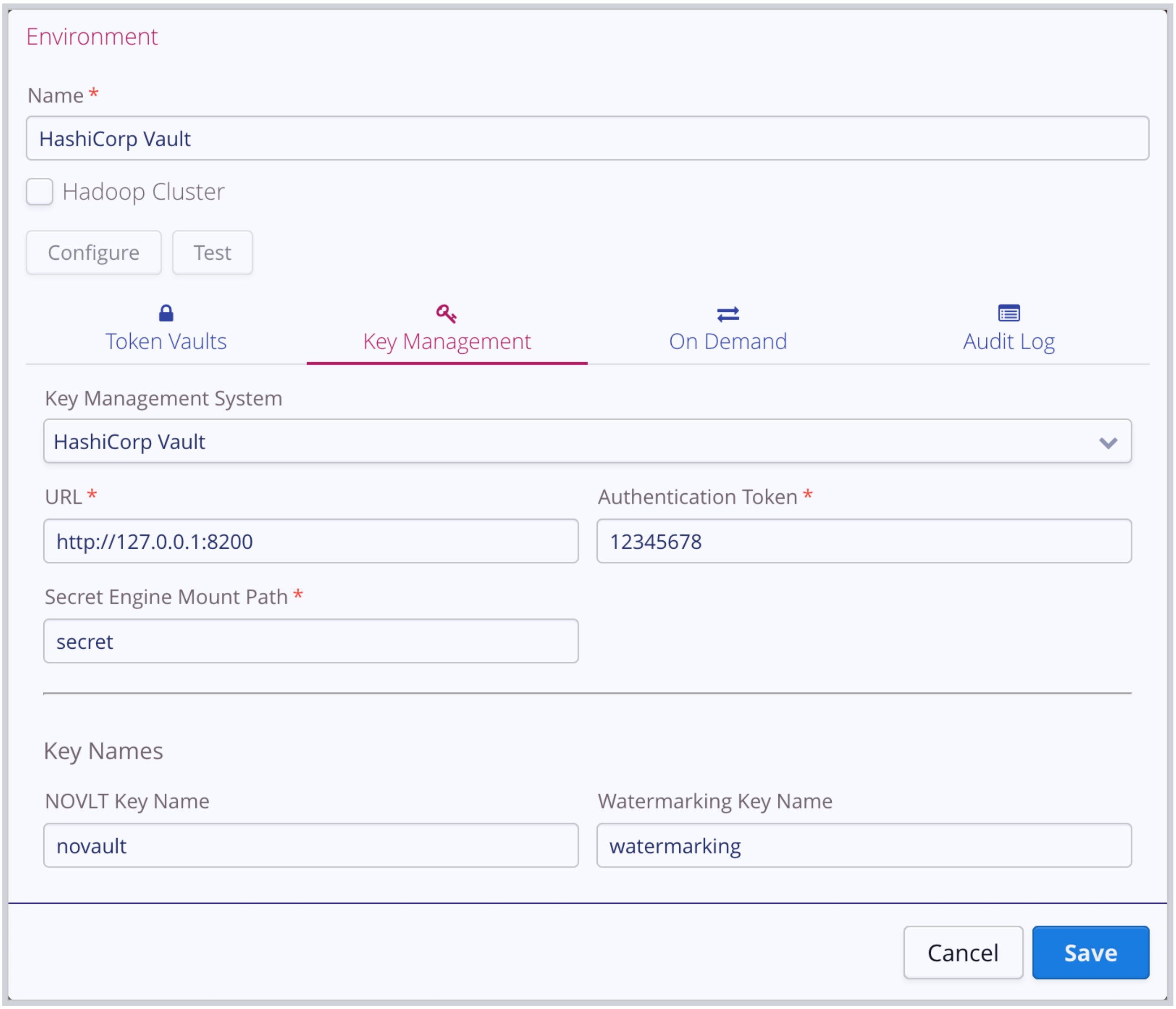
If you were to use NOVLT for unstructured inputs, the unpredictable length of potential inputs leads to output tokens that are impractically large. For example, an email address token might appear as follows:

```
sdilufhasku123dhfasldiu4fhaskdfjh.qweiu23jflasjd2sdfasdf23kflas4dkfj-
sdflj12nqoiskd12asdfjfasld59kfj@isdlwesdfasdjh82jld.10lsldjhfsdlkjf.c
om
```

6.8.2. Enabling NOVLT

To enable NOVLT:

1. In the Environments settings, click the **Key Management** tab.
2. In the Key Management System field, select a KMS.
3. In the URL field, enter the URL where the platform can contact the KMS.
4. In the Authentication Token field, enter the token ID.
5. In the Secret Engine Mount Path field, enter the path for the KV Secrets Engine.
6. In the NOVLT Key Name field, enter the key name.

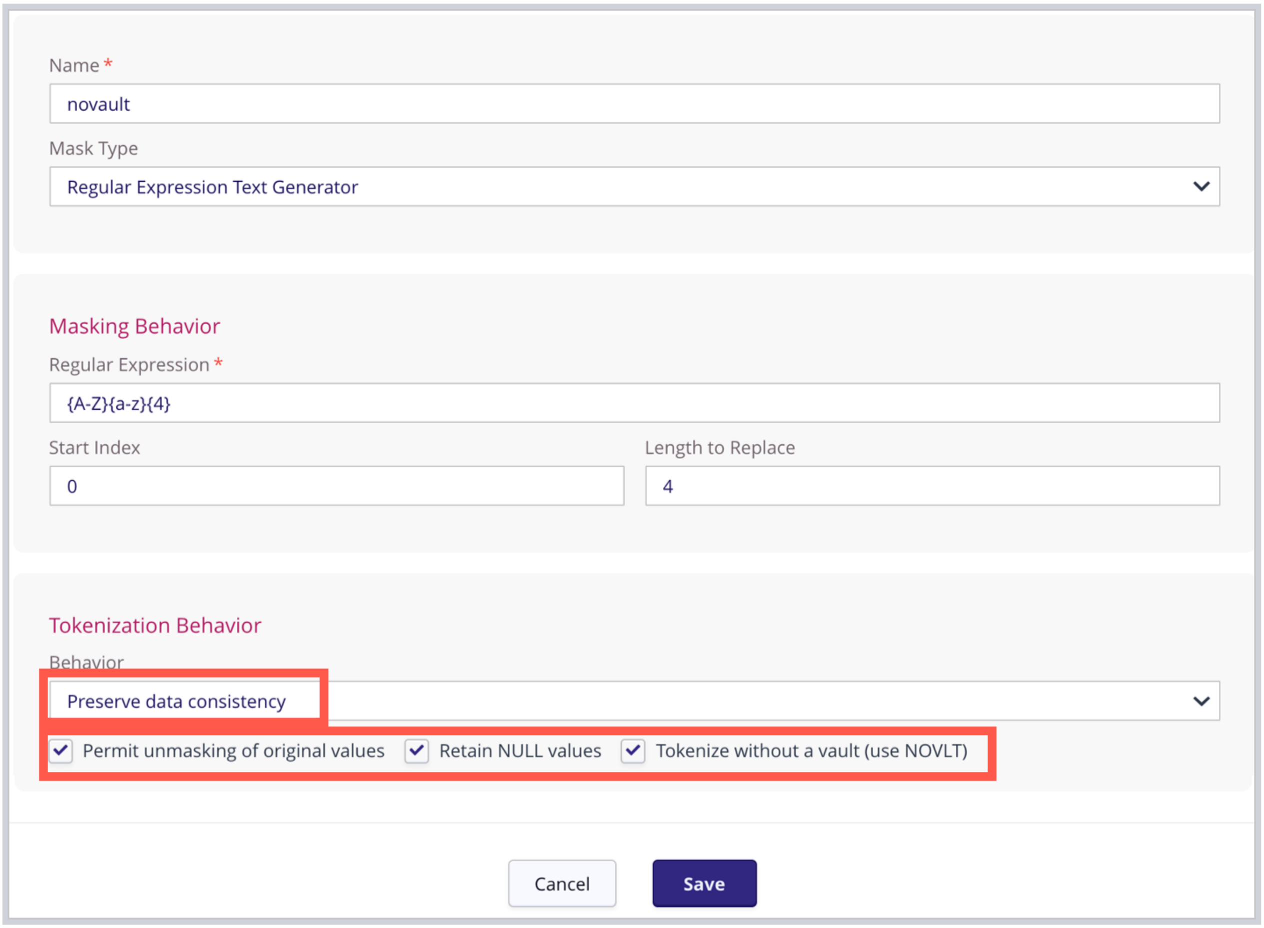


6.8.3. Configuring NOVLT on a Rule

You can use Privitar NOVLT with the following rules:

- Regular Expression Text Generator
- Regular Expression Number Generator

On the View masking rule page, select “Preserve data consistency” and check both the “Retain NULL values” and the “Tokenize without a vault (use NOVLT)” boxes.



Name *	novault
Mask Type	Regular Expression Text Generator

Masking Behavior

Regular Expression *	{A-Z}{a-z}{4}
Start Index	0
Length to Replace	4

Tokenization Behavior

Behavior	Preserve data consistency

- ✓ Permit unmasking of original values
- ✓ Retain NULL values
- ✓ Tokenize without a vault (use NOVLT)

Note that all input values must match the regular expression specified in the Regular Expression field.

Caution

Do not change the regular expression of a rule that you have previously used in jobs. Doing so would break consistency with all previously tokenized data. This is true even if you are simply extending the regular expression, for example, changing from [a-z]{1,3} to [a-z]{1,4}.

6.8.4. Key Management for NOVLT

Key Lengths and Key Algorithm

The NOVLT key must be 16, 24, or 32 bytes long (128, 192, or 256 bits). We recommend using 32-byte keys.

The NOVLT key is an Advanced Encryption Standard (AES) key. You can use any securely random data of the correct length.

For AWS-native deployments, start the name of the NOVLT key with `privitar-novlt-`. Create this key manually before trying to use it in the same account as the deployment.

6.8.5. Protection of Data Integrity with NOVLT

The Privitar NOVLT base secret is used to generate unique encryption keys for each rule and PDD. These unique, generated keys ensure consistency of tokenized output and also allow for unmasking of tokenization. The encryption algorithm is further strengthened using randomly generated salts; salt values are stored in the configuration database of the platform's control plane. The salt values should be a base64 encoded string of a maximum of 32 bytes.

As added security, the salt and base secret are stored separately (the base secret is stored in the KMS/secrets manager, and the salt is stored in the control plane configuration database). An attacker could only remove tokenization from previously generated output data if the attacker compromises both the secrets manager and the configuration database.

In the event that an attacker compromises a base secret, you can rotate the base secret.

6.8.6. Rotating Base Secrets

The base secrets are versioned, and individual PDDs are linked to specific versions of a base key. When you rotate a secret, you generate a new version of the secret that will be used to tokenize data in any new PDDs created after rotation.

7. Creating and Running Jobs

This part of the User Guide describes how to create and run Privitar Jobs.

7.1. What is a Job?

A Privitar Job defines the application of a Policy to some data, with the goal of publishing the output into a Protected Data Domain.

PDDs are selected when creating or running the Jobs to control the consistency of the output data, and to optionally associate the processed data with a Watermark.

It represents the concrete execution of the data transformations defined in the Policy.

There are three types of Jobs:

- **Batch Jobs** apply a Policy to datasets as batch operations. They are either executed on Hadoop clusters, reading and writing data in HDFS / cloud-based blob storage services or Hive, or on AWS Glue ETL, reading and writing data in Amazon S3.
- **Data Flow Jobs** apply a Policy to flows of streaming or batches of data. They are executed on external data flow platforms. Input and output data flows are configured in the external data flow platform.
- **Privitar On Demand Jobs** apply a Policy to batches of data received by a Privitar On Demand server, over a HTTP(s) API. The result of applying the Policy is returned to the caller of the API.

Batch Jobs apply a Policy to datasets as batch operations. These Jobs are executed:

- on Hadoop clusters, reading and writing data at rest in HDFS, Hive or cloud-based blob storage services, such as Amazon S3, Azure Blob Storage or Google Cloud Storage, or
- on AWS Glue ETL, a serverless data processing service in AWS, reading and writing data on Amazon S3.

Data Flow Jobs apply a Policy to flows of streaming data or batches of streaming. These Jobs are executed on external data flow platforms. Input and output data flows are configured in the external data flow platform.

Privitar On Demand Jobs apply a Policy to batches of data received by a Privitar On Demand server, over a HTTP(s) API. The result of applying the Policy is returned to the caller of the API.

7.2. What are Batch Jobs?

Batch Jobs apply a Policy in bulk on datasets located on a cluster. They contain a reference to a Policy and the location of specific input data. Batch Jobs are defined and executed from Privitar or via the Policy Manager Automation APIs.

It is possible to monitor progress and obtain potential error details of Batch Jobs. They also serve as a way to re-run specific processing if new data is received or if there is a change to a Policy.

The Schema used by the Policy contains a reference to the data item that is to be read, and the Batch Job points to its actual location. Batch Jobs are started from Privitar, but they are physically executed on the compute infrastructure (either Hadoop or AWS Glue) configured in the Privitar Environment.

When the Batch Job is run, the Policy is applied to the data and the resulting output data is published to a specific Protected Data Domain (PDD), in the PDD's output directory or Hive database.

After a Batch Job has run, its Job definition remains as a record in Privitar. This means that at any time the Batch Job can be re-run on new data. This is useful in the case where the source data has changed, new data was added or the Policy was updated and the privacy processing performed by the Batch Job needs to be repeated.

Running Batch Jobs against the same PDD guarantees data consistency (if the **Preserve data consistency** option is used) for each rule used in the Policy.

There are two types of Batch Job that can be run from Privitar:

- **HDFS Batch Jobs** process data from an HDFS location and write out the processed data into a PDD that is also stored in an HDFS location.
- **Hive Batch Jobs** process data stored in a Hive table or Hive view and write out the processed data into a PDD that is stored in the Hive database as a Hive table.
- **AWS Glue Batch Jobs** process data from an Amazon Simple Storage Service (Amazon S3) location and write out the processed data into a PDD that is also stored in an Amazon S3 location.

7.2.1. HDFS Batch Jobs

HDFS Batch Jobs are created to process data stored as Avro, Parquet, or CSV files in HDFS. HDFS Batch jobs can also process data stored on equivalent filesystems in the Cloud such as:

- Amazon S3 buckets using EMRFS
- Google Cloud Storage (GCS)
- Microsoft Azure Blob storage

Bad Record Handling

Batch Jobs can be configured to detect errors in execution and stop if some or all records cannot be processed due either to bad data or misconfiguration.

Partitioned Data

Often data will be split into manageable chunks based on a scheme around a variable such as ingest date or country of origin. Where this data is organised hierarchically in HDFS, Privitar can respect this partitioned layout and process data appropriately.

Empty and Hidden Files Handling

Empty files and hidden files (files in hidden directories) will be skipped during processing. If all files/directories in the input path are empty or hidden, a Job will fail.

7.2.2. Hive Batch Jobs

Hive Batch Jobs are created to process data in the cluster that can be accessed via Hive.

Read and write data via Hive

Data is accessed from a cluster using Hive, and is written directly to a Hive database. Privitar creates safe output data in a form that is ready for immediate use by data consumers in a way that is consistent with their existing data access patterns.

Efficient processing of partitioned data

Partitioned data is naturally represented as database columns in Hive, so handling of such data requires no special consideration when creating the Privitar Schema or Policy.

Inferring Schemas from Hive

If inferring a Schema from Hive, Privitar will automatically infer all the tables and columns present from the location and create default names for the Schema to match the names of the Hive tables and columns.

Support for querying Hive views

Privitar can query Hive views as readily as Hive tables, allowing for more flexible application of privacy protection to subsets of data that are defined as Hive views.

Support for filtering of data

Privitar supports row-level filtering of Hive data. For each column (both data and partition columns), a filter can be set to enable specific subsets of the data to be processed. The filters that can be applied are SQL-like query commands such as Greater than, Less than. For example, a filter could be applied to a column containing date information to only include rows of data from a specific date.

7.2.3. AWS Glue Batch Jobs

Batch Jobs running in an AWS Glue Environment will run using the AWS Glue Service.

AWS Glue Batch Jobs can read and write data to Amazon Simple Storage Service (Amazon S3) buckets.

The AWS Glue ETL service is used as a serverless data processing environment to run AWS Glue Batch Jobs. This means that Privitar submits data processing jobs to the AWS Glue service, which will run them on ephemeral infrastructure managed by AWS.

Writing to Amazon S3 Buckets

When setting the **Output Path** within a **Protected Data Domain (PDD)** and running a Privitar AWS Glue Batch Job, you should ensure that you only use an Amazon S3 bucket that you consider a safe and secure destination for your data.

To protect users from writing to unsanctioned Amazon S3 buckets in other AWS Accounts to the Privitar deployment (with a Privitar AWS Glue Batch Job), administrators must configure a list of Amazon S3 buckets that can be accessible. You might need to contact a system administrator to modify this configuration.

When output data is written to an Amazon S3 bucket that does not reside within the same AWS account as the Privitar Data Privacy Platform deployment, Privitar AWS Glue Batch Jobs automatically enables this output data to be read by the destination AWS account. This ensures that the data is accessible to end users operating from the destination AWS account. You should ensure that you only use Amazon S3 buckets that are owned by AWS accounts you know.

These Amazon S3 objects are created with the `bucket-owner-full-control` canned access control list (ACL) applied.

You can learn more about Amazon S3 Object Ownership in the AWS documentation (https://docs.aws.amazon.com/AmazonS3/latest/userguide/about-object-ownership.html).

7.3. Overview of HDFS Batch Jobs

Running an HDFS Batch Job involves specifying a Policy together with the location of the data to be processed, applying the Policy to that data and then publishing the output to a Protected Data Domain (PDD).

To create and run an HDFS Batch Job, follow these steps:

1. Check that the HDFS Environment has been set up correctly. (See ???.)
2. Import the Schema from HDFS. (See, Specifying Input Data Locations (HDFS and Cloud Storage).)
3. Create the Policy. (See, Creating a Policy.)
4. (Optional step). If not using an existing PDD, create a new PDD. (See, Creating a PDD.)
5. Create the Batch Job. (See, Creating a Batch Job).
6. Choose a name for the Batch Job and specify the Policy to use. (See, Choosing a Name and Policy for a Job.)
7. Specify the input Data Locations. (See, Specifying Input Data Locations (HDFS and Cloud Storage).)
8. Check any of the Advanced settings. (See, Advanced settings (HDFS Batch Jobs).)
9. Run the Batch Job and select the location for the PDD. (See, Running a Batch Job.)
10. Review the output of the Job. (See, Batch Job Output Visualisations (HDFS and Hive).)

7.3.1. Checking HDFS Environment setup

The HDFS Environment must be set up correctly in Privitar before an HDFS Batch Job can be run. In particular, you must have read access for the HDFS location you are reading data from and have write access to the data location where you will be writing out the processed data.

If you are not sure about the HDFS setup, contact your local system administrator before running an HDFS Batch Job. For more information, refer to Hadoop Cluster Environment Configuration.

7.4. Overview of Hive Batch Jobs

Running a Hive Batch Job involves specifying a Policy together with the location of the data to be processed, applying the Policy to that data and then publishing the output to a Protected Data Domain (PDD).

To create and run a Hive Batch Job, follow these steps:

1. Check that the Hive Environment has been set up correctly. (See below).
2. Import the Schema from Hive. (See, Creating a Schema from a Hive database.
3. Create the Policy. (See, Creating a Policy.)
4. (Optional step). If not using an existing PDD, create a new PDD. (See, Creating a PDD.)
5. Create the Batch Job. (See, Creating a Batch Job).
6. Choose a name for the Batch Job and specify the Policy to use. (See, Choosing a Name and Policy for a Job.)
7. Specify the input Data Locations. (See, Specifying Input Data Locations (Hive))
8. Check any of the Advanced settings. (See, Advanced Settings (Hive Batch Jobs))
9. Run the Batch Job and select the location for the PDD. (See, Running a Batch Job.)
10. Review the output of the Job. (See, Batch Job Output Visualisations (HDFS and Hive).)

7.4.1. Checking Hive Environment Setup

The Hive Environment must be set up in Privitar before a Hive Batch Job can be run. This will have been setup by your system administrator. (For more information, see Hadoop Cluster Environment Configuration.)

In particular, the Job output root location setting determines how the Hive PDD location is specified when you run a Batch Job. The PDD location in the Hive Environment can be defined in two parts:

- Hive Database; the name of the Hive database in which the tables will be created.
- HDFS location; the path in HDFS (or supported Cloud location) where the data for the Hive tables will be stored.

The Job output root location setting can be used to define a single location for all Hive PDDs. If this location has been defined, then the HDFS location path for the Hive PDD will not need to be specified when running a Hive Batch Job.

If you are not sure about the Hive setup, contact your local system administrator before running a Hive Batch Job.

7.5. Overview of AWS Glue Batch Jobs

Running an AWS Glue Batch Job involves specifying a Policy together with the location of the data to be processed, applying the Policy to that data and then publishing the output to a Protected Data Domain (PDD).

Note

AWS Glue Batch Jobs can only be defined on Policies defined on single-table schemas.

To create and run an AWS Glue Batch Job, follow these steps:

1. Check that the AWS Glue Environment has been set up correctly. (See ???.)
2. Import the Schema. (See, Creating a Schema from an AWS Glue Data Catalog.)
3. Create the Policy. (See, Creating a Policy.)
4. (Optional step). If not using an existing PDD, create a new PDD. (See, Creating a PDD.)
5. Create the Batch Job. (See, Creating a Batch Job).
6. Choose a name for the Batch Job and specify the Policy to use. (See, Choosing a Name and Policy for a Job.)
7. Specify the input Data Locations. (See, Specifying Input Data Locations (HDFS and Cloud Storage).)
8. Check any of the Advanced settings. (See, Advanced settings (AWS Glue Batch Jobs).)
9. Run the Batch Job and select the location for the PDD. (See, Running a Batch Job.)
10. Review the output of the Job. (See, Batch Job Output Visualisations (HDFS and Hive).)

7.5.1. Checking AWS Glue Environment setup

The AWS Glue Environment must be set up correctly in Privitar before an AWS Glue Batch Job can be run. In particular, you must have deployed the Privitar Platform in an AWS environment, with the correct permissions to access the data and submit Jobs to the AWS Glue service.

If you are not sure about the AWS Glue setup, contact your local system administrator before running an AWS Glue Batch Job. For more information, refer to AWS Glue Environment Configuration.

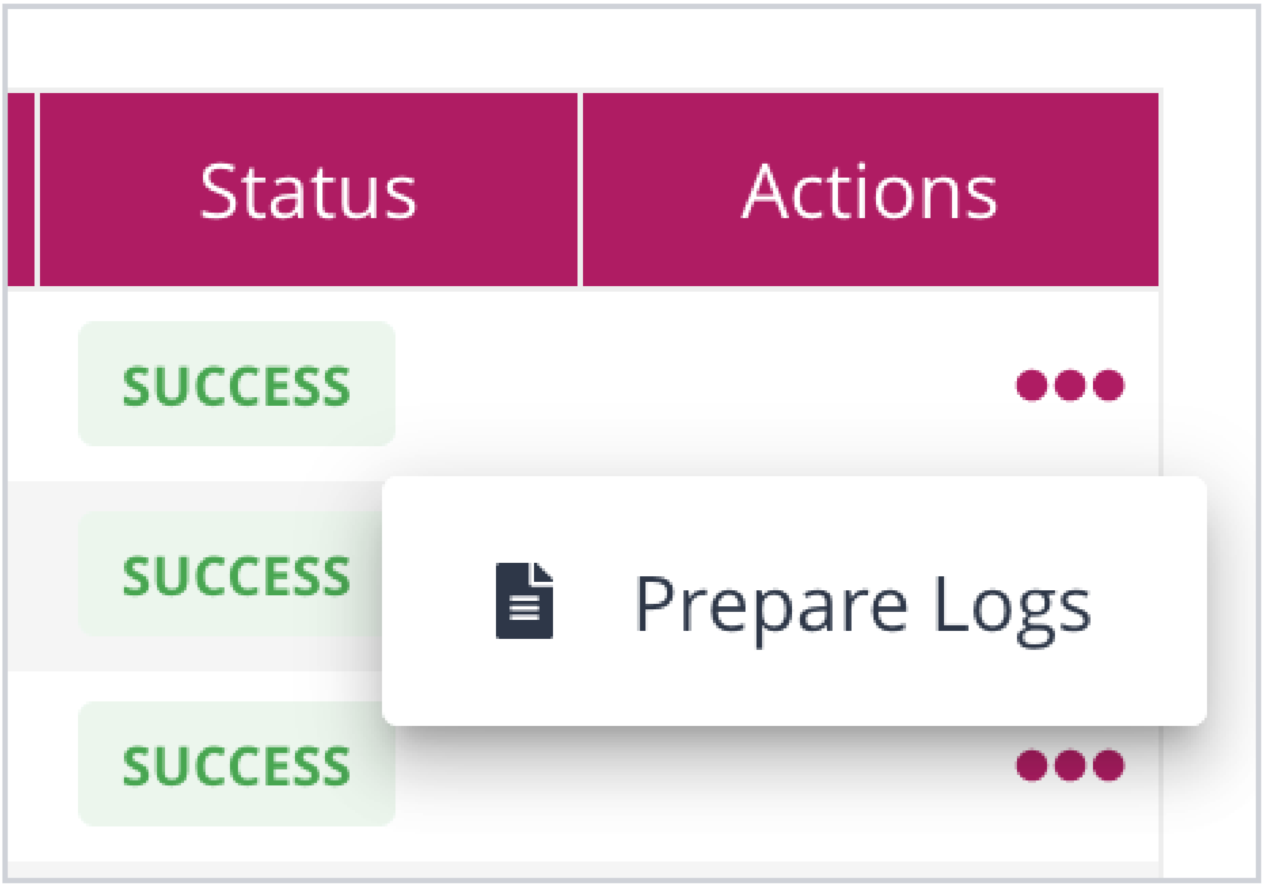
7.5.2. Downloading AWS Glue Logs

You can download logs produced as part of an AWS Glue batch job from the Job listing view.

Note

The Policy Manager collects logs from AWS. AWS states that logs may not be available for collection for up to 12 hours, but are typically available sooner. Therefore there may be a delay before you can download logs.

You must first perform an action to prepare the logs for download.



1. On the Jobs page, select a job.
2. On the View Job page, click the menu in the Actions column next to the job.
3. Click **Prepare Logs**.

When the logs have been prepared and are ready to download, the option will change to **Download Logs**. Logs remain available to download for 24 hours. After 24 hours, you need to click **Prepare Logs** again.

7.6. Creating a Batch Job

A Batch Job contains a reference to a Policy and the location of specific input data. It lets the Policy be applied to that data, publishing the output to a specified Protected Data Domain.

In addition to the Privitar user interface and Automation API, Jobs can also be imported from a previously exported file. See, Importing an Exported Job for more information.

To create a Batch Job to process specific input files, ensure that the location of the required data is known (paths to the required files, or a Hive database name), and that the Policy already exists and is not in Draft. Then take the following steps:

1. Select **Jobs** from the Navigation sidebar. The Jobs page is displayed.
2. Select the **Batch Jobs** tab.
3. Choose the Environment from the **Environment** list box.

   The chosen Environment must contain the data to be processed by the Policy, and will also be the destination of the processed data.

4. Click on **Create New Batch Job**. If asked select either **Hive** or **HDFS** from the list box. The **New Batch Job** window is displayed.

Please keep in mind that different types of Batch Jobs will be available, depending on the Environment in which you are creating the Job:

- AWS Glue Batch Jobs can only be created in AWS Glue Environments
- HDFS and Hive Jobs can only be created in Environments with an Hadoop cluster configured.

7.6.1. Choosing a Name and Policy for a Job

To set up a name and Policy for a new Job:

1. Enter a name for the Job in the **Name** box from the **New Batch Job** window. This name is used to identify the Job in the Privitar user interface and via Automation APIs.
2. Select a compatible, valid Policy to use for this Job from the **Policy** list box. This will determine what masking rules will be applied to the data processed by the Job.

   Note

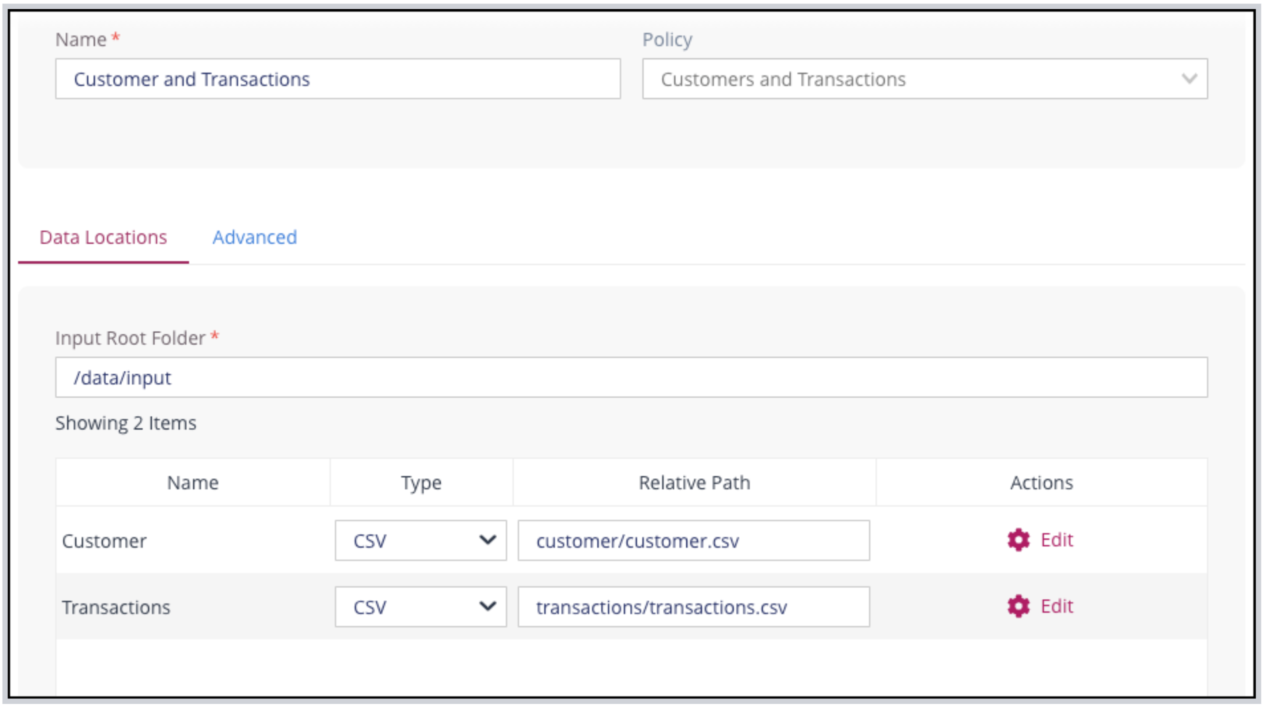
   Policies that are applied to Schemas with optional columns are not compatible with Batch Jobs.

The next step depends on whether an HDFS or Hive Batch Job is being created:

- HDFS Batch Job: Specify Input Data Locations. See, Specifying Input Data Locations (HDFS and Cloud Storage).
- Hive Batch Job: Specify Input Data Locations. See, Specifying Input Data Locations (Hive).

7.6.2. Specifying Input Data Locations (HDFS and Cloud Storage)

The **Data Locations** section enables the locations of input data to be specified. The data items referenced in this section must conform to the Schema of the Job's Policy.


Name *
Customer and Transactions
Policy
Customers and Transactions

Input Root Folder *
/data/input
Showing 2 Items

Name	Type	Relative Path	Actions
Customer	CSV	customer/customer.csv	Edit
Transactions	CSV	transactions/transactions.csv	Edit

To complete the Data Locations section:

1. Specify a valid pathname in the **Input Root Folder** box. All files to be read by the Job must reside under this location.
2. For each table displayed, specify the following:
   - Type of file in the **Type** column. This can be one of **CSV**, **Avro** or **Parquet**.
   - The path to the table in the **Relative Path** box.

     The files representing the input are referred to using a path that is matched relative to the folder specified in the **Input Root Folder** box, and may represent either a single file, or multiple files if wildcards are used. For more information about Relative Paths, see Partitioned Data and Wildcards (HDFS).

   If the selected input type is **CSV** or **Avro**, additional Settings for these file types can be configured by clicking on the *Cog* icon in the **Settings** column.

Reading from Cloud Storage

When using Privitar in a Cloud Environment, native Cloud storage platforms can be directly used to read and write data with Batch Jobs. This is configured from the **Data Locations** section of the Batch Job.

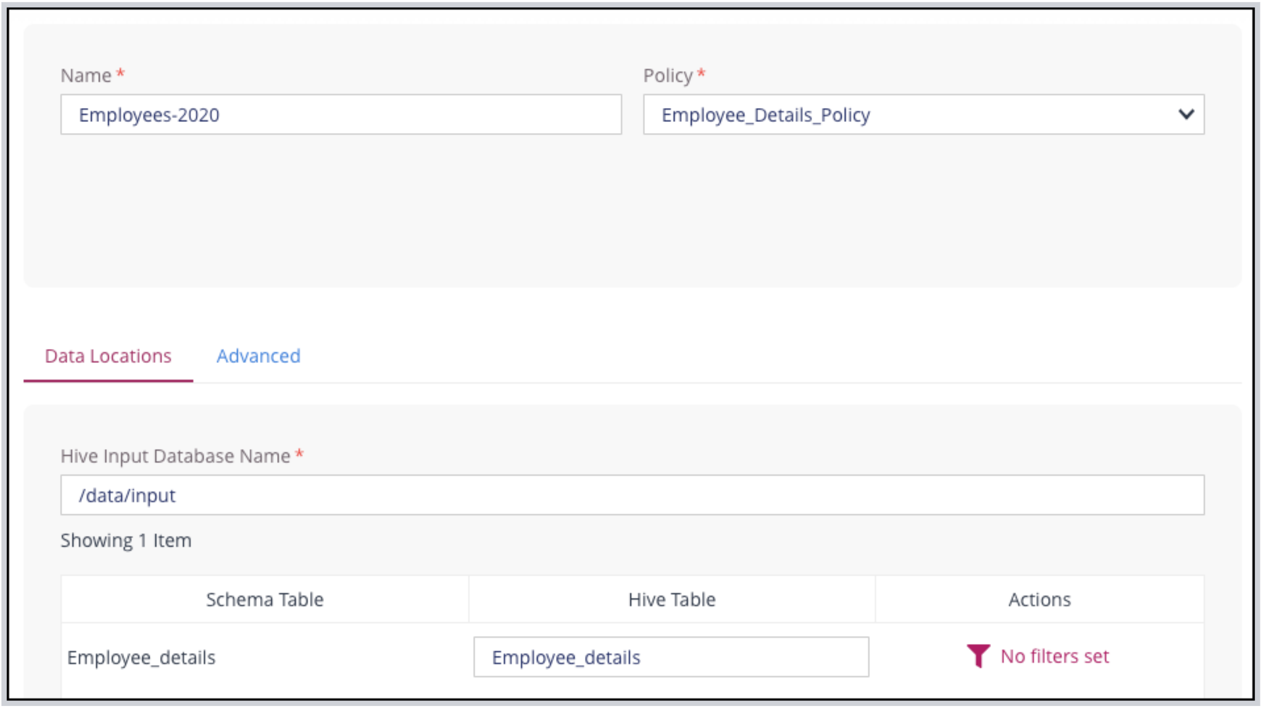
The following table defines the syntax to use when specifying a data location for the Cloud storage platforms that are supported by Privitar.

Compute Platform	Container	Location syntax
Amazon EMR Cluster	S3	`s3://<bucket-name>/`
AWS Glue	S3	`s3://<bucket-name>/`
Azure HDInsight	Blob	`wasb://<container-name>@<account-name>.blob.core.windows.net/`
Google Dataproc	Bucket	Prefix path with: `gs://`

Compute Platform	Container	Location syntax
BlueData	DataTap	Prefix path with: `dtap://`

7.6.3. Specifying Input Data Locations (Hive)

The Data Locations section allows the locations of input data to be specified. The Hive tables referenced in this section must conform to the Schema of the Job's Policy.



Name *: Employees-2020

Policy *: Employee_Details_Policy

Hive Input Database Name *: /data/input

Showing 1 Item

Schema Table	Hive Table	Actions
Employee_details	Employee_details	No filters set

To complete the Data Locations section:

1. Specify a valid database name in the **Hive Input Database Name** edit box. All data to be read by the Job must reside in this database as a Job can only be run on one database.
2. Check that the name of the Hive Table matches the name of the corresponding Schema Table name for the Hive table that you are going to use.

   Privitar uses the Schema Table name from the Privitar Schema, but if there is a mis-match, update the Hive Table name to match the table name of the underlying schema. (The table name can be checked by examining the Hive instance using a suitable database analysis tool such as **Hue**.)
3. If necessary, apply a filter to the Hive table that you are going to use.

   Applying a filter enables a subset of the data in the Hive table to be processed. For more information, see Filtering Hive Data.

7.6.4. Advanced settings (HDFS Batch Jobs)

The Advanced tab contains advanced settings to configure HDFS Batch Jobs.

Overwrite Behavior

Use the **Overwrite Behavior** list box to define the behavior in the event of existing files being encountered during the execution of the Job.

The following table defines the options available:

Option	Description
Always overwrite existing data	Existing files encountered during Job execution will be silently overwritten by Privitar.
Job fails if data aleady exists	Existing files encountered during Job execution will cause the Job to fail. This behavior may be useful when Privitar is controlled via the Automation API. When using the API, data being overwritten is not expected and should be flagged as a production issue.

Handling of Bad Records

Use the **Bad Record Handling** dropdown menu to define the behavior in the event of bad records being received during the execution of the Job.

The following table defines the options available:

Option	Description
Fail when all records are bad	The Job is only deemed to have failed when no records in the data input could be successfully processed.
Fail when bad record percentage exceeds threshold	The Job is deemed to have failed when greater than the specified percentage of records could not be processed.
Fail when any record is bad	The Job is deemed to have failed when one or more records could not be processed.

Note

In the event that data could not be processed, the details of the failure are displayed in the **Details** column in the **View Job** window. This window can be accessed by clicking on the name of the Job in the **Jobs** page.

Spark Parameters

The Spark settings determine parameters for submitting Spark jobs used to perform anonymization processing on the Privitar platform. The best settings for these parameters are dependent on the specification of the Hadoop cluster and the characteristics of input data.

Typically these settings are specified centrally on the Job's selected Environment. For more information, see Hadoop Cluster Environment Configuration. Privitar also supports overriding these settings on a per-Job basis.

7.6.5. Advanced Settings (Hive Batch Jobs)

The Advanced tab contains advanced settings to configure Hive Batch Jobs.

Overwrite Behavior

Use the **Overwrite Behavior** list box to define the behavior in the event of existing files being encountered during the execution of the Job.

This setting is useful when the Job is being used for a purpose where a strict, known output is required. For example, it may be necessary to know that the table contains only the result of the current Job run, and never the results of previous runs. In this case, the Truncate option would be used.

The table defines the options available:

Option	Description
Always insert new rows into existing tables	If the Hive table exists, insert into it, merging new records with existing data.
Fail if the tables already exist	If the Hive table exists, fail the Job. The table must not exist when the Job runs.
Always truncate existing tables before writing	If the Hive table exists, erase (truncate) it before writing any new records. This ensures that the table only contains the content of the most recent Job run.

Handling of Bad Records

Use the **Bad Record Handling** list box to define the behavior in the event of bad records being received during the execution of the Job.

The table defines the options available:

Option	Description
Fail when all records are bad	The Job is only deemed to have failed when no records in the data input could be successfully processed.
Fail when bad record percentage exceeds threshold	The Job is deemed to have failed when greater than the specified percentage of records could not be processed.
Fail when any record is bad	The Job is deemed to have failed when one or more records could not be processed.

Note

In the event that data could not be processed, the details of the failure are displayed in the **Details** column in the **View Job** window. This window can be accessed by clicking on the name of the Job in the **Jobs** page.

Spark Parameters

The Spark settings determine parameters for submitting Spark jobs used to perform anonymization processing on the Privitar platform. The best settings for these parameters are dependent on the specification of the Hadoop cluster and the characteristics of input data.

Typically these settings are specified centrally on the Job's selected Environment. For more information, see Hadoop Cluster Environment Configuration. Privitar also supports overriding these settings on a per-Job basis.

7.6.6. Advanced settings (AWS Glue Batch Jobs)

The **Advanced** tab contains advanced settings to configure AWS Glue Batch Jobs.

Overwrite Behavior

Use the **Overwrite Behavior** list box to define the behavior in the event of existing files being encountered during the execution of the Job.

The following table defines the options available:

Option	Description
Always overwrite existing data	Existing files encountered during Job execution will be silently overwritten by Privitar.
Job fails if data aleady exists	Existing files encountered during Job execution will cause the Job to fail. This behavior may be useful when Privitar is controlled via the Automation API. When using the API, data being overwritten is not expected and should be flagged as a production issue.

Handling of Bad Records

Use the **Bad Record Handling** dropdown menu to define the behavior in the event of bad records being received during the execution of the Job.

The following table defines the options available:

Option	Description
Fail when all records are bad	The Job is only deemed to have failed when no records in the data input could be successfully processed.
Fail when bad record percentage exceeds threshold	The Job is deemed to have failed when greater than the specified percentage of records could not be processed.
Fail when any record is bad	The Job is deemed to have failed when one or more records could not be processed.

Note

In the event that data could not be processed, the details of the failure are displayed in the **Details** column in the **View Job** window. This window can be accessed by clicking on the name of the Job in the **Jobs** page.

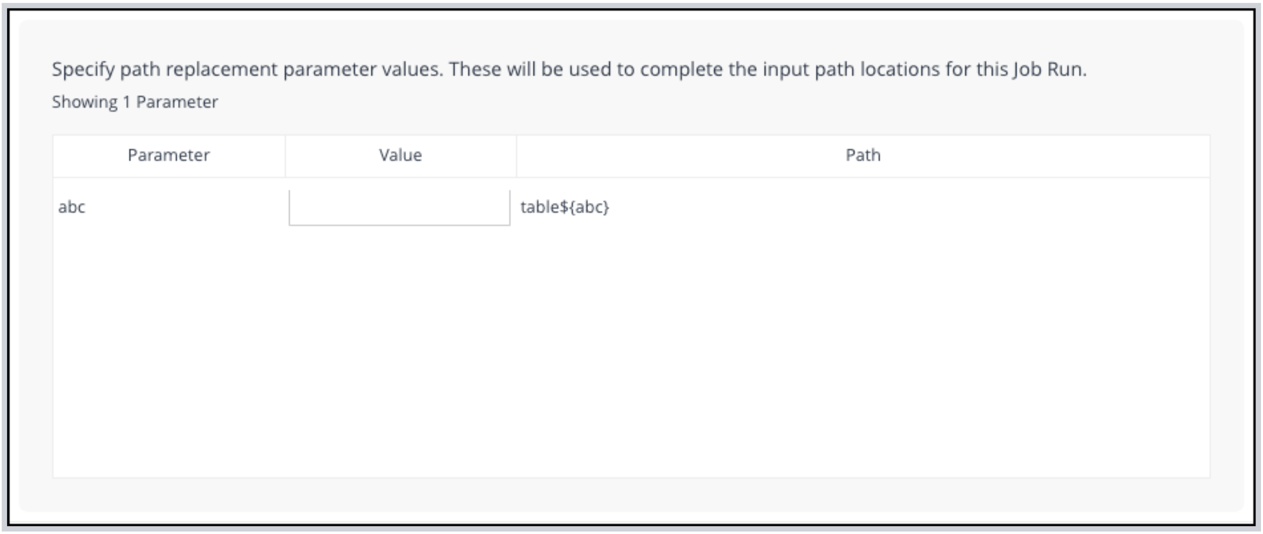
7.7. Running a Batch Job

Once created, Batch Jobs may be run from the Jobs list.

1. Select **Jobs** from the Navigation sidebar. The **Jobs** page is displayed.
2. Select the **Batch Jobs** tab.
3. Select the Environment where the Job definition is defined, from the **Environment** list box.
4. Find the Job that you want to run. Use the **Filter** search box to filter the Jobs if required.
5. Click on the *Play* icon in the end column alongside the Job that you want to run. The **Run Job** window is displayed.
6. Select the PDD that the Job will output data to from the **Add to an existing Protected Data Domain** list box.

   Or, select **Create new PDD** to create a new PDD. Enter a name for the new PDD in the edit box along with all the other required details and click **Save**. (For more information, see Creating a Protected Data Domain.)
7. Click on the checkbox alongside each Schema table that you want to include in the Job.

   Not all tables need be included. Only data corresponding to the selected tables will be processed.
8. (*Optional step*) If there are any values that need to be specified for either **Filters** (for Hive Batch Jobs) or **Path replacements** (for HDFS Jobs), a window is displayed asking for the values to be specified. For example:



Specify path replacement parameter values. These will be used to complete the input path locations for this Job Run.
Showing 1 Parameter

Parameter	Value	Path
abc		table${abc}

   For more information, see:

   - For more information about setting up Filters, see Setting Filters for a Hive Batch Job
   - For more information about setting up Path replacements, see Partitioned Data and Wildcards (HDFS).
9. (*Optional step*) If the Privitar `application.properties` file has enabled the option to provide advanced diagnostic output for Jobs, an **Advanced** tab is displayed. Select this tab to configure the output.

For more information about the options available from the **Advanced** tab, see Setting Diagnostic Options for Jobs (HDFS & Hive Batch Jobs).

10. Click on **Run** to start the Job.

7.7.1. Advanced Options

The settings described in this section are optional and do not need to be set in normal use.

Logging properties

Additional *log4j* Apache logging properties may be specified for the Job run.

Maximum Partition Size

Maximum Partition Size controls how much of an input file is processed in-memory at once by a Spark Executor. This setting can be used to control memory usage in the active Spark Executors. This setting is optional and should only be set if needed.

As an approximation, this value should be set according to the algorithm:

```
Maximum Partition Size < Executor Memory / (2 × Executor Cores)
```

AWS Glue Parameters

This option is only provided for Privitar AWS deployments.

The **Executors** option refers to the number of Spark executors that AWS Glue Batch Jobs will request to run a Batch job.

Note

Increasing this number will increase the cost of running this job in AWS. Contact your system administrator if you are in any doubt.

7.7.2. Setting Diagnostic Options for Jobs (HDFS & Hive Batch Jobs)

The **Advanced** tab on the **Run Job** window contains various options that can be used to generate diagnostic information about the running of a Job.

To enable the options, select the **Specify Diagnostic Options** checkbox. The window will display the options available.

The options are split into three sections:

- Spark
- Logging
- Metrics

Note

The **Advanced** tab is only displayed in the **Run Job** window if the Privitar `application.properties` file has been configured to provide advanced diagnostic output for Jobs.

Spark

There are various options that can be selected for the Spark execution.

Maximum Partition Size

The **Maximum Partition Size** controls how much of an input file is processed in-memory at once by a Spark Executor. This setting can be used to control memory usage in active Spark Executors. This setting is optional and should only be set if needed.

As an approximation, this value should be set according to the formula:

```
Maximum Partition Size < Executor Memory / (2 × Executor Cores)
```

Write Spark Events File

Select this checkbox to write to an events file all the Spark events that are generated by Spark during processing.

Log Garbage Collection Details

Select this checkbox to write to the log file all the details about garbage collection performed during Spark processing.

Spark Driver and Executor Options

Enter additional JVM options to pass to the Spark driver, in the `spark.driver.extraJavaOptions` edit field.

Enter additional JVM options to pass to the Spark Executor, in the `spark.executor.extraJavaOptions` edit field.

Logging

Any additional *log4j* logging properties to pass to Spark may be specified in this section.

Metrics

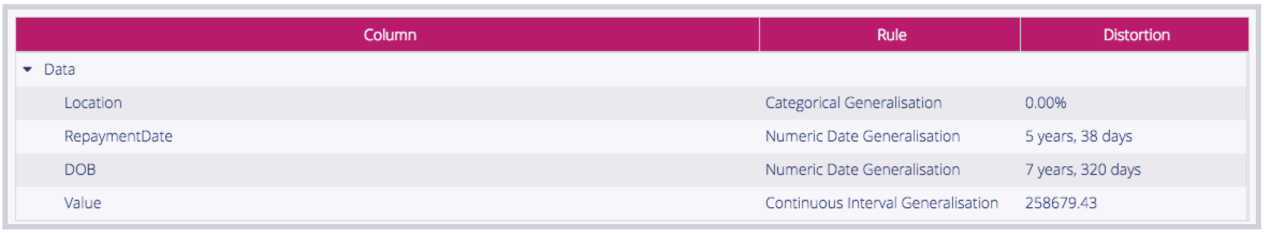
Any additional Metrics may be specified in this section for the Spark job.

7.8. Batch Job Output Visualisations (HDFS and Hive)

Privitar produces three different visualisations of output data. These visualisations are available from the **Job Details** section of a Job. Select **Manual Generalization** or **Automatic Generalization** to see the charts.

- Distortion Measures
- Cluster Size Histogram
- Cluster Size Bubble Chart
- Distortion Histogram

7.8.1. Distortion Measures

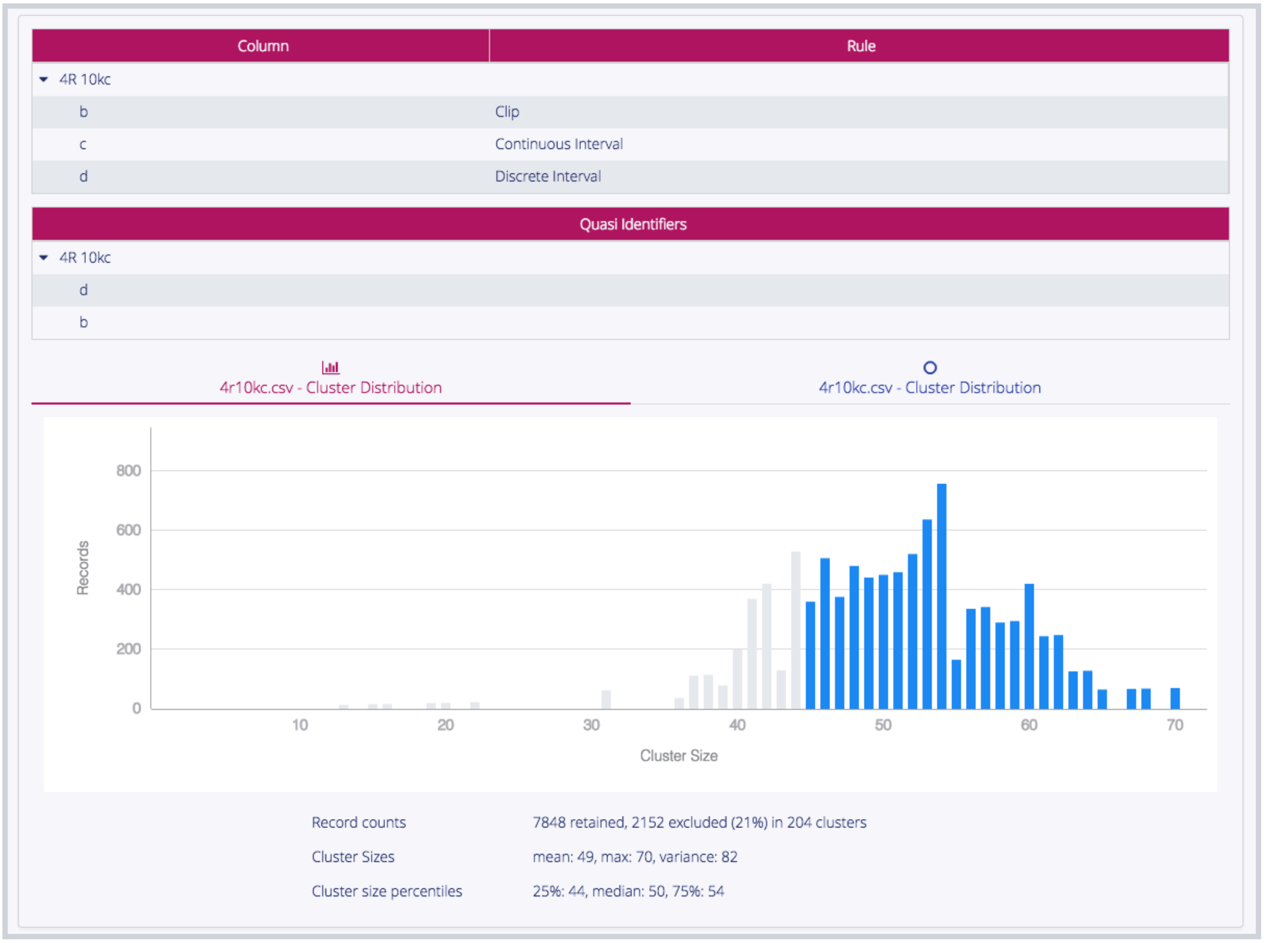


Column	Rule	Distortion
▾ Data		
Location	Categorical Generalisation	0.00%
RepaymentDate	Numeric Date Generalisation	5 years, 38 days
DOB	Numeric Date Generalisation	7 years, 320 days
Value	Continuous Interval Generalisation	258679.43

Privitar calculates a distortion measure for each quasi-identifier. This is the mean average error between the original data and the generalized output.

This value is intended to give a high-level summary of the trade-off between utility and privacy that has occurred as a result of the generalization process.

7.8.2. Cluster Size Histogram



Column	Rule
▾ 4R 10kc	
b	Clip
c	Continuous Interval
d	Discrete Interval

Quasi Identifiers
▾ 4R 10kc
d
b

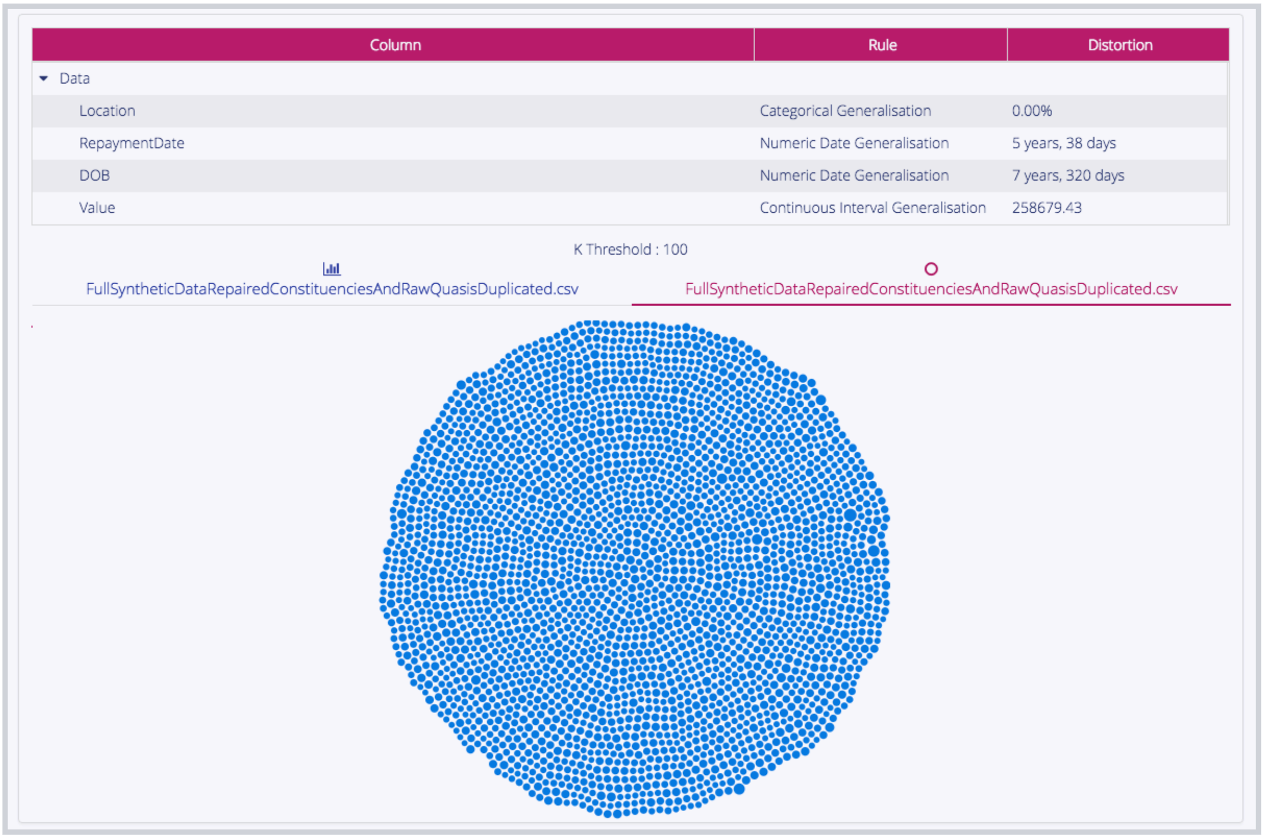
4r10kc.csv - Cluster Distribution
4r10kc.csv - Cluster Distribution
Records
800
600
400
200
0
10
20
30
40
50
60
70
Cluster Size

Record counts	7848 retained, 2152 excluded (21%) in 204 clusters
Cluster Sizes	mean: 49, max: 70, variance: 82
Cluster size percentiles	25%: 44, median: 50, 75%: 54

The Cluster Size Histogram visualises the distribution of cluster sizes in the output.

Clusters shown in grey do not meet the minimum cluster size threshold. For more information about cluster sizes, see What is k-anonymity?.

7.8.3. Cluster Size Bubble Chart



Column	Rule	Distortion
▾ Data		
Location	Categorical Generalisation	0.00%
RepaymentDate	Numeric Date Generalisation	5 years, 38 days
DOB	Numeric Date Generalisation	7 years, 320 days
Value	Continuous Interval Generalisation	258679.43

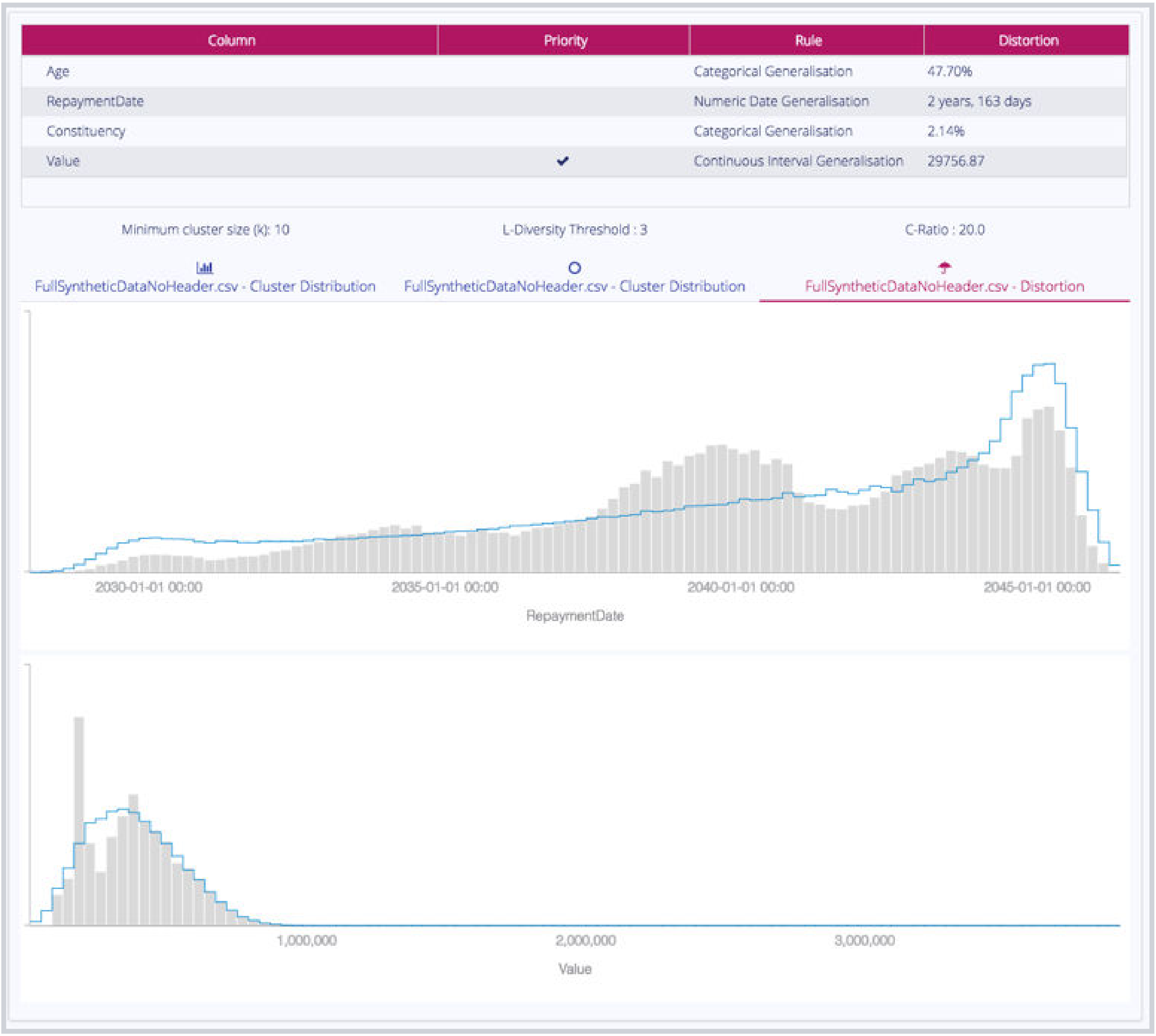
K Threshold : 100

FullSyntheticDataRepairedConstituenciesAndRawQuasisDuplicated.csv
FullSyntheticDataRepairedConstituenciesAndRawQuasisDuplicated.csv

The Cluster Size Bubble Chart visualises the relative sizes and counts of clusters in the output.

Clusters shown in grey do not meet the minimum cluster size threshold. For more information about cluster sizes, see What is k-anonymity?.

7.8.4. Distortion Histogram



Column	Priority	Rule	Distortion
Age		Categorical Generalisation	47.70%
RepaymentDate		Numeric Date Generalisation	2 years, 163 days
Constituency		Categorical Generalisation	2.14%
Value	✓	Continuous Interval Generalisation	29756.87

Minimum cluster size (k): 10 L-Diversity Threshold : 3 C-Ratio : 20.0

FullSyntheticDataNoHeader.csv - Cluster Distribution
FullSyntheticDataNoHeader.csv - Cluster Distribution
FullSyntheticDataNoHeader.csv - Distortion
2030-01-01 00:00
2035-01-01 00:00
2040-01-01 00:00
2045-01-01 00:00
RepaymentDate
1,000,000
2,000,000
3,000,000
Value

The Distortion Histogram compares the distributions of the original data and the generalized output data.

The original data is shown as the blue line, the generalized output is shown in grey.

7.9. Importing an Exported Job

Privitar can create Jobs by importing previously exported Job files.

A Job file contains the definition of the Job, and the Policy, Rules, and Schema it uses.

To import a Job:

1. Select **Jobs** from the Navigation sidebar.
2. Select **Import Job** from the **Actions** list box. The **Import Job** dialog box is displayed
3. Click on the **Browse...** link to select a JSON file to upload, or drag and drop a file directly on to the dialog box.

4. Review the name of the Job, Schema and Policy to be imported:
    - Click Import the included schema to import the Schema described in the file as a new Privitar Schema, or click Link with existing schema to re-use a Schema definition already in Privitar. The latter option is only available if at least one compatible Schema is stored in Privitar.
    - Click **Import** to proceed.
5. When importing the Policy used by the Job, Privitar will look in the Rules Library for the Rules included in the file:
    - If a compatible match is found (i.e. a Rule with the same name and definition), the existing rule will be re-used by the imported Policy.
    - If correspondingly named Rules exist, but their definitions are different from the definitions in the file, the conflicting rules will be listed in the Import dialog. The following actions are available to resolve the conflicts:
        - Select Resolve by creating new Rules to import the Rules from the file using new names.
        - Select Resolve by updating the Rules Library to overwrite the definition of the existing Rules in the Rules Library. Note that this operation will affect other Policies that might be using those Rules.
        - Select Resolve by using the current Rules from the Rules Library to use existing Rules and ignore the changes in the file.
6. Click **Import** to complete the import operation.

For more information on importing or exporting other Privitar objects, see Importing and Exporting Configurations.

7.10. Working with Existing Jobs

Details of Jobs and records of their runs remain in the system and can be accessed from the **Job Details** page. This page can be viewed by clicking on the name of the Job in the **Jobs** page.

The Jobs page lists all Job types that are enabled in your Privitar installation in their corresponding tabs: Batch Jobs, Data Flow Jobs and Privitar On Demand Jobs.

These frequently asked questions will help when working with existing Jobs.

7.10.1. Batch Jobs

The following sections cover some of the tasks associated with Batch Jobs.

All of the following sections assume you are viewing the **Job Details** page for a particular Job. To navigate to the **Job Details** page, follow these steps:

1. Select **Jobs** from the Navigation sidebar. The **Jobs** page is displayed.
2. Select the **Batch Jobs** tab.
3. Find the Job that you want to run. Use the **Filter** search box to filter the Jobs if required.

Finding the output data for a previously run Batch Job

When a Job is run successfully, its output data is written back to the location specified by the Protected Data Domain. To find the Protected Data Domain:

- Click on the name of the Job in the **Name** column.

  The **Job Details** page is displayed containing details of the Job together with a list of all the times that the job has been run.

  The PDD output for each Job run is shown in the **Protected Data Domain** column.

The output data can be found in the Protected Data Domain's details. This can be found from Protected Data in the Navigation sidebar. For more information on how to view the details of a PDD, see Viewing PDD details.

Re-running a Batch Job

You would typically re-run a Batch Job after new data becomes available or when a Policy is changed.

To re-run a Job, click on **Run** from the **Jobs** page.

Every new invocation of a Job is listed in the **Job Details** page.

Viewing more details about a Batch Job

To view more detail about the running of a Batch Job, in the **Job History** tab, click on **Show Details** in the **Details** column for the Batch Job you are interested in.

The **Job Results** dialog box is displayed showing information about the steps involved in executing a Job.

For each step, select the row to see a breakdown of the work done in that step.

STEP	Description
LOAD	The time taken to load an input file, the number of rows, and the total data size.
PARTITION (Optional)	The number (and size) of Spark partitions before and after the repartitioning of the Spark Dataframe.
MASKING	The rule that was applied to each column.
OUTPUT	The volume of data written after transformation and the location of the output.

Viewing diagnostic information about a failed Batch Job

Diagnostics provides access to the log file written while the Job was executing, and snapshots of the Policy and Environment that were used.

The snapshots are important because they provide a record of the Policy and Environment settings at the time of invocation. Privitar does not prevent a Policy or Environment from being modified after being used in a Job. When debugging, it may be useful to refer to these snapshots.

To view more detail about the running of a Batch Job, in the **Job History** tab, click on **Diagnostics** in the **Diagnostics** column for the Batch Job you are interested in. Click on **Download** to download the diagnostic file.

Finding a Batch Job's API ID

To invoke a Job from the Privitar Automation v3 APIs, its ID is required. This is a unique alphanumeric code associated with the Job.

The **Job Details** page displays the **Job ID** field containing the Job ID.

7.10.2. Data Flow Jobs

The following sections cover some of the tasks associated with Data Flow Jobs.

All of the following sections assume you are viewing the Job Details page for a particular Job. To navigate to the **Job Details** page, follow these steps:

1. Select **Jobs** from the Navigation sidebar. The **Jobs** page is displayed.
2. Select the **Data Flow Jobs** tab.

Finding the unique ID of a Data Flow Job

To set up a data flow plug-in Job, the Data Flow Job ID is required. This is a unique alphanumeric code associated with the Job.

The **Job Details** page displays the **Job ID** field containing the Job ID.

For more information about Data Flow jobs, see What are Data Flow Jobs?.

Downlading the input and output formats of a Data Flow Job

To configure a data flow plug-in, the input format and the output format of a Data Flow Job are required.

To download the Avro schema definition of a Schema:

1. Click on the row-select button to select a Data Flow Job.
2. Select **Export Input/Output Format** from the **Actions** menu. The **Export** dialog box is displayed.
3. Select **Input** or **Output** as the format to download.
4. Click on **Export** to download the file.

7.10.3. Privitar On Demand Jobs

The following sections cover some of the tasks associated with On Demand Jobs. For more information about Privitar On Demand Jobs, see About Privitar on Demand.

Finding the unique ID of a Privitar On Demand Job

To set up a Privitar on Demand Job that can be called through the SDK or Privitar on Demand HTTP API, the Job ID is required. This is a unique alphanumeric code associated with the Job.

To find this for a specific Privitar On Demand Job:

1. Select **Jobs** from the Navigation sidebar. The **Jobs** page is displayed.
2. Select the **Privitar On Demand Jobs** tab.
3. Select the name of the Job in the **Name** column.

   The **Job Details** page is displayed. The **Job ID** field contains the Job ID.

7.11. Partitioned Data and Wildcards (HDFS)

This section explains how to use Privitar to de-identify data that is supplied as a series of partitioned files in HDFS. For more information about filtering Hive data and partition columns, refer to Filtering Hive Data.

7.11.1. About Partitioned Data

Consider the following input folders, containing a partitioning of data across many files:

```
data
  2016-01
    20160101.avro
    20160101.avro
    ...
  2016-02
    20160201.avro
    20160201.avro
    ...
  ...
```

We wish to process all the Avro files into the same folder structure under the `result` folder:

```
result
  2016-01
    20160101.avro
    20160101.avro
    ...
  2016-02
    20160201.avro
    20160201.avro
    ...
  ...
```

Privitar supports this operation by being aware of folder structures underneath a root input folder (`/data` in this example), and creating a parallel structure under an output folder (`/result`).

The files representing the partitions are referred to using a relative path expression that is matched with the input folder as a base, and that may contain wildcards. In this example, the input folder is `/data`, and the required expression is `*/*.avro`.

There is no limit on the number of wildcard folder levels that can be used in relative paths.

Note

Each wildcard will match a single folder level. It is not possible to match multiple folder levels with a single wildcard. One wildcard must be used per folder level to identify the files, for example to match files one level deeper, use `*/*/*.avro`.

After processing, the files are created in corresponding positions by taking the part of the filename that matched the relative path and resolving it into the output folder.

Privitar exactly mirrors and replicates the input partition structure, as opposed to repartitioning based on the data. This is because the user might choose to transform, obfuscate or even remove the variables upon which the input data was originally partitioned.

7.11.2. Relative Path Examples

Relative paths take a wildcard expression that allows multiple files to be located in folders under the Input Root Folder.

Consider the same directory structure as above. All these examples assume a root input folder of `/data/`.

Example	Description
`2016-01/*.avro`	Will match all Avro files in `2016-01/`.
`*.avro`	Does not match any files, because there are no Avro files directly underneath the input root folder, and matching is not recursive through directories.
`*/*01.avro`	Wildcards can be placed in the filename part. Therefore, this example will match `/data/2016-01/20160101.avro` and `/data/2016-02/20160201.avro` (and so on) but not `/data/2016-01/20160102.avro`
`/2016-01/20160101.avro`	It is possible to omit wildcards completely. Therefore, this example refers to a single input partition file: `/2016-01/20160101.avro`.

To use wildcarded relative paths:

- When creating or Editing a Job, select the **Data Locations** tab (if present). Otherwise, the data locations will be displayed in the window underneath the **Input Root Folder** field.
- Enter a relative path with wildcards in the **Relative Path** field.

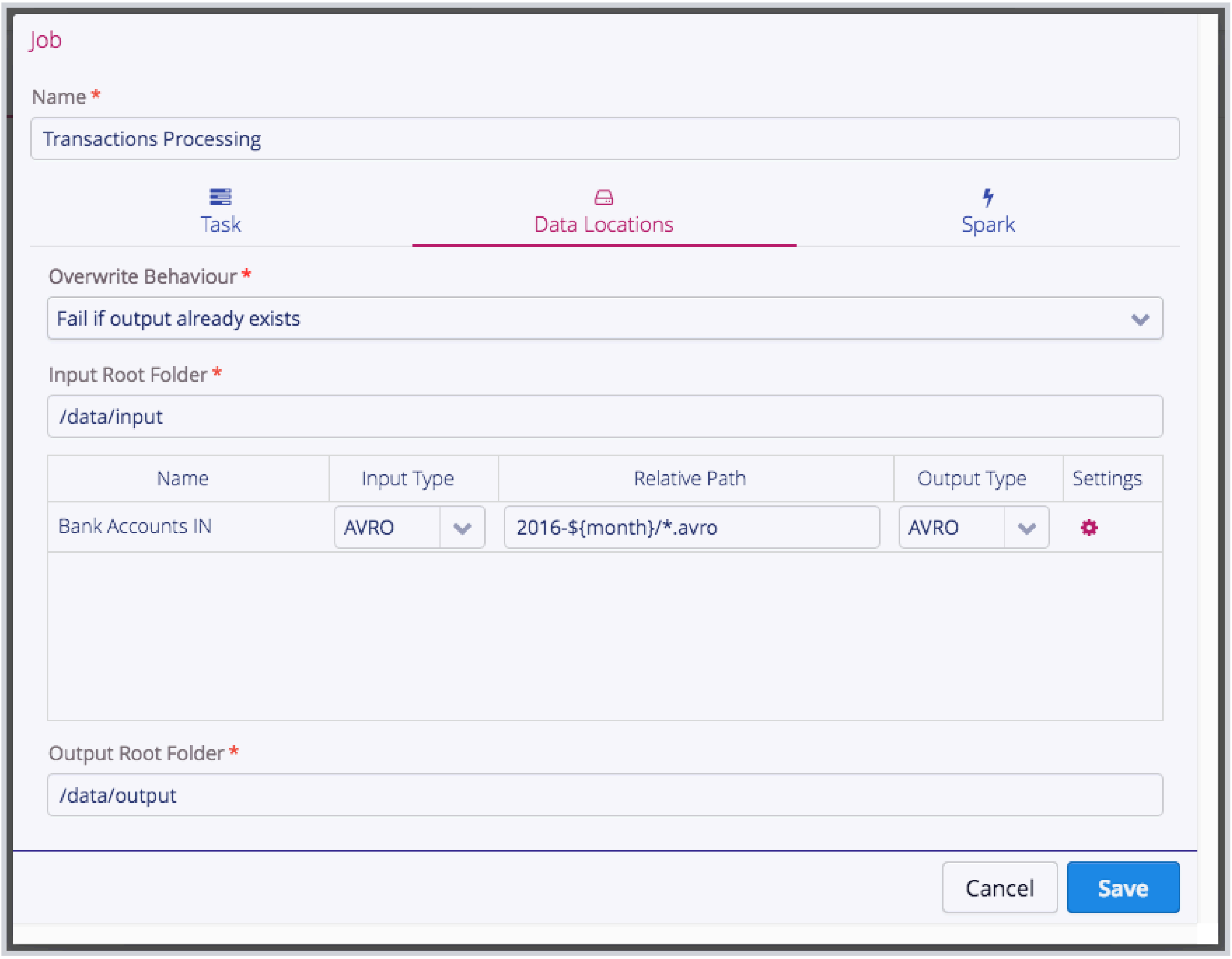
7.11.3. Path Replacements

In addition to the use of wildcards in relative paths it is also possible to use path replacements. (Use this format in particular when the Job is to be invoked using the Privitar REST API.) For example:

```
2016-${month}/*.avro
```

To specify a path replacement:

- When creating or Editing a Job, select the **Data Locations** tab (if present). Otherwise, the data locations will be displayed in the window underneath the **Input Root Folder** field.
- Enter a Relative Path with path replacements in the **Relative Path** column:



Name *
Transactions Processing

Overwrite Behaviour *
Fail if output already exists

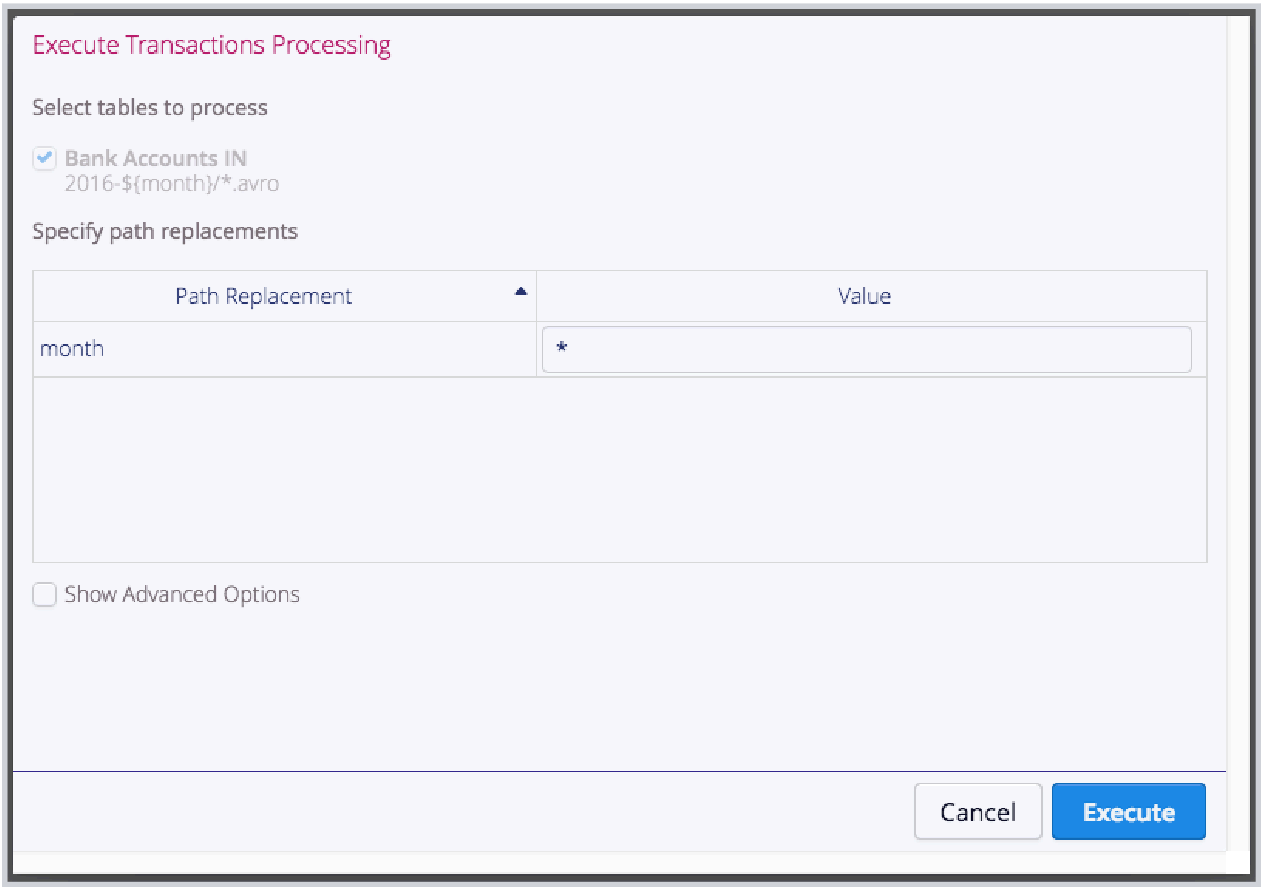
Input Root Folder *
/data/input

Name	Input Type	Relative Path	Output Type	Settings
Bank Accounts IN	AVRO	2016-${month}/*.avro	AVRO	⚙

Output Root Folder *
/data/output

In this example, the month replacement can be specified to invoke the Job to process specific matching files. Replacing `month` with `01` gives a resulting relative path of `2016-01/*.avro`.

When the Job is run, a message will be displayed in the **Run Job** window asking you to provide values for the variable parameters.


Execute Transactions Processing

Select tables to process

✓ Bank Accounts IN
2016-${month}/*.avro

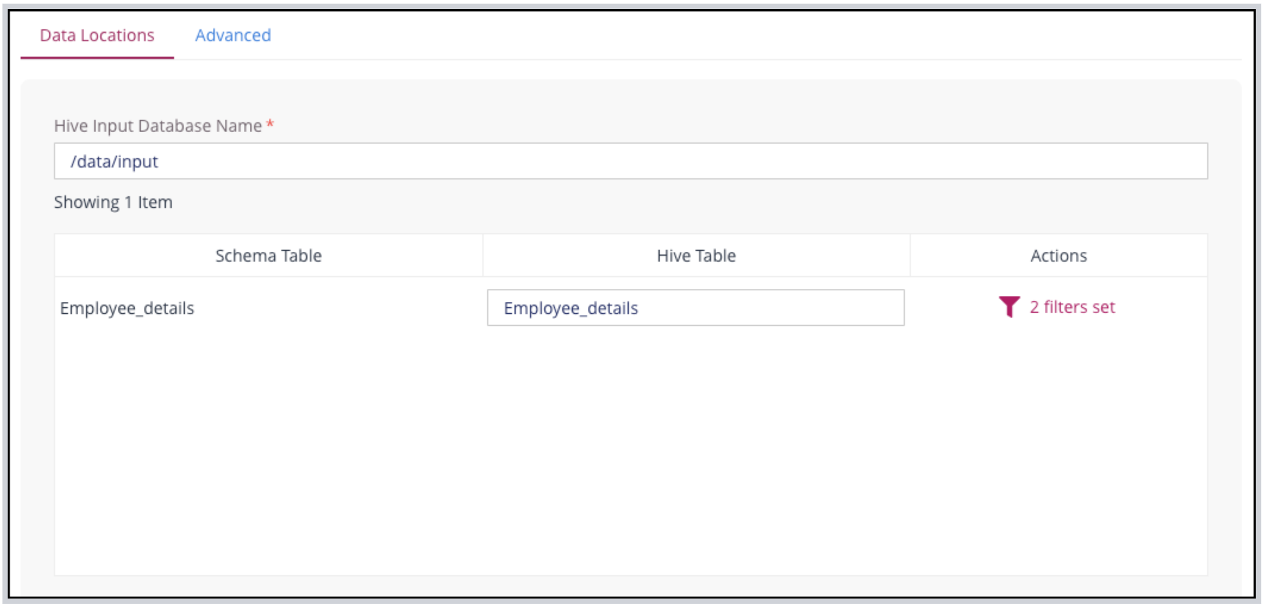
Specify path replacements

Path Replacement	Value
month	*

The Path Replacements can themselves contain wildcard symbols. When this is done, the entire replaced path is treated as if the wildcards were included directly. For example, replacing `month` in this example with `*` gives a relative path that is equivalent to `2016-*/*.avro`.

7.12. Filtering Hive Data

Privitar supports row-level filtering of Hive data prior to the data being processed in a Batch Job. For each column (both data and partition columns), a filter can be set to enable specific subsets of the data to be processed.



Hive Input Database Name *

/data/input

Showing 1 Item

Schema Table	Hive Table	Actions
Employee_details	Employee_details	2 filters set

The filters that can be applied are SQL-like query commands such as `Greater than` or `Less than`. For example, a filter could be applied to a column containing date information to only include rows of data from a specific date.

For a column containing the date format:

```
year-month-date
```

The following filter can be applied:

```
BETWEEN 2019-01-01 and 2019-06-30
```

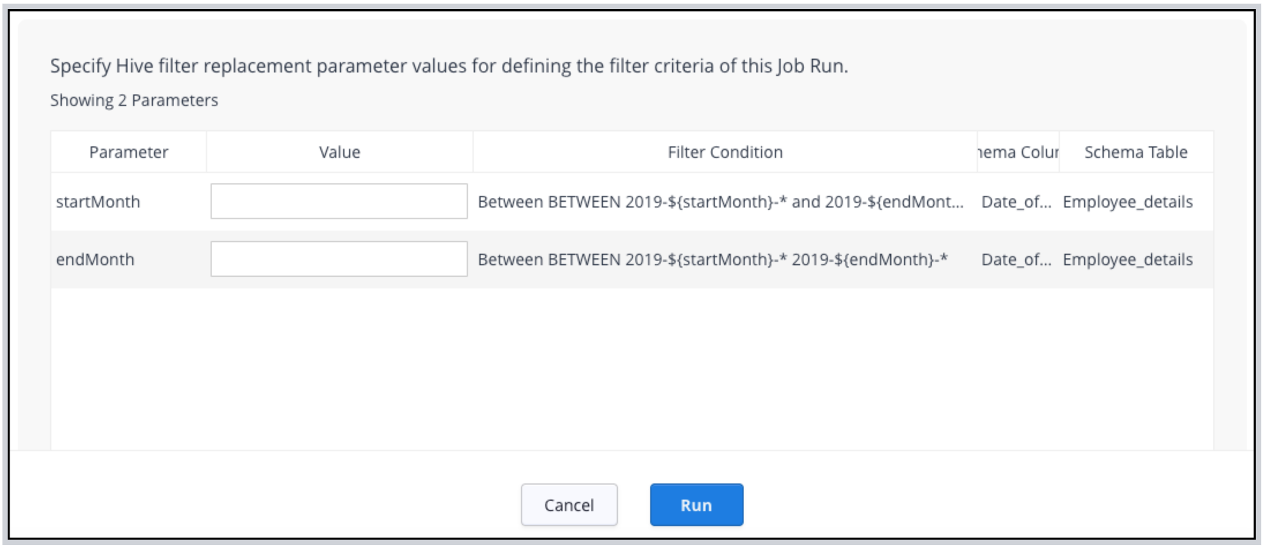
This means that the Batch Job will only include data rows that lie between the start of January 2019 and the end of June 2019 when the Batch Job is run.

A filter can also be defined using a variable parameter. The value of the variable will be requested when setting up the Job. For example, to specify the month as a variable for the column data in the previous example, the following filter can be defined.

```
BETWEEN 2019-${startMonth}-* and 2019-${endMonth}-*
```

When the Job is run, a message will be displayed in the **Run Job** window asking you to provide values for the variable parameters.

In this example, the month range needs to be processed. Any other filters which have been defined for this Job using variable parameters will also be shown:


Specify Hive filter replacement parameter values for defining the filter criteria of this Job Run.

Showing 2 Parameters

Parameter	Value	Filter Condition	ıema Colur	Schema Table
startMonth		Between BETWEEN 2019-${startMonth}-* and 2019-${endMont...	Date_of...	Employee_details
endMonth		Between BETWEEN 2019-${startMonth}-* 2019-${endMonth}-*	Date_of...	Employee_details

For more information about the filters that can be applied to row-level data, see Filter Types.

7.12.1. Filter Types

The following table lists the filter types that can be applied to filter row-level data in Hive Tables. The types of filter that are available will vary depending on the data type defined for each column. For more information about row-level filtering, see Filtering Hive Data.

Filter	Availability	Description
Equals	All types	`A = B` TRUE if expression A is equal to expression B, otherwise FALSE. (The Hive Job will process rows which equal the given value in the column this filter is applied to.)
Not Equal To	All types	`A != B` TRUE if expression A is NOT equal to expression B, otherwise FALSE. (The Hive Job will process rows which do NOT equal the given value in the column this filter is applied to.)
Greater Than	All types	`A > B` TRUE if expression A is greater than expression B, otherwise FALSE. (The Hive Job will process rows which have a greater value in the column this filter is applied to than the given value.)
Greater Than or Equal To	All types	`A >= B` TRUE if expression A is greater than or equal to expression B, otherwise FALSE. (The Hive Job will process rows which equal or have a greater value than the given value in the column this filter is applied to.)

Filter	Availability	Description
Less Than	All types	`A < B` TRUE if expression A is less than expression B, otherwise FALSE. (The Hive Job will process rows which have a value in the column this filter is applied to that is less than the given value.)
Less Than or Equal To	All types	`A <= B` TRUE if expression A is greater than or equal to expression B, otherwise FALSE. (The Hive Job will process rows which equal or have a value that is less than the given value in the column this filter is applied to.)
Like	Text	`A LIKE B` TRUE if string A matches the SQL simple regular expression B, otherwise FALSE. The comparison is done character by character. (The Hive Job will process rows that matches the column value. The given value to filter on must be in the SQL simple regular expression form.)
Regular Expression is Like	Text	`A RLIKE B` TRUE if any (possibly empty) substring of A matches the Java regular expression B, otherwise FALSE. (The Hive Job will process rows that matches any substring of the column value. The given value to filter on must be in the Java regular expression form.)
Between	All types	`A BETWEEN B AND C` TRUE if A is greater than or equal to B AND A less than or equal to C, otherwise FALSE. (The Hive job will process rows which have a greater column value than the first given value and lower column value than the second given value.)

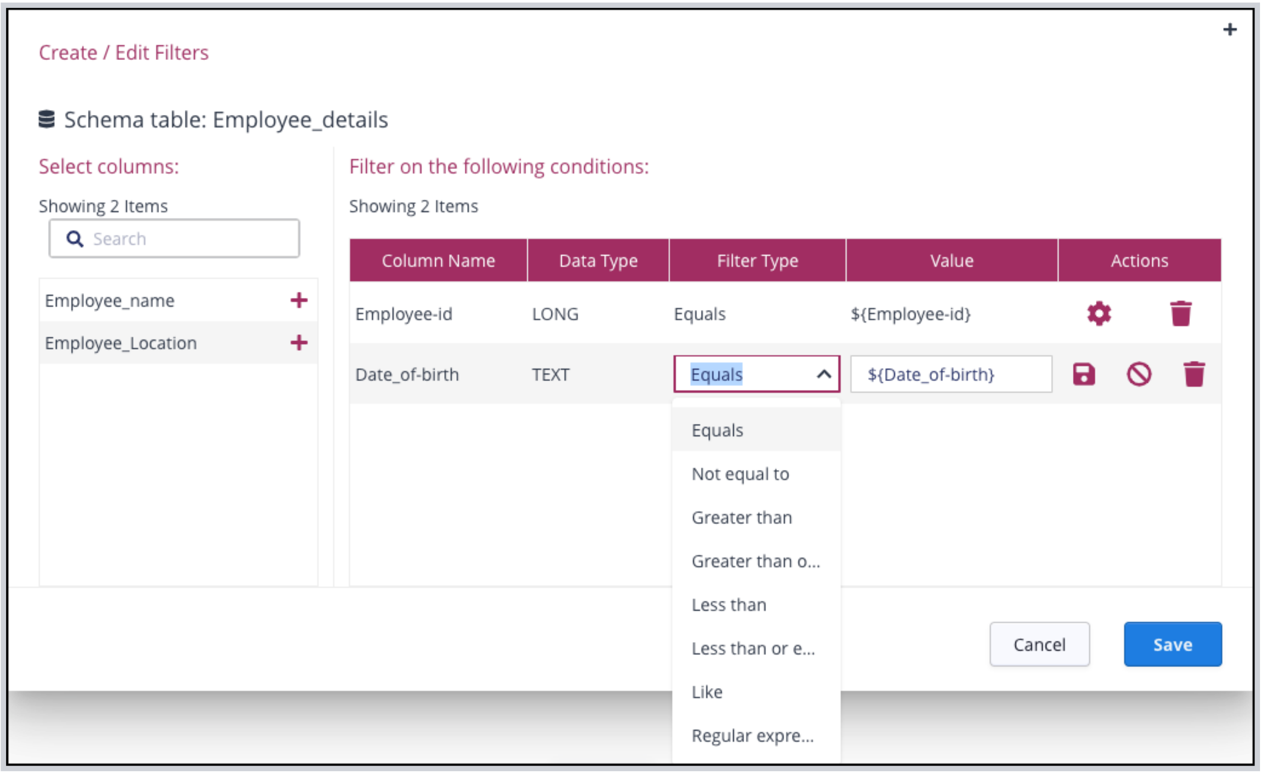
7.12.2. Setting Filters for a Hive Batch Job

To set or change Filters for a Hive Batch job:

1. Select **Jobs** from the Navigation sidebar. The Jobs page is displayed.
2. Select the Name of the Batch Job in the **Name** column. The **View Job** window is displayed providing details of the Job.
3. Select the *Edit* icon. The **Edit Batch Job** window is displayed.
4. Select the **Data Locations** tab.
5. Select the *Filter* icon in the **Actions** column. The **Create/Edit Filters** dialog box is displayed. The dialog box lists all the columns that are included in the selected Hive table.
6. Select the **+** symbol next to the column that you want to apply a filter to. (Use the **Search** box to search for a specific column.) The selected column moves into the **Filter on the following conditions** table. For each column selected, the table displays:
   - The name of the column, under **Column Name**.

- The type of data in the column, under **Data Type**.
- The filter to apply, under **Filter Type**. (The **Equals** filter is initially selected as a default setting, but you can change this to other filter types.)
- The value for the filter under **Value**. (A parameterized value is created by default (based on the **Column Name**) but you can change this to other parameter formats or even hard-code it to a specific value.)

7. Select the *Cog* icon in the **Actions** column. This enables the **Filter Type** column and **Value** column to be edited, so that a filter can be set for the selected data column.


Create / Edit Filters

Schema table: Employee_details

Select columns:

Showing 2 Items

Search

Employee_name

Employee_Location

Filter on the following conditions:

Showing 2 Items

Column Name	Data Type	Filter Type	Value	Actions
Employee-id	LONG	Equals	${Employee-id}	
Date_of-birth	TEXT	Equals	${Date_of-birth}	

8. Select the filter to apply to the column from the list box in the **Filter Type** column. (For more information about the filters that are available, see Filter Types.)
9. Enter any associated parameter values for the filter in the **Value** column.
10. Select the *Disc* icon in the **Actions** column to save the filter.
11. Create filters for any other other columns that you want to apply a filter to in the selected Hive table.
12. Select **Save** to save the filters. The **Edit Batch Job** window is displayed. The number of filters that have been set are shown in the **Actions** column.

For more information about row level filtering of Hive data, see Filtering Hive Data .

7.13. What are Data Flow Jobs?

Similarly to Batch Jobs, Data Flow Jobs apply a Policy on concrete input data. Instead of producing output files on HDFS, they produce a flow of output data. Inputs and output flows are managed on the external platforms executing the Job. Both input and output data can consist of a stream of records or a series of batches of data.

Input and output data flows are connected to the Privitar Data Flow plug-ins directly in the executing platform. The output data is published to the Protected Data Domain selected in the Data Flow Job configuration.

Pipelines using Data Flow Jobs with the same destination PDD will produce consistent data (if the **Preserve data consistency** option is used).

Once created, a Data Flow Job is referenced from the external pipeline by its ID.

Configuration such as bad record handling is delegated to the Data Flow execution platform.

7.13.1. Creating a Data Flow Job

A Data Flow Job contains a reference to a Policy and a Protected Data Domain. It enables a Policy to be applied to a flow of data, publishing an output flow.

Before creating a Data Flow Job you need to ensure that the Policy that will be used by the Job already exists and is not in Draft.

To create a Data Flow Job, follow these steps:

1. Select **Jobs** from the Navigation sidebar. The Jobs page is displayed.
2. Select the **Data Flow Jobs** tab.
3. Choose the Environment from the **Environment** list box.

   The chosen Environment must contain the data to be processed by the Policy, and will also be the destination of the processed data.
4. Click on **Create New Data Flow Job**. The **New Data Flow Job** dialog box is displayed.
5. Enter a name for the Job in the **Name** box.

   This name is used to identify the Job in the Privitar user interface and via Automation APIs.
6. Select a compatible Policy from the **Policy** list box.

   A compatible Policy must have the following features:

   - Use compatible masking rules. (For more information, see Masking Rules.)
   - Use Generalization strategies that are supported for this type of Job. (That is, not enforcing k-anonymity).
   - Use a Schema with one table.

   The **Allow local token caching check box** is enabled by default.

   Note

   When using consistent tokenization, a connection is made to the configured Token Vault to read/write tokens. To improve performance, a local in-memory cache is used. Caching can be disabled by un-checking this box, but this will have a detrimental effect on the processing performance.
7. Click on **Next**. The dialog box is updated with details about the PDD to be included in the Job.

8. Select an existing Protected Data Domain from the list or select **Add to a new Protected Data Domain** to create a new PDD.

   For more information about creating PDDs, see Creating a PDD.
9. Click on **Next** to create the Job.

   The Job appears in the index of Jobs on the **Jobs** page and the **Jobs Details** page for the Job contains the Job ID that can be used to reference the Job.

7.13.2. Configuring a Data Flow Pipeline

A Data Flow Job execution is not initiated from Privitar, but it is entirely managed externally, in the Data Flow platform of choice.

Privitar provides platform-specific plug-ins to execute Data Flow Jobs. The supported platforms are:

- Apache NiFi
- Apache Kafka Connect/Confluent Platform
- StreamSets

For other platforms, Privitar provides an SDK that can be used to build a custom integration.

To configure a Data Flow platform, the following information is required:

- The unique ID of the desired Data Flow Jobs. This can be found in the Jobs list. For more information, see Working with Existing Jobs.
- The Privitar URL. This can be provided by your administrator
- The credentials of a Data Flow Job Operator.

  The operator has to be set up as an API user that has a Role with a Run Data Flow permission in the Team that the Job is defined in. (For example, the default Data Flow Operator Role).

  For more information, see Managing API Users.
- The details of input and output data formats. These can be obtained from the Jobs list in Avro Schema format. These files can be imported directly in an Avro Schema Registry, or used as a reference for configuring the Data Flow platform.

  For more information, see Working with Existing Jobs.

Data Flow Execution

On startup, the Data Flow plugin will connect to the configured Privitar instance to download the Data Flow Job configuration.

While the pipeline runs, the data flow plugin will continuously process the incoming data, and apply the configured Policy.

The output data will be sent either to the next step in the pipeline or to the final destination of the data, depending on the specific configuration of the data flow pipeline.

Note

On NiFi and Apache Kafka/Confluent, any changes to Policies and Jobs will be dynamically applied after the refresh time interval (set by default to 10 minutes). On StreamSets, any changes to the Policy and Jobs or the Data Flow Job configuration while a pipeline runs, will not take effect until the pipeline process is restarted.

Data Flow supports the following authentication mechanisms between the Data Flow plugin and Privitar:

- Apache NiFi: Mutual TLS or Basic HTTP authentication.
- Apache Kafka Connect/Confluent Platform: Mutual TLS or Basic HTTP authentication.

  Note

  For more information on how to configure the Apache Kafka Connect/ Confluent Privitar Connector, see the separately provided *Kafka Connect Reference Guide*. (Please contact Privitar for further information about Apache Kafka Connect/Confluent integration.)

- StreamSets: Mutual TLS or Basic HTTP authentication.

  Note

  For more information on how to configure the StreamSets Privitar Data Processor, see the separately provided *StreamSets Reference Guide*. (Please contact Privitar for further information about StreamSets integration.)

Failed Records

The Data Flow plugin can be optionally configured with a failed records output, where all the input data records which Privitar failed to tokenize are output. The details of how to configure the failed records flow are specific to the data flow platform in use.

On NiFi, it is recommended that before the Privitar processor a ValidateRecord processor is placed to ensure the incoming data is valid (for example, using the correct date formats).

8. Unmasking De-identified Data

Under certain strict conditions it is possible to recover the original values for tokens contained in a Protected Data Domain (PDD). This is known as **Unmasking**. Privitar provides this unmasking capability to support:

- Storing de-identified data in remote systems while locally retaining Token Vaults.
- Processing records produced as part of an external analysis that contain masked values but conform to a different Schema.

8.1. Rules compatible with Unmasking

The de-identified output of certain Privitar Rules may be unmasked, given the appropriate Roles/permissions and Policy configuration. Output of several Rule types may be unmasked.

For a list of Rule types that are compatible with Unmasking, see Masking Rule Types Supporting Unmasking:

8.2. Restrictions on Unmasking

Unmasking the output from Rule types that support unmasking requires the following conditions to be met:

- The Privitar User has the correct Role and permissions for the unmasking being performed.
- It can be established which PDD and Rule produced the token.
- The token was produced by a Rule with **Preserve data consistency** and **Permit unmasking of original values** enabled.

And for all but the Encrypt and SecureLink Encryption Rules:

- A populated Token Vault. That is, the Job has been run at least once to populate the Token Vault, and the Token Vault has not since been deleted.

8.3. Unmasking Features

Privitar can unmask tokens in two ways:

- Unmasking a single token
- Unmasking a file of tokens

8.4. Unmasking a Single Token

To unmask a single token value in a PDD:

1. Ensure the you have the Investigator role, or an equivalent role that provides the permission:
    - **Unmask Token** for the **Protected Data** object. (For more information, see Managing Roles.)

2. Select **Protected Data** from the Navigation sidebar.

   The **Protected Data** page is displayed, containing a list of the PDDs that have been created.
3. Select the PDD by clicking on the name of the PDD in the PDD index listing.

   The **Protected Data Domain Details** window is displayed for the selected PDD.
4. Select **Unmask value**.

   The **Unmask** dialog box is displayed.
5. Select the rule that generated the token that you want to unmask, from the **Rule** list box.

   If the Rule is not listed, check that the rule you have selected is one of the Rule types that supports unmasking. (See, Unmasking De-identified Data). The Rule also needs to have been applied to the data with the options **Preserve data consistency** and **Permit unmasking of original values** enabled.
6. Enter the tokenized value that you wish to unmask in the **Token** edit box.
7. Select **Unmask** to display the original value.

   For security reasons, enter a second set of Privitar credentials. This is required for the Investigator role before the unmasking can be performed.
8. If the token has an original value, it is displayed.

8.5. About Unmasking jobs

Privitar can unmask data in the same execution engines used for masking:

- Columnar CSV, Avro or Parquet data files in HDFS.
- Streams of records in Apache NiFi.
- Streams of messages in Apache Kafka.
- Streams of messages in Streamsets.
- Records passed via HTTP REST API to Privitar On Demand.
- Embedded in Java-based applications with the Privitar SDK.

In all cases, the general process of unmasking data is as follows:

1. Either via the Privitar user interface or via the Automation API, create an Unmasking Job that describes the nature and structure of the data that contains masked tokens. Individual Unmasking Jobs are bound to the PDD that contains the masked data to unmask.
2. Include the ID of the Unmasking Job in the corresponding execution engine configuration. Each Unmasking Job has a unique identifier for this purpose.
3. When the Unmasking Job is applied to the data, a copy of the input is produced with the appropriate values converted from their de-identified form back to their unmasked values.

8.5.1. Unmasking Job Permissions

To ensure that Unmasking Jobs are only used in an authorised way, Privitar has a set of Unmasking Job Permissions that must be present on a user before an Unmasking Job will operate.

In particular, Permissions are provided for Create, Edit, Run Batch and Run Data Flow. Only users with the correct Permissions in their assigned Roles are allowed to manipulate and execute Unmasking Jobs.

For more information about Permissions for Unmasking Jobs, see Role Permissions.

8.5.2. Working with Unmasking Jobs

Unmasking Jobs that process data files on HDFS are referred to as Batch Unmasking Jobs. For more information, see Creating Batch Unmasking Jobs.

Unmasking Jobs that process data in streaming systems such as Apache NiFi, Apache Kafka and Streamsets are referred to as Data Flow Unmasking Jobs. Unmasking Jobs for Privitar on Demand are similar. For more information, see Creating and Running Data Flow/POD Unmasking Jobs.

8.5.3. Creating Batch Unmasking Jobs

To create a Batch Unmasking Job:

1. Select **Protected Data** from the Navigation sidebar.

   The **Protected Data** page is displayed, containing a list of the PDDs that have been created.
2. Select the PDD by clicking on the name of the PDD in the PDD index listing.

   The **Protected Data Domain Details** window is displayed for the selected PDD.
3. Select the **Unmasking Jobs** tab.
4. Select **Batch** from the **Create Unmasking Job** list box. The **Create Unmasking Batch** Job page is displayed.
5. Complete the **Job** tab. For more information, see Defining Unmasking Jobs.
6. Complete the **Data Locations** tab to reference the HDFS data file that is to be processed and to specify where the unmasked data should be written:
   - The entry in the **Input root location** box determines the base location in HDFS containing the data file to be unmasked.
   - The entry in the **Relative path to file** box is interpreted relative to the **Input root location** to identify the file. This may contain wildcards and path replacements. (For more information, see Partitioned Data and Wildcards (HDFS).)
   - The **Output root location** is used as the base location for the unmasked output. The data file itself is written to this location by adding the **Relative path to file**.
7. If required, select the **Advanced** tab if you wish to customize the Spark execution of the Unmasking Job.

   For more information, see the Apache Spark documentation.

8.5.4. Running Batch Unmasking Jobs

Batch Unmasking jobs can be run from the Privitar User-interface and from the Automation API.

From the User-interface

To run a Batch Unmasking Job via the Privitar user interface:

1. Select **Protected Data** from the Navigation sidebar.

   The **Protected Data** page is displayed, containing a list of the PDDs that have been created.
2. Select the PDD by clicking on the name of the PDD in the PDD index listing.

   The **Protected Data Domain Details** window is displayed for the selected PDD.
3. Select the **Unmasking Jobs** tab.
4. Select the Unmasking job from the list of Jobs shown on the page.
5. Select **Run** from the **Actions** list box.

   The Job will run and status updates for the Unmasking job will be displayed in the **Protected Data Domain** page.

From the Automation Rest API

An Unmasking Job can also be run from the Privitar Automation API.

For more information about the Automation API, see the Automation API Explorer in the APIs section of the *Privitar User Guide* in the Privitar UI application.

Some things to be aware of:

- Note the Job ID displayed on the Unmasking Job. This is required to refer to the Unmasking Job from the REST API.
- The API User making the API call must have appropriate Unmasking Job Permissions. (For more information, see Role Permissions.)
- The status of running Unmasking Jobs are displayed on the Unmasking Jobs list, and can be monitored via the Unmasking Batch Job Run endpoint:

  ```
  GET endpoint (/unmasking-jobs/batch/{id}/runs/{run-id})
  ```

8.5.5. Creating and Running Data Flow/POD Unmasking Jobs

Creating a Data Flow Unmasking Job

To create a Data Flow Unmasking Job for use with Apache NiFi, Apache Kafka, Streamsets or the Privitar SDK:

1. Select **Protected Data** from the Navigation sidebar.

   The **Protected Data** page is displayed, containing a list of the PDDs that have been created.
2. Select the PDD by clicking on the name of the PDD in the PDD index listing.

   The **Protected Data Domain Details** window is displayed for the selected PDD.
3. Select the **Unmasking Jobs** tab.
4. Select **Data Flow** from the **Create Unmasking Job** list box. The **Create Unmasking Data Flow Job** page is displayed.

5. Complete the **Job** tab. For more information, see Defining Unmasking Jobs.
6. If required, select the **Advanced** tab if you wish to enable or disable Token caching.

Creating POD Unmasking Jobs

To create a Privitar On Demand Unmasking Job:

1. Select **Protected Data** from the Navigation sidebar.

   The **Protected Data** page is displayed, containing a list of the PDDs that have been created.
2. Select the PDD by clicking on the name of the PDD in the PDD index listing.

   The **Protected Data Domain Details** window is displayed for the selected PDD.
3. Select the **Unmasking Jobs** tab.
4. Select **Privitar On Demand** from the **Create Unmasking Job** list box. The **Create Unmasking POD** Job page is displayed.
5. Complete the **Job** tab. For more information, see Defining Unmasking Jobs.

Running Data Flow and POD Unmasking Jobs

To use a Data Flow or Privitar On Demand Unmasking Job, refer to separate Privitar documentation about deploying to Apache NiFi, Apache Kafka, Streamsets or Privitar On Demand.

Some things to be aware of:

- Take note of the **Job ID** displayed on the Unmasking Job. This is required input to the Apache NiFi, Apache Kafka, Streamsets or Privitar On Demand configuration.
- The user that executes the Unmasking Job must have appropriate Unmasking Job Permissions. For more information, see Role Permissions.

8.5.6. Defining Unmasking Jobs

When creating an Unmasking Job, it is necessary in all cases to describe which Rules were originally used to produce the masked input.

- If the records to be unmasked originated from a Policy, Privitar can infer the Unmasking Job settings from that Policy automatically.
- If the records conform to an **existing** Privitar Schema, the Unmasking Job settings must be specified explicitly. This may be the case when the data format has been modified since being generated.

To define an Unmasking job entails completing the Job tab for an Unmasking Job.

1. Enter a name for the Unmasking Job in the **Name** box. This name is used to identify the Unmasking Job in the Privitar interface.
2. Select **Choose** to define the Schema of the data to be unmasked.

   The **Select Schema** dialog box is displayed. From here, you can select the data to be unmasked from either a Policy or based on a Schema.
3. Click on the button alongside the source of the data to be unmasked. Use the **Filter** box to filter the list of available sources:

If the Schema is based on a Policy:

a. Click on **Records produced by a Policy**.
b. Select the Policy from the list of policies displayed.
c. Click on **OK**.

   The Rules used in the Unmasking Job are filled-in on the **Job** tab automatically.

If the Schema is based on an existing Schema:

a. Click on **Records based on an existing Schema**.
b. Select the Schema from the list of Schemas displayed.
c. In the **Job** tab, use the **Assign Rule** and **Assign rule To Selection** actions to specify for each field, either:
   - the Rule that produced the value for the field, or
   - the **Retain Unchanged** rule.

4. The **Job** tab is complete. Continue to the remaining tabs to complete the full definition of the Job.

9. Watermarking a Dataset

The watermarking capability of the Privitar Platform can be used to embed a unique stamp into the content of files written to a Protected Data Domain (PDD). This stamp can be used to trace the origin of a file to the PDD for which it was originally produced. The presence of a watermark is a good incentive to a file's recipient to make sure they are not careless or malicious with the data, and also gives traceability in the event of a data breach or if data turns up somewhere unexpected.

The stamp is contained within the tokenized values used for de-identification. Given an arbitrary file, Privitar can investigate the content to identify which PDD, if any, a file belongs to. Locating the PDD gives access to its metadata properties, such as the file's intended purpose, who authorized the release of the data, and the Policy it was released under.

9.1. Conditions on Watermarking

To embed a watermark, the following conditions must be met:

- The policy must include at least one rule from the following list: Regular Expression Number Generator rule, Regular Expression Text Generator rule, or Hash Text rule.
- The PDD must have been created with the Embed Watermarks option selected. This option is provided in the PDD creation dialogue.
- Because of the way the watermark is embedded, the file must have a large enough number of rows to contain the watermark. As a rule, files larger than 15,000 rows, with enough unique values, can contain watermarks.
- Privitar NOVLT does not currently support watermarking.

Note

If you select "Tokenize without a vault (use NOLVT)," the platform will not embed a watermark.

Warning

When running data flow or POD jobs, the PDD is still editable after you have processed data into that PDD. If watermarking was previously disabled, and you enable it after you processed data, this could make it harder to investigate the watermark when some of the data doesn’t have a watermark.

It is possible to embed watermarks into the output data when using data flow jobs, batch jobs, and POD jobs.

If the above conditions are met, each column mapped to a Regular Expression Text Generator rule, a Regular Expression Number Generator rule, or a Hash Text rule will contain a watermark when you run a job for that policy in the PDD.

9.2. Embedding a Watermark

Before producing a file with an embedded watermark, note that there are conditions that must be met. For more information, see Watermarking a Dataset.

Once these conditions are met, the watermark will be embedded in columns de-identified using a Regular Expression Text Generator rule, a Regular Expression Number Generator rule or a Hash Text rule.

To produce a file with an embedded watermark:

1. Create a PDD either from the **Protected Data Domain** page, while running a Batch Job, creating a Data Flow Job or a POD job.
2. Ensure the **Embed Watermarks** checkbox is selected when the PDD is created.
3. The metadata properties specified when the PDD is created will become associated with the watermark. These values will be displayed when a file from this PDD is investigated.
4. Run a Job in the PDD, either by proceeding with the Run Job process, using an existing Job or by configuring a Data Flow or POD pipeline.

During this Job run, any output file produced using a Policy with a Regular Expression Text Generator rule, a Regular Expression Number Generator rule or a Hash Text rule will contain a watermark.

9.3. Investigating Watermarks

The Investigation page in the Privitar Platform can be used to investigate a watermark. This allows sample data in HDFS to be analyzed for the presence of a watermark.

The investigation process may take some time, depending on the data size, and so Privitar allows multiple ongoing investigations to be tracked on the Policy Manager interface.

Once the process is complete, the record of the investigation remains present with a link to the PDD, if one was identified.

The watermark investigations require an Environment with either a configured Hadoop Cluster and HDFS to contain the sample or an AWS Glue environment with the sample on S3. Before beginning an investigation, note the location on HDFS of the data sample.

To investigate a file to see if it contains a watermark:

1. Select **Investigation** from the Navigation sidebar.

   The **Investigation** page is displayed displaying a list of previous investigations that have been performed.
2. Click on **Investigate File** to begin an investigation.

   The **Investigate File** dialog box is displayed.
3. Select the Environment that contains the file to investigate, from the **Environment** list box.
4. Specify the full pathname of the file (including the name of the file), in the **File Path** edit box.

If necessary, specify the type of file from the list box. (Click on the *Settings* icon to specify options for reading and writing CSV files.)

5. Click on **Start** to begin the investigation process.

   The pending investigation is shown as a new entry on the **Investigations** page. The outcome of the process can be:

   - A PDD was identified by a watermark match.
   - No PDD was identified.
   - An error occurred during the investigation process. For example, an incorrect pathname was specified.

   Refer to the following sections for more information.

9.3.1. A PDD was Identified

When a PDD is identified, its name is displayed as a **Match** in the entry for the investigation on the **Investigations** page. The PDD that has been identified is displayed, along with a link to the PDD's detail page.

9.3.2. No PDD was Identified

There are several reasons why a PDD might not be identified:

- The file did not contain a watermark.
- The file has been tampered-with to the extent that it was not possible to confidently extract the watermark.

In both these cases, the Investigation is marked as **No Match** on the **Investigations** page.

9.3.3. An error occurred during the Investigation

If an error occurs during extraction, then a **Failed** status is displayed in the entry for the investigation on the **Investigations** page.

A **View Log** link to the execution log is provided for troubleshooting purposes in the "..." menu.

10. Privacy Support

This section contains information about the features that Privitar provides in support of specific data privacy requirements mandated by the following Data Privacy laws and regulations:

- GDPR (General Data Protection Regulation)
- HIPAA (Health Insurance Portability and Accountability Act)

The features are categorized according to the law or regulation that they provide support for:

- GDPR
  - **Remove Token Vault mapping**. This is an API feature that can be used to support **Right to be Forgotten** (RtbF) requests. RtbF is part of GDPR.
- HIPAA
  - **Generalize Date**. This is a Privitar Rule that can be used to satisfy the HIPAA (Health Insurance Portability and Accountability Act) requirement for the storage of ages and dates contained in patient healthcare data.

10.1. Removing Token Vault Mapping

This section describes how to use the Remove Token Vault Mapping feature in the Automation API. This feature enables the mapping between an original value and a tokenized value to be removed from the Token Vault. This feature provides support for meeting **Right to be Forgotten (RtbF)** requests under the **General Data Protection Regulation (GDPR)**. For more information, see What is the Right to be Forgotten?.

10.1.1. Requirements

The following requirements must be met in order for you to be able to use this feature:

- You must be using either an OracleDB, Postgres, HBase or DynamoDB for your Token Vault. (See, Token Vault Types.)
- You must have the **Remove Token Mapping** permission to execute the request. (See, Managing Roles.)
- You must understand the impact of using this feature. For more information, see Important Considerations.

10.1.2. Execution

To remove a Token Vault mapping, a POST request needs to be submitted to the following PDD API endpoint:

```
/policy-manager/api/v3/pdds/{pddId}/remove-token-mapping-requests
```

The endpoint expects the following JSON body:

```
POST
/policy-manager/api/v3/pdds/abc123/remove-token-mapping-requests
```

```
HTTP/1.1
Content-Type: application/json

{
  "identifier": {
    "value": "12345",
    "dataType": "TEXT"
  },
  "rule": {
    "id": "1234ab"
  }
}
```

10.1.3. Status Endpoint

The status of the request can be retrieved by submitting a `GET` request to:

```
/policy-manager/api/v3/pdds/{pddId}/remove-token-mapping-requests/{requestId}
```

The possible statuses are `SUCCESS`, `FAILURE` and `RUNNING`. The endpoint will return `200 OK` regardless of the actual status. For example:

```
HTTP/1.1 200 OK
Content-Type application/json

{
  "status": "FAILURE",
  "failureReason": "Failed to connect to POD",
  … other JSON fields
}
```

10.1.4. Result

After a token mapping has been removed, attempts to Unmask and re-identify the value will fail.

The value will be assigned a new token should it be encountered again by the same rule in the same PDD, and the previous token will not be reused. For more information, see Important Considerations.

10.1.5. What is the Right to be Forgotten?

The **Right to be Forgotten (RtBF)**, also called the *right to erasure*, is one of the data subject rights enshrined in the General Data Protection Regulation (GDPR). Article 17 of the GDPR empowers data subjects to request the deletion of data relating to them, in certain circumstances.

For more detail on the regulatory approach to the RtbF and the circumstances in which data controllers are required to delete the data, please see the Privitar whitepaper, GDPR: The Right To Be Forgotten.

There are two common challenges for data controllers responding to deletion requests:

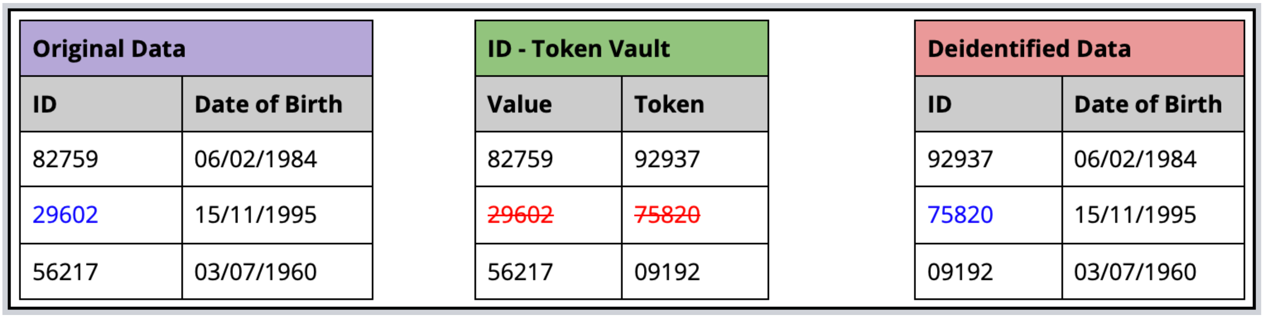
- GDPR's accountability principle requires that the data controller is able to demonstrate compliance. This means that the *deletion* should be auditable.
- Managing the knock-on effect of deleting a customer's record in a *live* system on downstream systems. For example, analytics or reporting.

Privitar addresses both of these considerations.

In the Privitar Platform, the Token Vault stores the mappings between the original and de-identified data. This mapping enables users to link the de-identified data back to its original value, provided the user has the appropriate permissions and the rule has been marked as reversible. This functionality is known as Unmasking in Privitar.

The Token Vault is the only place where the relationship between the original data and de-identified data is stored. Breaking the link between de-identified data stored in the Protected Data Domain (PDD) and the Token Vault enables de-identified data in the PDD to be used for analytics, while still fulfilling the RtbF needs of individuals who are included in that de-identified dataset.

In the example below, the mapping between the original ID `29602` to its de-identified ID `75820` is removed in the Token Vault. Following the removal, it is now no longer possible to identify the original value of `75820`. In effect, `29602` has been 'forgotten', but the utility of the de-identified data is preserved.



Original Data	
ID	**Date of Birth**
82759	06/02/1984
29602	15/11/1995
56217	03/07/1960

ID - Token Vault	
Value	**Token**
82759	92937
~~29602~~	~~75820~~
56217	09192

Deidentified Data	
ID	**Date of Birth**
92937	06/02/1984
75820	15/11/1995
09192	03/07/1960

Deleting token mappings can enable organisations to respond to an RtbF request in one of two ways:

- *Forget* the individual by anonymizing data. Deleting the token mapping may be sufficient to anonymise the data within the meaning of Recital 26 of the GDPR.
- Reduce the risk to the individual's rights and freedoms, allowing processing based on the 'legitimate interest' legal basis to continue. Deleting the token map makes it more difficult to re-identify an individual in the dataset, reducing the risk posed by the processing to their rights and freedoms. The GDPR allows controllers to continue processing if they can demonstrate an overriding legitimate interest for that processing to continue. Deleting the token mapping reduces the risk to the individual and may allow the legitimate interest to override that now-reduced risk to the individual.

There are some instances where deleting token mappings will not, on its own, be sufficient to meet RtbF requirements. This depends on factors including: the nature of the dataset (including whether it contains quasi-identifiers), the existence of other data and who has access to the data. For more information, see the Privitar whitepaper: GDPR: The Right To Be Forgotten.

10.1.6. Important Considerations

This section contains some important considerations when using the Remove Token Vault Mapping Feature. For more information about using the feature from the Automation API, see Removing Token Vault Mapping.

Unique Identifiers

The feature is intended to be used on unique identifiers. That is, it is designed for direct identifiers such as: Passport Numbers, National Insurance Numbers, Social Security Number, Internal System IDs. Values such as Names, Date of Birth and Country are quasi-identifiers and are not unique.

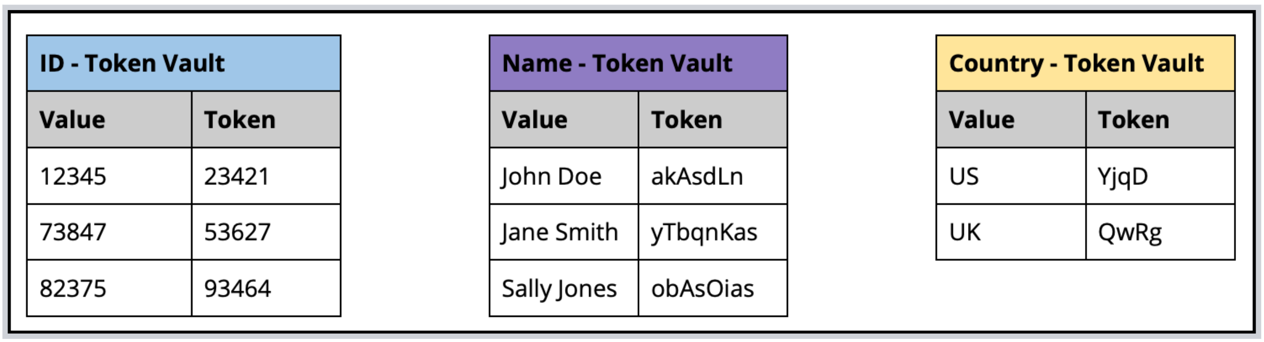
If Token Vault mappings are removed for quasi-identifiers, this will impact all records which have the same input value and use that mapping. For example, in the following dataset, there are three users from two distinct countries: US and UK.

ID	Name	Country
12345	John Doe	US
73847	Jane Smith	UK
82375	Sally Jones	US

As the countries have been tokenized consistently, US will always map to `YjqD` and UK will always map to `QwRg`.

ID	Name	Country
23421	akAsdLn	YjqD
53627	yTbqnKas	QwRg
93464	obAsOias	YjqD

In the Token Vault, there are only two mappings stored for `Country` as these are not unique:

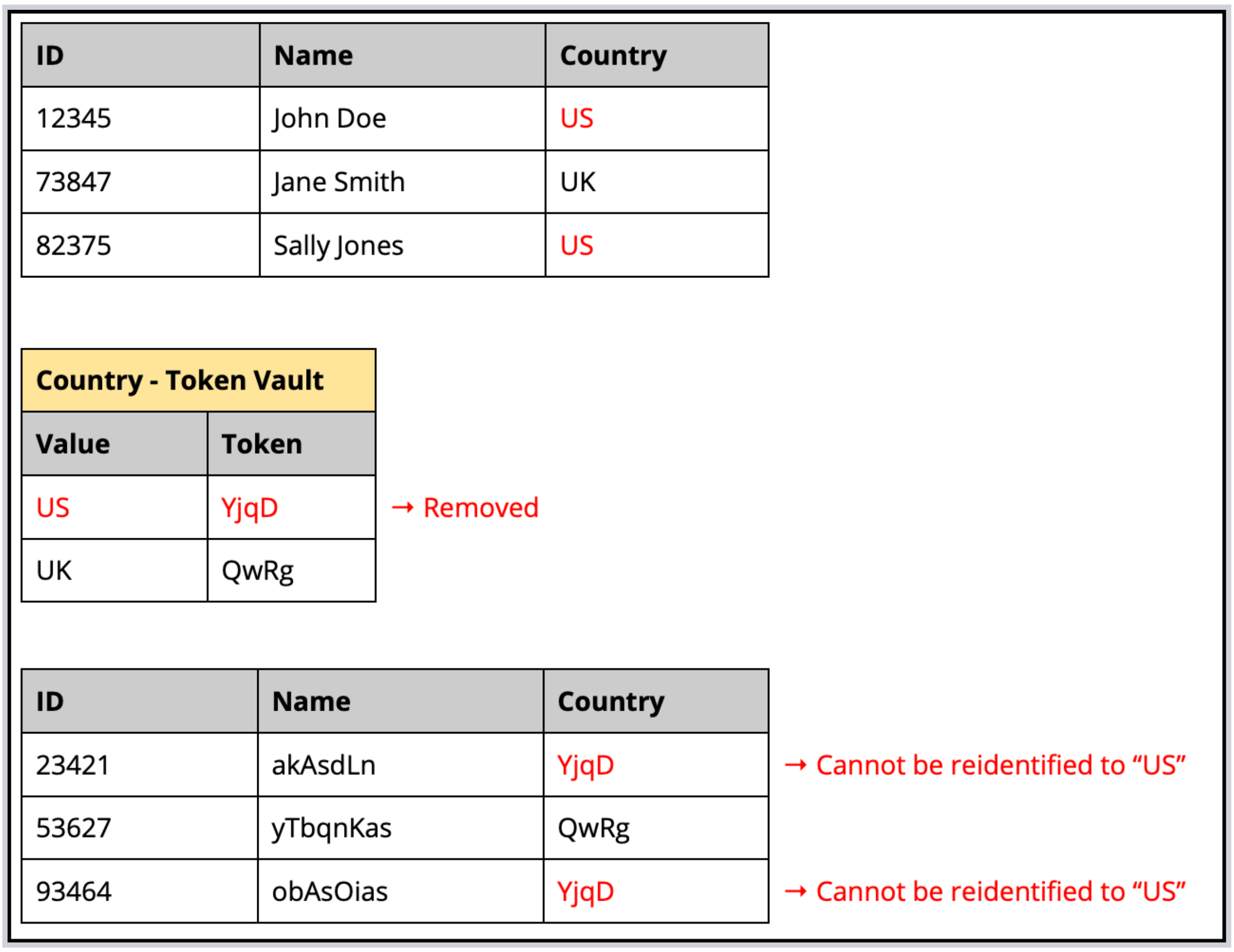


ID - Token Vault	
Value	**Token**
12345	23421
73847	53627
82375	93464

Name - Token Vault	
Value	**Token**
John Doe	akAsdLn
Jane Smith	yTbqnKas
Sally Jones	obAsOias

Country - Token Vault	
Value	**Token**
US	YjqD
UK	QwRg

If the mapping for the Country US `YjqD` is removed, this will affect **all** users associated with the Country US and prevents `YjqD` from being reidentified as US. In this case, this removal would affect both `John Doe` and `Sally Smith` in the de-identified dataset.



ID	Name	Country
12345	John Doe	US
73847	Jane Smith	UK
82375	Sally Jones	US

Country - Token Vault		
Value	**Token**	
US	YjqD	→ Removed
UK	QwRg	

ID	Name	Country	
23421	akAsdLn	YjqD	→ Cannot be reidentified to "US"
53627	yTbqnKas	QwRg	
93464	obAsOias	YjqD	→ Cannot be reidentified to "US"

Removing token mapping is irreversible, so it is important to consider if the token mapping that is being removed is unique, otherwise this could impact other records and affect the utility of a PDD.

Time Taken for a Mapping Removal – Cached Data

Although mappings could be removed from the Token Vault, they may still be present in the masking/unmasking caches. If the mapping still exists, an individual is not considered to be *forgotten* in the context of RtbF.

To ensure mappings are also removed in the cache, data in the cache is automatically removed four hours after being written.

After a *remove token mapping* request has been executed successfully and four hours have passed, the mapping stored in the cache is removed so the individual is then considered to be *forgotten*.

Token Reuse

Once a token has been assigned and the token mapping has been removed, the token will not be reused.

For example, John has the ID 12345 which maps to the token 23421. If this token mapping is removed, the token 23421 will never be reused.

If John re-enters the dataset with the same ID 12345, the ID will be assigned a completely new token.

Audit

When a *remove token mapping* request is executed, the details of the request are detailed in the Audit logs (excluding the sensitive/identifying value being removed). For more information, see Managing the Audit log.

10.2. Setting HIPAA Privacy Rules

This section describes how to use the Generalize Date rule to de-identify a dataset containing details about the age of individuals, such that it is compliant with HIPAA Privacy Rules.

To be compliant with HIPAA Privacy Rules following the *Safe Harbor* method, an Individual's Age and Date data must be de-identified following HIPAA guidelines. In summary, the guidelines are:

- If a person is under the age of 90 their date of birth needs to be generalized to only show the year. If the person is 90 or older then the date of birth needs to be generalized to only show that they are 90 or older.

For more information about de-identification requirements for patient data, refer to the HIPAA Safe Harbor field requirements.

In Privitar, the HIPAA requirement to de-identify a dataset so that it is compliant with the HIPAA Privacy rule for an Individual's Age and Date, can be achieved using the **Generalize Date** rule. This rule can be used replace a date with a constant date, if the input date is outside of a user-defined date range.

The date-range can be specified as an Absolute date or as a Relative date. The default input date format is `yyyy-MM-dd`.

For more information about the options available for use with the Generalize Date rule, see Generalize Date.

10.2.1. Setting the HIPAA Privacy Rule

To set the HIPAA Privacy Rule in Privitar, configure the Generalize Date rule as follows:

- Generalization Behavior - set to **01-01-Original value**
- Masking Behavior - set to either Absolute date or Relative date:
  - Absolute date:
    - Set constant before a date.

      If date is before, **1930-12-11**, set date to **1930-01-01**,
    - Set constant after a date.

      Do not set.
  - Relative date:
    - Set constant before a date.

If date is, **90 years** before today, set date to **1930-01-01**,

10.2.2. Setting the HIPAA Privacy Rule in the API

The following JSON file shows how to set the HIPAA Privacy Rule in Privitar using an Absolute date. In this example, the Absolute date is set to `1930-12-18`. That is, 90 years earlier than the date, `2020-12-18`. Dates later than this are generalized to `<Original-value>-01-01`:

```
{
  "team": {
    "id": "031c4fd3"
  },
  "name": "hipaa",
  "type": "GENERALIZE_DATE",
  "properties": {
    "minDate": "1930-12-18",
    "minDateReplacement": "1930-01-01",
    "generalizedMonthReplacement": 1,
    "generalizedDayReplacement": 1
  }
}
```

The following JSON file shows how to set the HIPAA Privacy Rule in Privitar using a Relative date. In this example, the Relative date is set to `90 years` (`minDateDelta` and `deltaUnit`). That is, 90 years earlier than the current date. Dates later than this are generalized to `<Original-value>-01-01`:

```
{
  "team": {
    "id": "031c4fd3"
  },
  "name": "hipaa",
  "type": "GENERALIZE_DATE",
  "properties": {
    "minDateDelta": "90",
    "deltaUnit": "years",
    "minDateReplacement": "1930-01-01",
    "generalizedMonthReplacement": 1,
    "generalizedDayReplacement": 1
  }
}
```

11. APIs

The Privitar platform offers the following APIs for integration with scripts and custom applications:

- The **Privitar Automation REST API** can be used to automate configuration and workflow processes on the Privitar platform. For example, it includes endpoints to programmatically get, create, edit and delete Schemas, Policies, Protected Data Domains and Jobs. It also can be used to execute and monitor Batch Job runs.

  For more information about the Automation API, see the Automation API Explorer in the APIs section of the *Privitar User Guide* in the Privitar UI application.
- The **Privitar on Demand API** can be used to perform Privitar On Demand (POD) operations.

  For more information about the POD API, see the POD API Explorer in the APIs section of the *Privitar User Guide* in the Privitar UI application.
- The **SecureLink Unveiler API** can be used to Unveil. That is, to re-identify tokenized SecureLink identifiers.

  For more information about the SecureLink Unveiler API, see the SecureLink Unveiler API Explorer in the APIs section of the *Privitar User Guide* in the Privitar UI application.
- The **Public Services API** can be used to obtain information about the resources used for Data Flow Jobs and POD/SecureLink Unveiler Jobs.

  For more information about the Public Services API, see the Public Services API Explorer in the APIs section of the *Privitar User Guide* in the Privitar UI application.

11.1. Automation API (v3)

The Automation API is a REST API that can be used to automate processes on Privitar. That is, programmatically get, create, edit and delete Privitar objects such as Schemas, Policies, Protected Data Domains or Jobs. It also can be used to execute and monitor Batch Job runs.

The Automation API documentation, including a list of all available endpoints and operations, is provided as an interactive explorer.

For more information about the Automation API, see the Automation API Explorer in the APIs section of the *Privitar User Guide* in the Privitar UI application.

11.1.1. Authentication

The API supports the following authentication methods that are associated with Privitar API User accounts:

- Basic Access authentication: the HTTP (or HTTPs) Basic Authentication header is required, containing the details of an existing Privitar API User.
- Mutual TLS authentication: a TLS client Certificate is required. The certificate's Common Name must match the Common Name configured in the details of an existing Privitar API User.

Basic Access authentication should be only used with HTTPS in a production environment, otherwise it's not secure (over plain HTTP)

The API Users calling specific API endpoints must have the corresponding Roles (permissions) assigned per Team.

The available authentication mode is configured per Privitar deployment. Please consult your System Administrator to know which authentication modes are available to you and appropriate for your environment.

For example, it is recommended that Basic Access authentication is only performed using HTTPS in a production environment.

The Privitar on Demand, SecureLink Unveiler and Public Services APIs are based on API Users and always require a TLS certificate and Common Name setup.

11.1.2. Error Handling

The Automation REST API returns a common set of errors. This section describes the type of error messages that are returned by the API and the format of the messages.

Error Response format

The format of the body of all error responses from the API is:

```
{
  "code" : "APPLICATION_ERROR_CODE",
  "message" : "Error X occurred.",
  "errors" : [
      {
        "code" : "APPLICATION_SUB_ERROR_CODE",
        "message" : "Sub-error Y occurred.",
        "field" : "Name"
      }
  ]
}
```

The meaning of the fields used in this format are described in the table:

Field	Description
`code`	An application error code. This can be used to uniquely identify the error.
`message`	A short message to describe the error. This will offer more context on the error. For example, the id of an entity that could not be found.
`errors`	A list of one or more sub-errors that describe the issue in more detail. One example of this is when validation fails when creating a resource. In that situation, for each missing field a different sub-error will be present in the response. It is not always present.
`errors[ ].code`	An application sub-error code.
`errors[ ].message`	A short message to describe the sub-error.

Field	Description
`errors[ ].field`	If applicable, the name of a field that isn't valid. For example, when a required field is not specified when creating a resource. If an error occurs in a top-level field, it is the name of the field that is returned in the error message. If the error occurs in a nested entity, the error message will return the full path to the erroneous field. For example, `policy.masking.mappings`.

API Errors

There are many errors that can be generated by the API. The table below provides a representative sample of the errors that may be encountered, together with an explanation as to the likely cause of the error.

HTTP Status code	Application Error code	Description
400 BAD_REQUEST	BAD_REQUEST	An error occurred due to invalid user input. If the input can be classified as being syntactically or semantically invalid, then the `INVALID_INPUT` error code is returned. However, If the input cannot be classified as `INVALID_INPUT`, then the `BAD_REQUEST` error is returned.
400 BAD_REQUEST	INVALID_INPUT	One of more of the properties that were supplied to create or modify an object are not syntactically or semantically valid.
400 BAD_REQUEST	DEPENDENCY_NOT_FOUND	One of the resources that the resource currently being created or updated relies on could not be found.
400 BAD_REQUEST	PRECONDITION_FAILED	The operation cannot be completed due to the current state of the system.
401 UNAUTHORIZED	UNAUTHORIZED	The client is not authenticated.

HTTP Status code	Application Error code	Description
403 FORBIDDEN	FORBIDDEN	The current user does not have permissions for the operation being attempted.
403 FORBIDDEN	INVALID_LICENCE_KEY	There is a problem with the licence key. It might be invalid, expired or missing. Please contact your account manager for help.
403 FORBIDDEN	FEATURE_DISABLED	A required feature for this operation is disabled. Please contact your system administrator for help.
403 FORBIDDEN	EXTERNAL_USER_MANAGEMENT	The users of the system are managed externally and cannot be accessed by this operation.
404 NOT_FOUND	NOT_FOUND	The requested resource does not exist in the system.
409 CONFLICT	CONFLICT	The resources affected by the operation have been modified concurrently.
500 INTERNAL_SERVER_ERROR	CONFIG_MISMATCH	There is an issue with the configuration of the environment.
500 INTERNAL_SERVER_ERROR	INTERNAL_SERVER_ERROR	An unexpected error occurred.
501 NOT_IMPLEMENTED		This endpoint is not implemented.
503 SERVICE_UNAVAILABLE	SERVICE_UNAVAILABLE	External components such as Privitar On Demand or SecureLink could not be reached.

API Sub-errors

The API returns the following Sub-errors. All these errors are flagged as `400 BAD_REQUEST` response status codes.

Application Error Code	Application Sub-Error Code	Description
INVALID_INPUT	MISSING_FIELD	No value was specified for a mandatory field.

Application Error Code	Application Sub-Error Code	Description
INVALID_INPUT	EMPTY_FIELD	The supplied value is empty or only contains whitespace.
INVALID_INPUT	MAX_LENGTH_FIELD	The supplied value exceeds the maximum supported length.
INVALID_INPUT	OUT_OF_BOUNDS_LIST	The specified collection of values exceeds the maximum supported amount.
INVALID_INPUT	INVALID_FIELD	The specified value is not valid for the respective field.
INVALID_INPUT	INVALID_OBJECT	One of the objects that were specified for the current operation is not valid.
BAD_REQUEST	INVALID_QUERY_PARAM	One of the query parameters is not valid.
INVALID_INPUT	INVALID_FIELD_URI	A field was expecting a valid URI but the one sent was malformed.
INVALID_INPUT	INVALID_FIELD_STRING_FORMAT	A field was expecting to conform to some pattern, but it did not.
INVALID_INPUT	INVALID_FIELD_DATE_FORMAT	A field was expecting a valid date string format (according to `Java.time.date` format) but it was malformed.
INVALID_INPUT	INVALID_TIME_FORMAT	A field was expecting a valid time string format (according to `Java.time.date` format) but it was malformed.
INVALID_INPUT	INVALID_FIELD_NUMERIC_BOUNDS	A field was expecting a number between either a minimum bound, maximum bound or both; and it violated these bounds.

11.1.3. Search Query

Endpoints that accept the `search-query` parameter support searching through a collection.

Single Term Query

In its simplest form, a valid query is expressed by a field name and a single term the field needs to match to. The match is case insensitive:

```
field:textToMatch
```

A single term is a single word with no whitespaces.

The wildcard * can be used to search for 0 or more characters and can be used at the beginning, at the end or at both ends of a term:

```
field:*matchSuffix
field:matchPrefix*
field:*matchAnywhere*
```

Boolean Operators

Multiple conditions can be expressed on the same field or on different fields via Boolean operators AND and OR. For example:

```
field:blue OR field:green
```

Multiple conditions can be grouped via parentheses to form sub-queries. For example:

```
field:blue OR (field:green AND other:red)
```

Special Characters

The following characters are reserved characters that are used for search grammar and need to be escaped when used in a term:

```
+ - ! ( ) { } [ ] ^ " ~ * ? : \
```

Phrases

A field can be exactly matched to a phrase (multiple terms) by including the phrase within quotes. Wildcards are not supported with phrase matching. For example:

```
field:"Match a multi term phrase"
```

Supported Fields

The set fields name that can be used in a search query is specific to the endpoint. The names are listed in the API specification for each endpoint.

11.2. Privitar On Demand API (v1)

The Privitar on Demand API can be used to execute Privitar on Demand Jobs. It is provided by the Privitar on Demand server.

The separate Privitar Automation REST API can be used to configure and manage, that is, get, create and delete, Masking and Unmasking Jobs on Privitar on Demand.

11.2.1. Authentication and Authorisation

The Privitar On Demand API authentication is based on API Users using mutual TLS authentication. The users must have specific Run POD permissions assigned per team for both Masking and UnMasking POD operations.

For more information about permissions, see Managing permissions for POD and SecureLink Jobs.

11.2.2. Documentation

For more information about the POD API, see the POD API Explorer in the APIs section of the *Privitar User Guide* in the Privitar UI application.

11.3. SecureLink Unveiler API (v1)

The SecureLink Unveiler API can be used to unveil (that is, re-identify), data tokenized by SecureLink Tokenization Rules that had the Produce reversible tokens option enabled. For more information, see Using the Rules Library.

The SecureLink Unveiler API expects that all required fields that have been tokenized using the SecureLink Tokenization Rule in a Policy are present at the time of unveiling. Optional fields can be missing.

11.3.1. Authentication and Authorisation

The Unveiler API authentication is based on API Users using mutual TLS authentication. The users must have specific Run Unveiler permissions assigned per team. For Remasking, the API user will need to have specific permissions created in both teams; one for the team that generates the source PDD and the other for the team that will run the Remasking operation for the destination PDD. For more information about permissions, Managing API Users.

11.3.2. Documentation

For more information about the SecureLink Unveiler API, see the SecureLink Unveiler API Explorer in the APIs section of the *Privitar User Guide* in the Privitar UI application.

11.4. Public Services API (v2)

The Public Services API can be used to obtain information about the resources used for Data Flow Jobs and POD/Unveiler Jobs. This includes: Job definitions for Data Flow jobs, POD/Unveiler URL details and Schema inputs and outputs for given POD/Unveiler Jobs.

11.4.1. Authentication and Authorisation

The Public Services API authentication is based on API Users using mutual TLS authentication. For more information about permissions, see Managing permissions for POD and SecureLink Jobs.

11.4.2. Documentation

For more information about the Public Services API, see the Public Services API Explorer in the APIs section of the *Privitar User Guide* in the Privitar UI application.

12. About Privitar on Demand

Privitar On Demand (PoD) is a server component that provides an HTTP API for applying Policies.

This enables a variety of integration use cases where de-identification can be initated from other applications and services.

12.1. Privitar on Demand Jobs

Using Privitar On Demand requires the creation of a Privitar On Demand Job. This Job defines the Policy and Protected Data Domain that the API will use when accepting data. The Job ID of a Privitar On Demand Job must be supplied when calling the Privitar on Demand HTTP API. Privitar on Demand Jobs can be created through the Privitar user interface or the Policy Manager Automation API.

API calls to Privitar On Demand Jobs with the same destination PDD will produce consistent data (provided the Preserve Data Consistency option is set on the corresponding masking Rules)

12.2. Calling Privitar on Demand

Once created, a Privitar On Demand Job is referenced from the external calling application by its Job ID. Changes to the Policy or the Job configuration are propagated periodically to the Privitar On Demand server.

Privitar provides a Java client SDK library that can optionally be used to call the Privitar On Demand APIs.

12.3. Creating a Privitar On Demand Job

A Privitar On Demand Job contains a reference to a Policy and a Protected Data Domain. It lets the Policy be applied to data supplied in an HTTP API call, with the de-identified response returned to the API caller.

To create and execute Privitar On Demand Jobs, the Environment has to be set up accordingly. For more information, see Privitar on Demand Environment Configuration.

Before creating a Data Flow Job you need to ensure that the Policy that will be used by the Job already exists and is not in Draft.

To create a Privitar On Demand Job, follow these steps:

1. Select **Jobs** from the Navigation sidebar. The Jobs page is displayed.
2. Select the **Privitar On Demand Jobs** tab.
3. Choose the Environment from the **Environment** list box.

   The chosen Environment must contain the data to be processed by the Policy, and will also be the destination of the processed data.
4. Click on **Create New Privitar On Demand Job**. The **New Privitar On Demand Job** dialog box is displayed.

5. Enter a name for the Job in the **Name** box.

   This name is used to identify the Job in the Privitar user interface and via Automation APIs.
6. Select a compatible Policy from the **Policy** list box.

   A compatible Policy must have the following features:

   - Use compatible masking rules. (For more information, see Masking Rules.)
   - Use Generalization strategies that are supported for this type of Job. (That is, not enforcing k-anonymity).
   - Use a Schema with one table.

   The **Allow local token caching check box** is enabled by default.

   Note

   When using consistent tokenization, a connection is made to the configured Token Vault to read/write tokens. To improve performance, a local in-memory cache is used. Caching can be disabled by un-checking this box, but this will have a detrimental effect on the processing performance.
7. Click on **Next**. The dialog box is updated with details about the PDD to be included in the Job.
8. Select an existing Protected Data Domain from the list or select **Add to a new Protected Data Domain** to create a new PDD.

   For more information about creating PDDs, see Creating a PDD.
9. Click on **Next** to create the Job.

   The Job appears in the index of Jobs on the **Jobs** page and the **Jobs Details** page for the Job contains the Job ID that can be used to reference the Job.

12.4. Calling a Privitar On Demand Job

Two options are available for running a Privitar On Demand Job via a Privitar On Demand server:

- Use of the Privitar client SDK Java library (JavaDocs for the client SDK are provided as part of the Privitar release bundle).
- Direct invoking of the Privitar On Demand HTTPS API. For more information, see Privitar On Demand API (v1).

Each system making an API call must be registered with Privitar as an API User with the corresponding permissions for the specific Protected Data Domains these de-identification operations involve.

The steps required to invoke Privitar On Demand programmatically are beyond the scope of this document.

13. About Privitar SecureLink

Privitar SecureLink allows datasets about a pool of individuals taken from across a group of organisations ("contributors") to be joined into combined datasets for analysis by a central organisation (the "recipient").

Taken individually, many of the contributors (and indeed the recipient) may have constraints that make it impossible to share data with another contributor. For example, commercial issues such as competition, or industry-wide regulations may prohibit the sharing of identifiable data on individuals.

SecureLink allows each individual's data, identified by a single type of direct identifier, to be joined at the recipient without revealing either the original identifier, or any party being able to join their data directly to any other party's data.

This is achieved through the use of a homomorphic encryption protocol involving an intermediary party, which obscures the original ID value as it passes through the system, and ensures it is mapped deterministically to a new value at the recipient.

13.1. How to configure SecureLink

There are two parts to the configuration required to use SecureLink in Privitar.

Note

Please note that the configuration instructions given below assume that the required SecureLink server components are already available.

The configuration required is as follows:

- Create appropriately configured Environments for the recipients. This action only needs to be performed once per recipient.
- Create Jobs that the contributors and recipient will use day-to-day for pooling data via SecureLink.

13.2. Creating an Environment for a SecureLink Recipient

A recipient organisation has to configure an Environment in a certain way to be compatible with its contributors.

The Environment for a recipient requires a Privitar On Demand server and a SecureLink Intermediary server:

1. Select **Environment** from the Navigation sidebar.
2. Select the **SecureLink** tab.
3. Enter the location of the Intermediary server in the **Intermediary URL** field.
4. If Unveiling is required, enter the location of the Unveiler server in the **Unveiler URL** field.

5. Select the **On Demand** tab.
6. Enter the location of the Privitar On Demand server in the **Privitar On Demand URL** field.
7. Click **Save**.

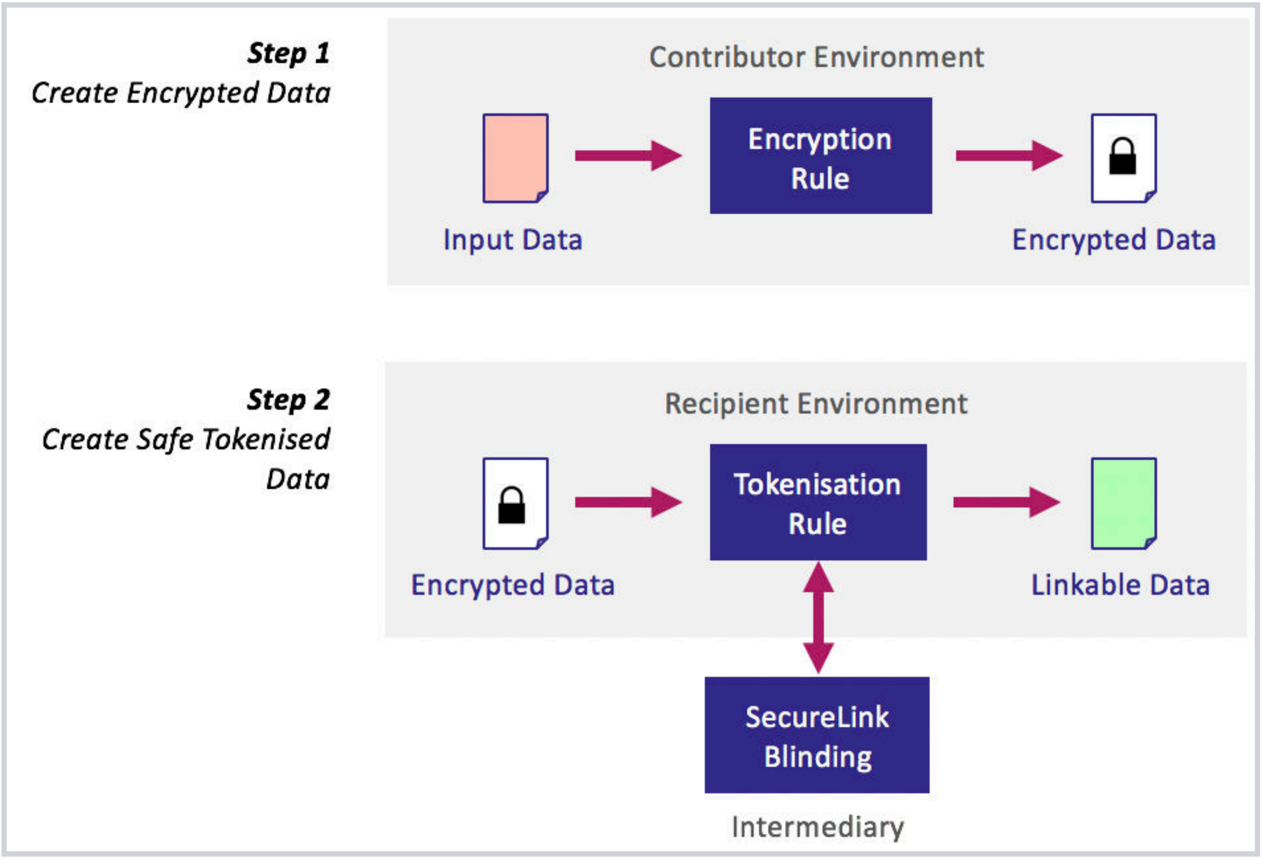
13.3. Creating SecureLink Jobs

In addition to the SecureLink Environments, two Jobs are required, corresponding to the *contributor* and *recipient* parts of the SecureLink process.

The process is summarized as follows:

1. *Contributor*: This Job uses the SecureLink Encryption Rule to encrypt direct identifiers in input data. These Jobs can be run through Batch, Data Flow or Privitar On Demand. The Job references the appropriate SecureLink destination environment in its configuration.
2. *Recipient*: This Job uses the SecureLink Tokenization Rule to convert the encrypted direct identifiers produced by the contributor into tokenized identifiers. These Jobs can only be run through Privitar On Demand (not Batch or Data Flow).

Together, these steps implement the SecureLink process.



The following sections describe how to configure Jobs for the two steps.

13.3.1. Contributor Jobs

Contributor Jobs produce encrypted ciphertext from raw direct identifiers. This is performed by a *SecureLink Encryption Rule*. The Policy for this Job must include this Rule.

1. Create a *SecureLink Encryption* Rule. For more information, see Using the Rules Library.
2. Create a Policy. For more information, see Creating a Policy.

   In the Policy, ensure that the SecureLink Encryption Rule is applied to the direct identifier field. For more information, see Setting a Masking Rule.
3. Create a Batch, Data Flow or Privitar On Demand Job, referencing the Policy created earlier. For more information, see What is a Job?.

   Note

   Ensure that the SecureLink destination Environment is set correctly on the contributor's Job.
4. Running this Job will produce output containing encrypted identifiers suitable for use with a recipient Job.

13.3.2. Recipient Jobs

Recipient Jobs consume encrypted output from a Contributor Job, and produce tokenized linkable identifiers. This is performed by a SecureLink Tokenization Rule. The Policy for this Job must include this Rule.

1. Create a *SecureLink Tokenization* Rule. For more information, see Using the Rules Library.

   The Rule includes a regular expression option that determines the format of the linkable identifier. As with other tokenization rules, output from this Rule will be random, but consistent within a PDD.
2. Create a Policy. For more information, see Creating a Policy.

   In the Policy, ensure that the SecureLink Tokenization Rule is applied to the field containing the encrypted value.
3. Create a Data Flow Job, referencing the Policy created above. For more information, see What is a Job?.

Running this Job will produce output containing linkable tokenized identifiers.

13.4. Rotating SecureLink Keys

SecureLink relies on four secret values that together give it its security characteristics. These secret values are of two types:

- **Encryption keys used for communication between SecureLink components**. These are held by the Intermediary, Unveiler (if configured) and Recipient components (such as Privitar On Demand).
- **A SecureLink blinding secret**. This is a secret value that is used by the SecureLink Intermediary to compute secure base values that are used as input to SecureLink Tokenization. It is held by the Intermediary.

13.4.1. Key Rotation

It is good practice to periodically replace encryption keys with new versions. This process is called key rotation. Privitar supports rotating both types of secret on a per-Environment basis. Either the encryption keys or the blinding secret may be rotated at one time. The process is initiated from the **Environments** page, as described below.

> **Note**
>
> The process may take a period of time to complete, and that the Environment may not be edited while the rotation is in progress. Other Privitar operations, such as running Jobs, are not affected.

To rotate secret values for a specific Environment:

1. Select **Environments** from the Navigation sidebar.
2. Select the required SecureLink Environment, and from the **Actions** list box, choose either **Rotate SecureLink Keys** or **Rotate SecureLink Blinding Secret**.
3. Click **OK**.

   The appropriate key(s) will now be rotated. Environments where this process is running are displayed with an indicator.

13.4.2. Determining When Last Key Rotation Occured

To determine when the last key rotation was completed:

1. Select **Environments** from the Navigation sidebar.
2. Select the required Environment by clicking on the name of the Environment in the **Name** column.
3. Select the **SecureLink** tab.

   The time of the last key rotation is displayed.

14. The Instrumentation System

The Privitar Instrumentation System is an analytics and diagnostics tool for the Privitar Data Privacy Platform. It is designed to aggregate performance data and product usage pattern information to better understand User behavior, so that you can get direct feedback about the efforts of your privacy enhancing programs. It is provided as a standard feature of the Privitar Platform.

The Instrumentation System implements an end-to-end data pipeline from an installation of the Privitar Data Privacy Platform at a customer site back to an aggregated data store owned and managed by Privitar.

14.1. Overall Architecture

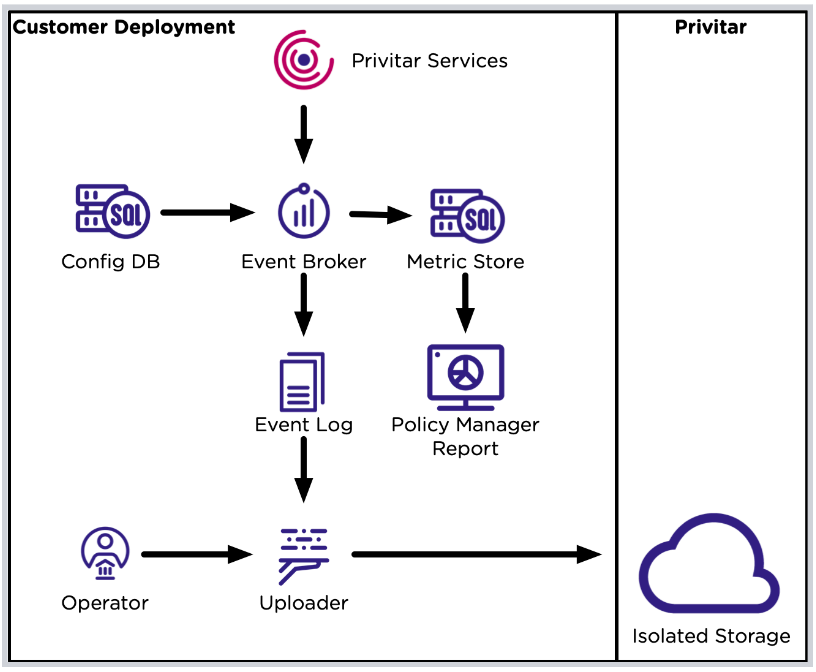
The architecture consists of the following components:

- An **Event Broker** that collects events that have been produced as a result of various tasks being undertaken on the Privitar Platform. Events are collected in two ways:
  - Events are sent directly to the Event Broker from various services running on the Privitar Platform (**Privitar Services**).
  - The Event Broker runs a background service that queries the Privitar Policy Manager configuration database (**Config DB**) to gather data about Jobs that have been run and any additional audit data that has been generated.
- Event data from the Event Broker is copied to two storage locations:
  - An **Event Log** containing the events that have been collected by the Event Broker regarding the usage of the Privitar Platform. The events are written to the Event Log in JSON format.
  - A **Metric Store** that stores aggregated metric data about the usage of the product. This data is presented as a report that can be accessed from Policy Manager.
- A command line tool (**Uploader**) that is used to upload the Event Log files produced by the Event Broker to a Privitar-owned, secure data store (**Isolated Storage**). The type and location of the data store is agreed with the customer.
- The process for running the Uploader is managed and can be configured by the customer (**Operator**). To ensure that no data leaves the customer environment without the consent of the customer, the Uploader is configured by default to be triggered by the designated operator at the customer premises.

These components work together to form a two-step process of collecting the event data and subsequently uploading the data to a Privitar data storage location.

Both steps are performed separately and asynchronously.

The operation of these components is shown in the diagram below:



From the customer side:

- Data is generated by Jobs running from any Privitar Platform, including the Batch Processor running on Policy Manager, together with jobs generated from POD, Data Flow Jobs (NiFi) and from the SDK.
- From the Policy Manager, the customer will be presented with a screen showing aggregated figures for Privitar de-identification operations and other figures indicating overall usage of Privitar.

From the Privitar side:

- Data received from the Telemetry Uploader is stored in a Privitar-owned secure storage, such as an Amazon AWS S3 bucket. Depending on customer policy, the Uploader will run periodically, will retry on failure and remove files from the Event Log when they have been successfully uploaded.
- Privitar will use data analytics tools to analyze the data returned from the customer site.

14.2. Event Broker

The Event Broker is responsible for:

- Converting historic audit and Job run data into the new event format.
- Periodically checking for new product usage data.
- Writing events to the Event Log file.
- Calculating and storing data metrics.
- Receiving events from other Privitar services.

14.2.1. Operation

The first time that the Event Broker is started a service runs and queries the Privitar configuration database (ConfigDB) for all events that have taken place on the Privitar platform since the platform was operational. These historical Events are extracted from the Event and JobRun tables stored in the database. (This service will not be run again.)

The Event Broker also receives events directly from Privitar services. These events are passed through by the Event Broker where they are processed to create metric data and written directly to the Event Log.

14.2.2. Configuration

The Event Broker is configured using a properties file (in a similar manner to the `application.properties` file that is used to configure other features of the Privitar Platform).

The Event Broker configuration options include:

- The type of database to be queried. Options include: PostgreSQL, Oracle and MySQL.
- The username and password credentials for the Configuration Database. These credentials can be encrypted in the same way that these details are encrypted in the main application.properties file.
- Startup options for the Event Broker.

14.2.3. Event Log

Event data is stored in the Event Log in JSON format. Files will be split when they reach a certain configurable size. These are the files that will be sent back to Privitar or that you can analyze at your site for your own data analytics.

14.2.4. Metric Store

Specific events are processed by the Event Broker to create running metrics that can be displayed in the Policy Manager. The Metric Store captures the number of PPOs and the number of records processed over time.

These figures are stored in a separate schema/database to the configuration database. Values are aggregated and written every minute to reduce the number of rows being created in the database.

The Metric Store supports the same database types as the configuration database.

14.3. Uploader

This section describes the functionality of the Uploader.

14.3.1. Overview

The Uploader is responsible for sending Event Logs back to Privitar. It is a separate component from the Event Broker. The Uploader will copy events from the Event Log to a Privitar-owned secure data store such as an Amazon AWS S3 bucket.

14.3.2. Operation

The Uploader is a tool that is manually executed from the command-line (CLI). The command to invoke the uploader is wrapped in a script with the correct initial arguments set. It is not a long running service and will only upload when manually triggered.

In order to ensure that the customer has complete control over the sending of data from their organization back to Privitar, it is the customer who manages and configures the operation of the Uploader. By default, the Uploader must be triggered by an operator at the customer side in order for data to be sent back to Privitar.

By default, the uploader checks whether files already exist in the S3 bucket before uploading to ensure that we don't re-upload files every time the uploader is executed. This can be manually turned off.

The uploader also expects log files to be in the file structure that the event broker writes them to. This is to protect the user from accidentally uploading from the incorrect location. It is also possible to turn this option off.

14.3.3. Configuration

There are a series of command-line options that configure the operation of the Uploader:

- The folder where the Event Log is stored.
- The location details of the remote storage. For example, for an S3 bucket the region and the name of the S3 bucket.
- A Preview option to view the Event Logs that will be copied to the remote store.

14.4. About the Event Data

This section provides an overview of the Event data that is captured, including the type of data that is collected, how the data is stored to ensure there are no privacy or security concerns and the format of the stored data.

14.4.1. Data Capture Process

There is a two-step event data capture process:

1. Gather instrumentation data from Privitar into Event Log files, using the Event Broker.
2. Use the Uploader to upload the files to a Privitar-provided, over secure communication to a secure data store such as an Amazon AWS S3 bucket.

These steps are performed separately and asynchronously. To ensure that no data leaves your environment without your consent, the Uploader is configured to be triggered by you from your environment.

14.4.2. Data categories

The data is captured in the form of events that are triggered by a certain defined event taking place on the platform.

The data capture process captures both historic event data as well as current event data.

Historic data

Event level data for the following instrumentation categories are extracted from pre-existing logs in your current configuration database:

- Schema
- Policies
- PDDs
- Rules
- Jobs
- Batch Job runs
- Team
- Users
- Environments

These intermediary files will be generated into synthetic events that can be written to the Event Log file.

All objects like Schemas, Rules, Policies and PDDs will only be identifiable by random IDs. When the events are captured, information that can be considered business sensitive, such as names or any personal identification data will be stripped from the event before it is added to the Event Log.

It is possible to confirm that this de-identification of the data has occurred by downloading the data and reviewing all the generated files in a text editor.

Current Data

Privitar services, such as Policy Manager generate event-level data that is sent to the Event Broker. The following categories of event can be generated:

- Schema: created, updated and deleted
- Policy: created, updated and deleted
- PDD: created, updated, deleted and closed
- Rules: created, updated and deleted
- Jobs: created, updated, deleted, started
- Job_Run: Batch and Data Flow
- PPO: number of PPOs
- Teams: created, updated and deleted
- Team membership: added and removed
- Token mapping removal: start and finish
- Users: created, updated, logon, type of user
- Environments: created, updated and deleted
- Data Flow Processors: created, closed and metrics
- Policy Manager: started
- Service Status: current status and changes to the current status

For each event that is generated, the following details are always included:

- `event_type`; type of event generated.
- `timestamp`; time and date that the event was created.
- `unique_id`; unique identifier for the event.
- `event_source`; indicates if the event is taken from the Config database or directly from a service.

There will also be additional details captured that are specific to the event.

For example, this is the format of the event (`PDD_CREATED`) that is generated when a PDD is created:

```
"@timestamp": "2020-06-23T08:37:05.801Z",
    "eventSource": "SYNTHESISED",
    "eventType": "PDD_CREATED",
    "parameters": {
      "watermarked": "false",
      "team": {
        "id": "66586",
        "uniqueId": "qk7oshxo"
      },
      "uniqueId": "hzkmk54j",
      "metadata": [
        "Recipient",
        "Description",
        "Approver",
        "Intended Use",
        "Restrictions"
      ],
      "objectType":
"com.privitar.agrotera.dashboard.domain.ProtectedDataDomain",
      "environment": {
        "id": "66609",
        "uniqueId": "yqblk6y4"
      },
      "id": "66613"
    },
    "timestamp": "2020-06-23T08:37:05.801",
    "uuid": "f2b5848c-b85b-4d2a-8ad7-d73eb1004ec4",
    "@version": "1",
    "customer": "test-upload"
```

For more information about the details that are captured for each event, refer to the "Event Types" chapter of this document.

15. Environments Administration

There are two main groups of tasks that are associated with the administration of the Privitar platform:

- Environment Administration tasks.
- Superuser Administration tasks.

This section describes **Environments Administration** tasks. For more information about the administration tasks performed by Superusers, see Superuser Administration

15.1. Environment Administration

Environment-related administrative tasks in Privitar include the following topics:

- Environments are used to configure infrastructure such as Hadoop clusters, Token Vaults, Privitar On Demand servers, SecureLink or Key Management Systems that are available to specific Teams.
  - For more information about creating, deleting and sharing Environments, see Managing Environments.
  - For more information about operating Privitar in a Hadoop cluster, see Administering in a Hadoop Cluster.
- Import and Export include features for importing and exporting Schemas, Policies, Rules, and Jobs between Privitar installations. For more information, see Importing and Exporting Configurations.

15.2. What are Environments?

Environments are used to configure infrastructure such as Hadoop clusters, Token Vaults, Privitar On Demand servers, SecureLink or Key Management Systems that is available to specific Teams.

In Privitar, an Environment defines:

- The Token Vault connection and authentication details.
- The Key Management System connection and authentication details.
- The Hadoop Cluster that:
  - Contains input data and output data in HDFS or Hive
  - May store Token Vaults in HDFS
  - Handles the work of anonymization processing
  - Contains encryption keys in a KMS
  - Contains user and role information for access control
  - Provides the catalogue for metadata sharing
- Privitar On Demand server settings
- SecureLink settings
- AWS Glue settings

Please note that not all of the above settings are required or available in all deployments. Which Environments settings are available will depend also on the deployment location. For example, AWS Glue settings will only be available in AWS deployments.

Privitar supports adding multiple Environments. Environments that have been created can be deleted and also shared between Teams.

When a Job is run, the Environment in which the Job runs can be selected. For example, an HDFS path in one Environment may not be valid in another; or a data flow platform may not have access to the configured Token Vault to perform consistent tokenization. It is therefore important to select the correct Environment when using Privitar.

15.3. Managing Environments

This section describes all the tasks associated with Privitar Environments.

15.3.1. AWS Glue Environment constraints

AWS Glue Environments have the following constraints:

- They are not able to reference Hadoop Clusters.
- The only supported Token Vault type is **AWS DynamoDB**.

Please choose **Custom** instead of **AWS Glue** when creating a new Environment to configure an Hadoop Cluster or use a different Token Vault type.

15.3.2. Creating and Editing an Environment

To create or edit an Environment:

1. Choose **Environments** from the navigation menu.
2. Select **Custom** from the **Create New Environment** list box or select the name of the Environment from the **Name** column of an existing Environment.
3. Configure the Environment. For more information, see:
    - Token Vault Environment Configuration
    - Key Management Environment Configuration
    - Privitar on Demand Environment Configuration
    - SecureLink Environment Configuration
    - Hadoop Cluster Environment Configuration
    - AWS Glue Environment Configuration

15.3.3. Deleting Environments

To delete an Environment:

1. Choose **Environments** from the navigation menu.
2. Select the row of the Environment you want to delete and click **Actions > Delete**.
3. Confirm the Deletion by clicking **Delete**.

15.3.4. Sharing Environments

To support collaboration and multi-tenancy patterns, Environments can be shared with other Teams. A Shared Environment can be used by any team to create and run Jobs, but can only be amended by the Team that created it. In particular:

- By default, an Environment is not shared after creation. That is, the Environment is only available to the Team that has created it.
- If an Environment is configured to be shared, it is shared globally. That is, the Environment becomes available for use to all Teams.
- Only the Team that has created and owns the Environment is able to delete or amend it. Every other Team has read-only access.
- In order to ensure autonomy, the visibility of Policies, Protected Data Domains, Jobs and Schemas continues to be limited to the creator of these objects.

Other things to note about Shared Environments are:

- Once an Environment has been shared it cannot be unshared. That is, the only way to discontinue its use by other Teams is to delete it.
- If an Environment with Hadoop cluster(s) is shared, any participating Team will be able to execute a Job on the Hadoop cluster using any of the defined Service Users.
- When triggering Watermarking Investigations using Privitar, you can only select Environments that have been created by the Team you are currently logged into. You are not able to select any global Environments that were created by other Teams.

To share an Environment that has already been created:

1. Choose **Environments** from the navigation menu.
2. On the Environments index page, select the row of the Environment you want to share and click **Actions > Share** .
3. Confirm Sharing by clicking **Share Environment**.

   The Environment is now available for global use to other Teams and marked as **Shared** on the **Environments** page.

15.4. Key Management Environment Configuration

There are various settings that need to be configured for any type of key management system (KMS) that is enabled in an environment:

Setting	Description
Key Management System	Select which type of KMS to use. The following KMS options are available, depending on whether you are using Hadoop: • **Hadoop** (default Hadoop KMS) • **Ionic Machina** (optional KMS for use with Hadoop) • **AWS Secrets Manager** (can be used with or without Hadoop) • **HashiCorp Vault** (can be used with or without Hadoop) If you select **None**, any hashing rules, encryption rules, derived tokenization, and HDFS Token Vault encryption will not be available for this environment.

Setting	Description
KMS Location	**Hadoop KMS URL:** The URL of the Hadoop KMS (if Hadoop is selected as the KMS).
	Ionic Machina Persistor Path: The path to the Ionic Machina Persistor (if Ionic Machina is selected as the KMS.) **Note** For more information about setting up Ionic Machina as the KMS for the Privitar platform, see the separately provided *Ionic Machina Reference Guide*. (Please contact Privitar for further information about Ionic Machina integration.)

15.4.1. AWS Secrets Manager KMS

Privitar supports AWS Secrets Manager as a key management system (KMS).

Create a KMS Key to Be Used by the AWS Secrets Manager

Note

This is optional but recommended for production environments.

To learn more: AWS Documentation: Creating symmetric CMKs

1. Open AWS Key Management Service (KMS) and create a symmetric key.
2. Define the administrative permissions. Select an admin user who can manage and rotate the key, if needed.
3. Define the key usage permissions. This needs to be a role with permissions to the data processor. You can provide permissions to the same role to which you provided access to AWS Secrets Manager.
4. Create the key and make note of the key ID. You will use this key ID when you configure the KMS environment in Policy Manager.

AWS Secrets Manager KMS Configuration

To configure AWS Secrets Manager as the KMS in the Policy Manager:

1. In the Environments settings, click the **Key Management** tab.
2. **Key Management System**—Select "AWS Secrets Manager."
3. **AWS Region**—Set this to the AWS region that you will use to access the AWS Secrets Manager. For example, `us-west-2`. To ensure consistent hashing, it is important to use AWS Secrets Manager from the same region. If the region is not defined, the Privitar processing engine will use the default region (for example, POD, Hadoop nodes, or SDK), which might be subject to change.
4. **AWS Endpoint**—Set this to the URL of the AWS endpoint that is used to make a private connection between your VPC and AWS Secrets Manager. When you use a VPC

service endpoint, communication between your VPC and Secrets Manager occurs entirely within the AWS network and requires no public Internet access. For security, we recommend that you create this endpoint. For more information, see AWS Secrets Manager - User Guide.

5. **AWS KMS Key**—We recommend defining a master key that AWS Secrets Manager will use to protect every secret that it stores. For more information, see AWS KMS Developer Guide.
6. **AWS Secrets tags**—Enter the tags that will be attached to any secrets created by the Privitar Platform.
7. **NOVLT Key Name**—Enter the key name.

For AWS Secrets Manager in Glue Environments, you cannot configure the KMS through the user interface or an API. The configuration will be as follows:

- **AWS Region**—Always the same as the region of the Glue environment
- **AWS Endpoint**—The default endpoint (not configurable for Glue environments)
- **AWS KMS Key**—The AWS account's default customer master key (CMK), the one named `aws/secretsmanager`, for the region
- **AWS Secrets tags**—Tags are the ones provided to cloud formation as the TAGS parameter

15.4.2. HashiCorp Vault KMS

Privitar supports HashiCorp Vault as a key management system (KMS). Note that despite the name, HashiCorp Vault is a KMS, not a token vault.

To learn more about HashiCorp Vault, visit HashiCorp's training materials: https://learn.hashicorp.com/collections/vault/getting-started

HashiCorp Vault KMS Requirements

The platform requires the KV Secrets Engine - Version 2 (https://www.vaultproject.io/docs/secrets/kv/kv-v2) for key storage.

Although each secret stored in HashiCorp Vault can contain multiple key-value pairs, the platform only reads the value stored under the key “value”.

HashiCorp Vault KMS Adapter

The Privitar Policy Manager automatically supplies the correct adapter for Hadoop.

To use HashiCorp Vault as a KMS in other dataflow environments (POD, SDK, Apache NiFi, and StreamSets), set the adapter JAR file on the classpath of the environment. The JAR file will be called either `adapters-kms-hashicorp-vault` or `privitar-kms-hashicorp-vault-driver`.

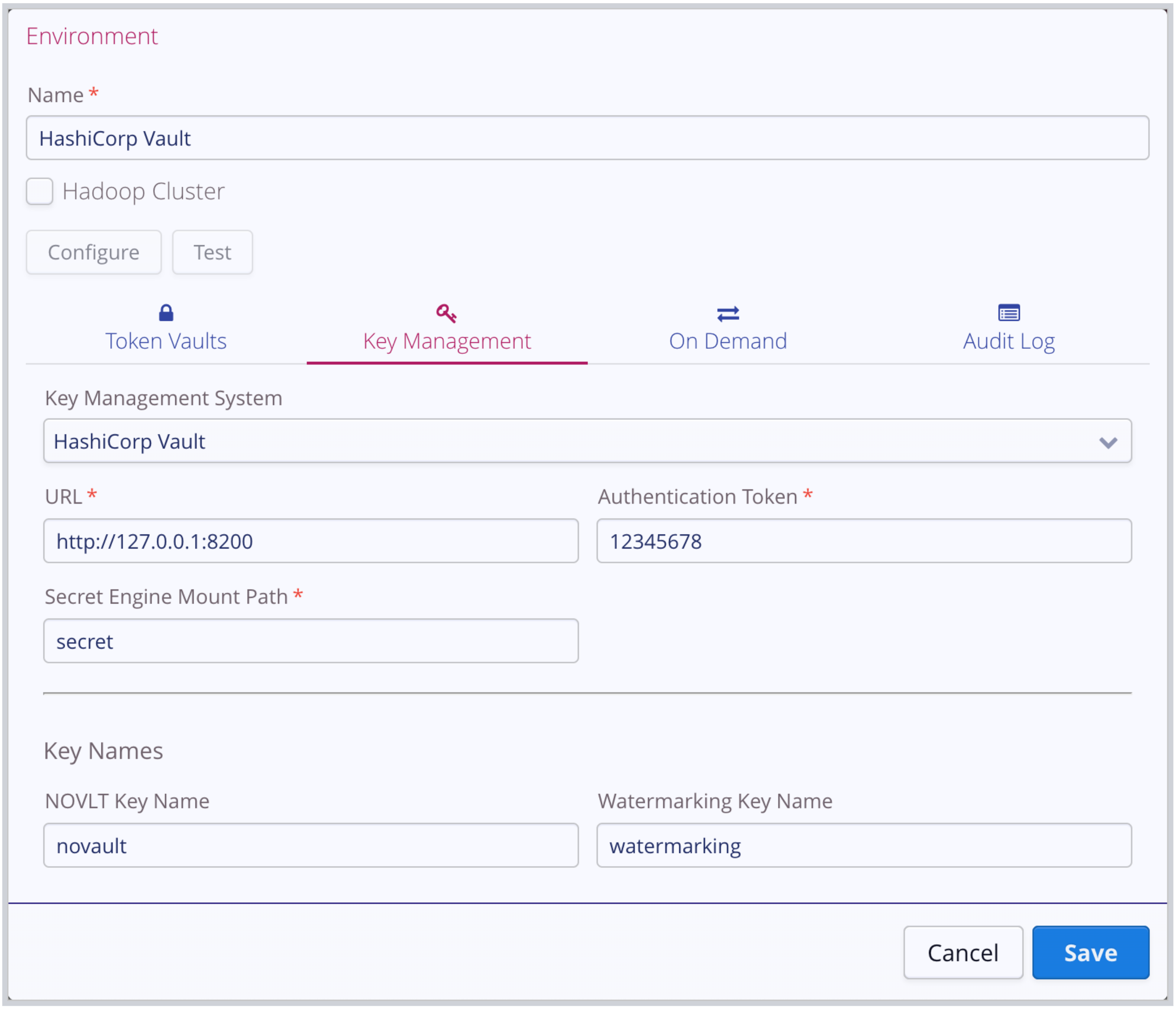
HashiCorp Vault KMS Authentication

The platform can authenticate with HashiCorp Vault KMS through authentication tokens.

HashiCorp Vault KMS Configuration

To configure HashiCorp Vault as KMS in the Policy Manager:

1. In the Environments settings, click the **Key Management** tab.
2. **Key Management System**—Select "HashiCorp Vault."
3. **URL**—Enter the URL where the platform can contact HashiCorp Vault KMS.
4. **Authentication Token**—Enter the token ID.
5. **Secret Engine Mount Path**—Enter the path for the KV Secrets Engine (see HashiCorp Vault KMS Requirements).
6. **NOVLT Key Name**—Enter the key name.



Name * HashiCorp Vault

☐ Hadoop Cluster

Key Management System: HashiCorp Vault

URL *: http://127.0.0.1:8200

Authentication Token *: 12345678

Secret Engine Mount Path *: secret

Key Names

NOVLT Key Name: novault

Watermarking Key Name: watermarking

Key Generation Example

To generate a secure, random key called "test" under a KV Secrets Engine mounted at the path “secret,” you would run:

```
echo -n `openssl rand -base64 32` | vault kv put secret/test value=-
```

This example assumes that you have already authenticated with a vault.

15.5. Privitar on Demand Environment Configuration

Configure the Privitar On Demand server to be associated with this Environment.

Setting	Description
Privitar On Demand URL	HTTP URL for the Privitar On Demand server that should be associated with this Environment.
Use Privitar On Demand to close Protected Data Domains with JDBC and DynamoDB token vaults	When **enabled**, Protected Data Domains (PDDs) that use JDBC or DynamoDB token vaults will be closed via Privitar On Demand. This setting is recommended and enabled by default when a Privitar On Demand Environment is configured. When this option is **disabled**, Privitar will close PDDs with JDBC and DynamoDB token vaults.

15.6. SecureLink Environment Configuration

Configure the SecureLink settings for this Environment.

Setting	Description
Intermediary URL	HTTP URL for the SecureLink Intermediary server. This is the server that is used by SecureLink Tokenization rules in this Environment.
Unveiler URL	HTTP URL for the SecureLink Unveiler server. If configured, the SecureLink Unveiler supports the controlled re-identification of linkable SecureLink tokens produced by SecureLink Tokenization Rules.

15.7. Hadoop Cluster Environment Configuration

A Hadoop Cluster can optionally be added to an Environment.

To add a Hadoop cluster, select the Hadoop Cluster checkbox and click the **Configure** button to open the Hadoop Cluster configuration dialog box. The dialog box contains six tabs. Each tab has a number of settings that can be used to configure the Hadoop cluster.

- Authentication
- Spark
- Hive
- Data Locations
- Cluster
- Metadata

In addition, a **Hadoop Cluster Test** button is provided to validate the primary aspects of the configuration.

15.7.1. Authentication

Setting	Description
Use Kerberos	Enable this option to use a Kerberos keytab to authenticate with a Hadoop cluster. When checked, the following settings are required: • **Keytab path**: the location of the keytab file on the local filesystem. • **Principal**: name of the principal from the keytab that should be used. (In some Privitar installations, a Primary Principal and a Secondary Principal option may be available as part of a failover recovery strategy. In this case, Privitar will use one of the Principals to authenticate with the cluster. (For more information about setting up Primary and Secondary principals, see Failover Recovery Configuration). • **Key Distribution Centre (KDC) URL**: the URL of the Kerberos Key Distribution Centre. This can be found in the `/etc/krb5.conf` file on the cluster, under the `admin_server` entry for the desired Realm. • **Realm**: The Kerberos Realm to use for this Environment. The list of available realms can be found in the `/etc/krb5.conf` file on the cluster.
Protected Data Domain output paths are managed by Sentry	Enable this option if permissions on Protected Data Domain (PDD) HDFS output paths are managed by Sentry. This prevents Privitar from setting permissions and ownership of PDD output directories and files. PDD output directories must be created and the permissions set outside Privitar. Any user running Jobs must have read/write/execute access on the PDD output directories.

The Job Authentication section determines the user that Batch Jobs are run as.

In all cases, the user specified is impersonated in the cluster when the Job runs. This relies on appropriate configuration in the cluster to allow impersonation of the user. If specified above, authentication is performed using a Kerberos keytab.

Setting	Description
Privitar user	Run Jobs as the OS-level user that is running Privitar. If the Job is run in a Kerberized cluster, this is the Kerberos Principal.
Service user	Specify a list of users that can run Jobs in the cluster. For each user, specify the group that should own the folders and files created when running the Jobs on the cluster. The user specified must belong to the group, or the Jobs will fail to run. The choice of which user to use is made in the **Run as user** box in the Job itself.

15.7.2. Spark

The Spark settings specify parameters for submitting Spark jobs used to perform processing in Privitar. The best settings for these parameters are dependent on the specification of the Hadoop cluster and the characteristics of the input data.

The following table provides a basic overview of the available parameters. For more information on tuning, please contact your system administrator.

Setting	Description
Master	Choose `yarn-cluster` to submit jobs to YARN (Recommended). Otherwise, use one of the `local[x]` options to submit Spark jobs directly.
Driver Memory (GB)	Heap memory allocated to the Spark Driver.
Executor Memory (GB)	Heap memory allocated to each of the Spark Executors.
Executors	Number of Spark Executors requested from YARN when the Job is submitted.
Executor Cores	Number of cores per Spark Executor requested from YARN when the Job is submitted.
Memory Overhead (MB)	Amount of additional memory allocated to the Spark Executor in addition to the Executor memory. This memory is allocated on top of the heap memory.
Allow Jobs to override Spark settings	This setting determines whether individual Jobs that use this Environment can supply their own values for the parameters on this page, overriding the values that are currently displayed in the dialog box. If this check box is enabled, the **Spark** tab on the Job configuration dialog box can be edited.
YARN Queue Name	The name of the YARN queue that Spark jobs will be sent to when executing Privitar Jobs configured to use this Environment. If left blank, the setting defaults to no queue.

15.7.3. Hive

Privitar can read data, write data, and import Schemas from Hive databases in the cluster.

In order to enable this functionality, the `hive-site.xml` file must be added to the **Cluster** tab, and the settings on the **Hive** tab should be reviewed.

A summary of the available settings is provided in the table below:

Setting	Description
Hive JDBC Connection URL	The JDBC URL for the cluster's Hive installation. This value is used when importing Schemas from Hive databases. During the import process, Privitar connects using JDBC to read Hive table definitions. These definitions are converted to Privitar Schemas.
Enable Hive Batch Jobs	Select this option to allow the use of Hive Batch Jobs in the Privitar APIs and user interface. This allows users to create and execute Hive Batch Jobs that apply Policies to data stored in Hive databases.
Thrift URL (read only)	The Thrift URL is automatically determined based on the hive-site.xmlconfiguration added on the Cluster tab. This value is used while executing Hive Batch Jobs to allow Privitar to communicate with the cluster's Hive installation when querying or writing data.
Warehouse location (read only)	The Warehouse location is automatically determined based on the hive-site.xml configuration added on the Cluster tab. This location is often used by Hive installations as a central location for Hive data.

Setting	Description
Job output root location	When executing Hive Batch Jobs, the HDFS location of output Hive data is controlled by the Job's PDD. This setting determines whether PDDs in this Environment specify the location individually on a per-PDD basis, or whether all PDDs write their data underneath a single location. The choice depends on your organisation's preference for storing Hive table data files. If **Allow PDDs to specify their own** is selected, then PDDs are individually configured with the location that should be used to hold output Hive table data. This location is specified as part of the PDD creation process. If **Specific location** is selected, then the single location for all output Hive table data must also be specified. When Hive Batch Jobs are run, table data is written to HDFS in a location formed as follows: `Configured location / Database name.db / Table name` where the database name is specified in the PDD, and table name is specified in the Hive Batch Job. If desired, this can be set to the **Warehouse location** that is also displayed on the Hive tab.
Hive output table type	This setting determines whether Privitar creates internal or external tables when running Hive Batch Jobs. The choice determines the effect on underlying HDFS data if tables are later dropped using Hive. If Internal is specified, dropping the table will also remove the underlying data in HDFS. If External is specified, dropping the table only 'unlinks' the data from Hive, leaving the files on HDFS unchanged. The choice depends on your organisation's preference for storing Hive table data files. For more information about Hive tables, see the Apache Hive documentation.

15.7.4. Data Locations

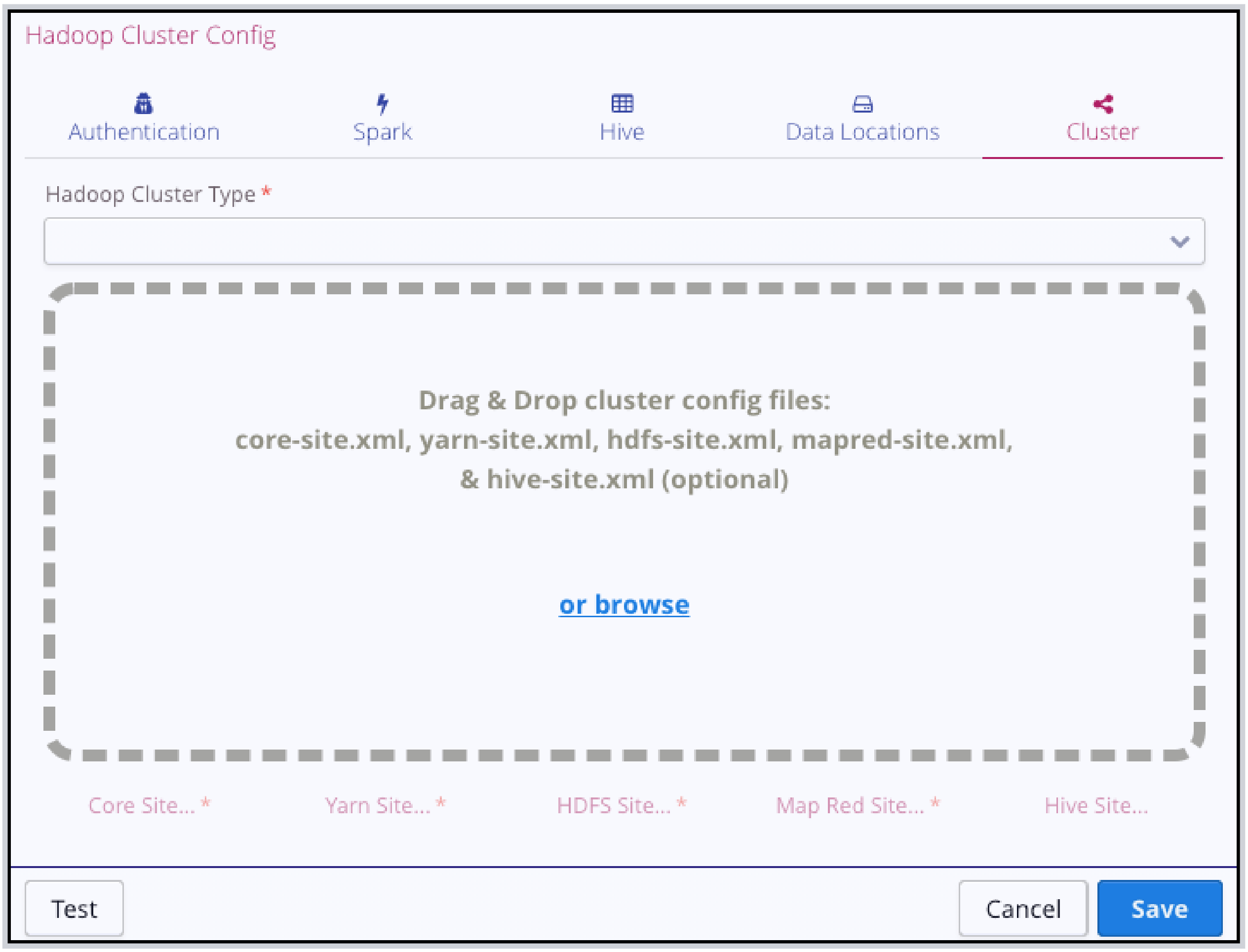
The options available for specifying data locations for Hadoop is described in the following table:

Setting	Description
Lookup Files Path	Location of lookup mappings for the Lookup masking rule. In the Policy configuration for a Lookup rule, a file containing the replacements used by that rule is specified. Lookup Files Path is the base location for such files.
Report Output Path	Location of internal status files written by Privitar during the execution of a Job. No customer data is stored in these files.

Setting	Description
Dropped Rows Path	When reading CSV data containing malformed rows, Privitar can write a sample of these malformed rows into a location specified here, for debugging purposes. The number of rows written is capped at 100 rows. If no location is specified, no rows are written.

15.7.5. Cluster

In the **Cluster** tab, select the cluster type from the **Hadoop Cluster Type** list box:

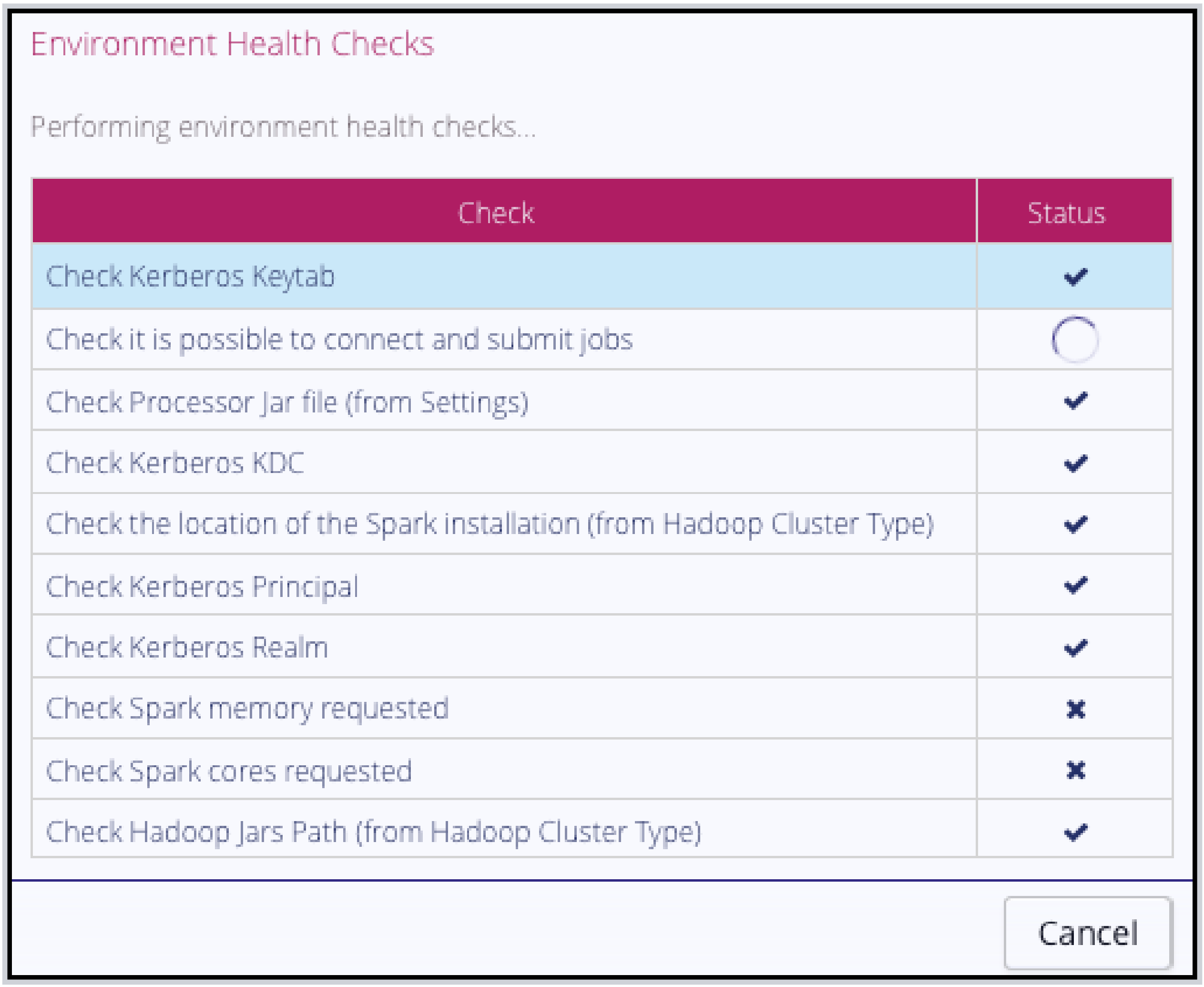


Add the configuration files to the cluster configuration by dragging them onto the dialog box. (The `hive-site.xml` configuration file is optional.) Typically, the configuration files are provided by a Hadoop administrator.

For more information on creating new Cluster types, see Creating Hadoop Cluster Types.

15.7.6. Hadoop Cluster Test

Hadoop Clusters provide the ability to test themselves, initiated from the **Test** button on the **Environment** dialog box and Hadoop Cluster Config dialog box:


Environment Health Checks

Performing environment health checks...

Check	Status
Check Kerberos Keytab	✓
Check it is possible to connect and submit jobs	○
Check Processor Jar file (from Settings)	✓
Check Kerberos KDC	✓
Check the location of the Spark installation (from Hadoop Cluster Type)	✓
Check Kerberos Principal	✓
Check Kerberos Realm	✓
Check Spark memory requested	✖
Check Spark cores requested	✖
Check Hadoop Jars Path (from Hadoop Cluster Type)	✓

Using this button causes a sequence of checks to be performed on the Privitar platform that verify:

- The location of the Privitar .jar file and Spark binaries are correctly specified in the **Settings** dialog box.
- That it is possible for the Privitar application to connect to the Hadoop cluster and successfully submit processing jobs.
- That the Environment is able to successfully allocate the cores and memory requested on the Spark tab.

If any of these checks fail, a debugging message is displayed in the dialog box.

15.8. AWS Glue Environment Configuration

The **AWS Glue Environment** tab is only available when Privitar AWS has been deployed.

15.8.1. AWS Glue Settings

The following settings can be used to customize the AWS Glue Environment:

Setting	Description
Maximum number of Glue executors	The maximum number of Spark executors that AWS Glue Batch Jobs will request to run a Batch job. This option can be overridden for each Job run. **Note** Increasing this number will increase the cost of running this job in AWS. Contact your system administrator if you are in any doubt.
Configured cloud native region	The region that Privitar is going to use to: • Import AWS Glue Data Catalog schemas • Run AWS Glue ETL Jobs • Store DynamoDB tables. The default region is the region that was specified when Privitar AWS was deployed.

15.8.2. AWS Glue Environment constraints

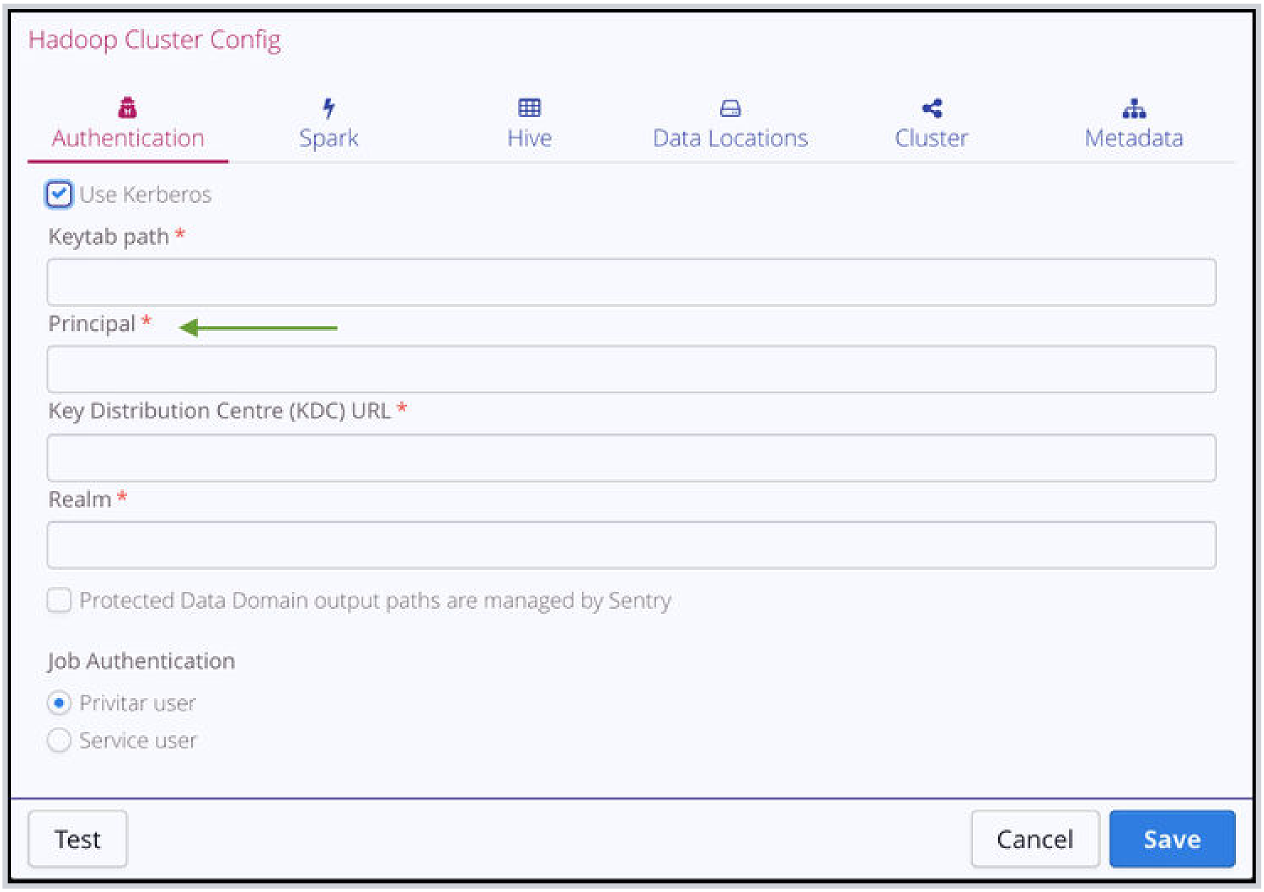
AWS Glue Environments have the following constraints:

- They are not able to reference Hadoop Clusters.
- The only supported Token Vault type is **AWS DynamoDB**.

Please choose **Custom** instead of **AWS Glue** when creating a new Environment to configure an Hadoop Cluster or use a different Token Vault type.

15.9. Failover Recovery Configuration

In a standard installation of Privitar that uses Kerberos to authenticate with the Hadoop cluster, a single Principal is defined.


Hadoop Cluster Config

[✓] Use Kerberos

Keytab path *

Principal *

Key Distribution Centre (KDC) URL *

Realm *

[ ] Protected Data Domain output paths are managed by Sentry

Job Authentication

[● Privitar user]

○ Service user

However, in an environment with an overall Failover strategy for a Hadoop cluster using Kerberos authentication, it is possible to setup Privitar to be part of this. In this case, two instances of Privitar are installed:

- One instance is defined as the current **Hot** instance running on one machine.
- The other instance is defined as the backup **Cold** instance running on another machine.

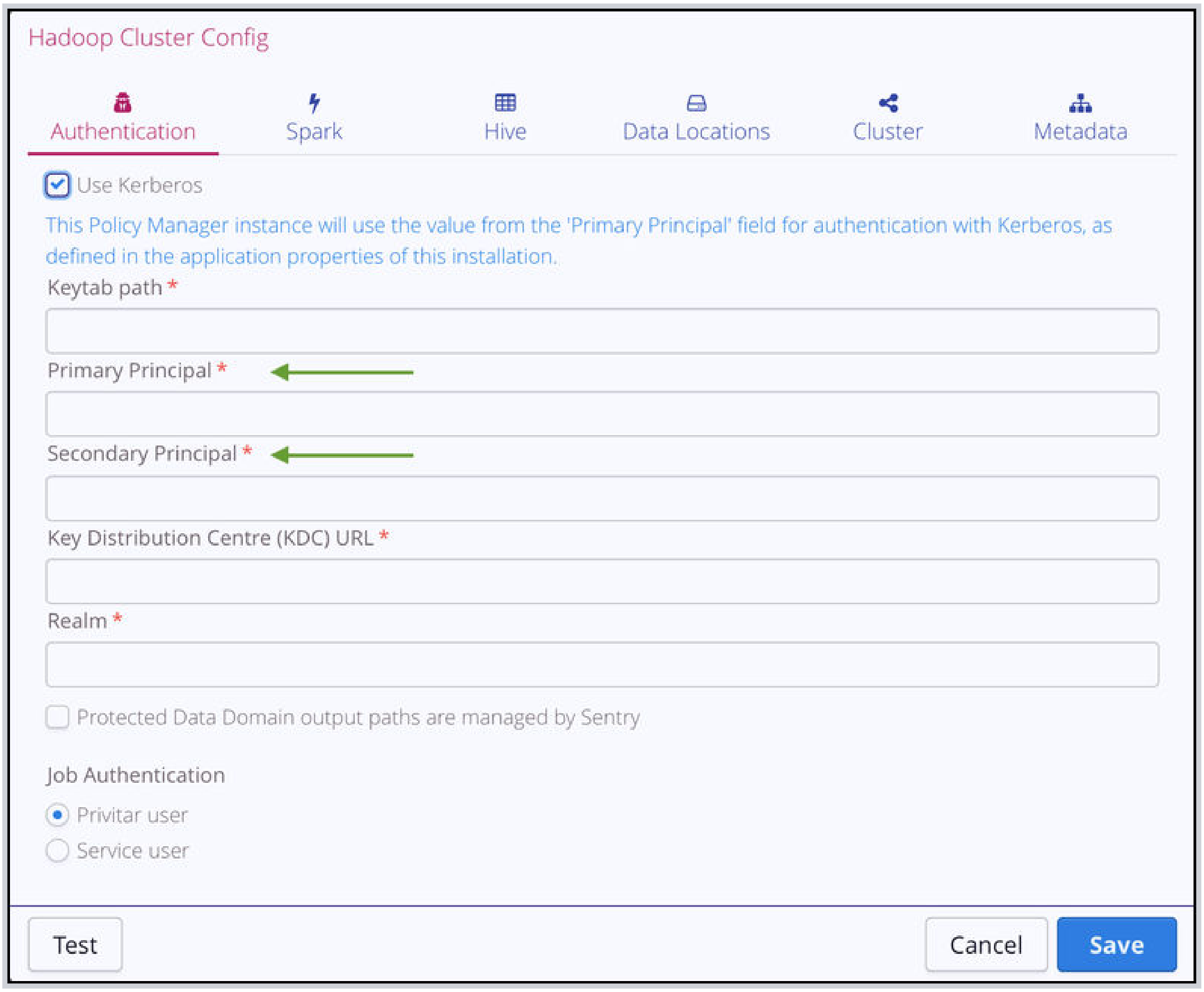
In the event of the Hot instance failing, the Cold instance takes over.

Both instances point to the same configuration database, but they will use different Principals (Primary or Secondary) to authenticate with the Hadoop cluster.

Note

Privitar does not incorporate the functionality to switch between the two instances. It is assumed that switching to the secondary machine is enabled as part of the overall Failover strategy.

The Primary and Secondary principals are defined in the **Authentication** tab of the **Hadoop Cluster Config** dialog box:


Hadoop Cluster Config

[✓] Use Kerberos

This Policy Manager instance will use the value from the 'Primary Principal' field for authentication with Kerberos, as defined in the application properties of this installation.

Keytab path *

Primary Principal *

Secondary Principal *

Key Distribution Centre (KDC) URL *

Realm *

☐ Protected Data Domain output paths are managed by Sentry

Job Authentication

● Privitar user

○ Service user

The table below provides an example of a Failover recovery setup, with two instances of Privitar running on separate machines. (Machine names are examples.)

Instance	Machine name	Principal to Use	Principal Settings (Primary/ Secondary)
1 (Hot)	host-001.realm.com	PRIMARY	**privitar/ host-001.realm.com@REALM.COM**
			privitar/ host-002.realm.com@REALM.COM
2 (Cold)	host-001.realm.com	SECONDARY	privitar/ host-001.realm.com@REALM.COM
			privitar/ host-002.realm.com@REALM.COM

In the above example, the **Hot** instance will use the Kerberos Primary principal for authentication:

`privitar/host-001.realm.com@REALM.COM)`

The **Cold** instance will use the Kerberos Secondary principal for authentication:

`privitar/host-002.realm.com@REALM.COM`

For more information on configuring more than one instance of Privitar, contact your system administrator.

15.10. Token Vault Environment Configuration

When tokens are generated using rules with the **Preserve Data Consistency** option enabled, the produced tokens are stored in a Token Vault. (For more information about the Rules that support tokenization, see Masking Rule Types Supporting Unmasking.)

The Environment settings on the **Token Vault** Configuration tab configure how these tokens are generated and stored.

The following Token Vault Types are supported:

- **None** (no Token Vault is used, therefore no consistent tokenization is possible within this environment)
- **HDFS**
- **HBase** (and Google Cloud **Bigtable**)
- **JDBC**
- Amazon **DynamoDB**

HDFS or HBase Token Vaults can only be selected if a Hadoop Cluster is configured in the Environment.

Amazon DynamiDB is the only Token Vault supported when using Privitar AWS in an AWS Glue Environment.

15.10.1. HDFS Token Vault

When selecting HDFS Token Vault the following settings are available:

Property	Description
Vault Path	HDFS location under which files containing Token Vaults will be written.
Use Derived Tokenization	If checked, random token generation is seeded using a value derived from the encrypted value of the input. The token can be generated from the input without checking a Token Vault, which makes the tokenization process significantly faster. If this strategy is selected, a **Derived Tokenization Key Name** must be specified. (This requires a Key Management System (KMS) to have been configured for use with Privitar. For more information, see Key Management Environment Configuration.) The tokens produced still exhibit uniform randomness, and the input value from which any token was derived cannot be guessed without knowledge of the encryption key used to encrypt the input value.

Property	Description
Token Vault Encryption	If set, a **Vault Encryption Key Name** must be specified. (This requires a Key Management System (KMS) to have been configured for use with Privitar. For more information, see Key Management Environment Configuration.) Any changes to this setting are not automatically applied to existing Token Vaults. Changes are reflected in existing Token Vaults when: • A Job is run. Any affected Token Vaults are updated to reflect the new settings. • The **Update Encryption** action is invoked from the Environments list page. This action starts a background process that applies any changed settings to all existing Token Vaults referenced by the Environment. This applies to both enabling or disabling Token Vault encryption.

15.10.2. HBase and Google Cloud Bigtable Token Vault

Select **HBase Token Vault** for both HBase and Google Cloud Bigtable clusters. The following settings are available:

Property	Description
hbase-site.xml	The HBase XML configuration. To generate a `hbase-site.xml` file for Google Cloud Bigtable, please see the Google Cloud documentation.
Namespace	The HBase namespace that this Token Vault should reside in. The default name is **default** but it is highly recommended to change it to a custom Namespace. **Note** Privitar will not automatically create a Namespace; it must exist before use.
Column Family	The HBase column family this Token Vault should use. The default is **F**. **Note** Privitar will not automatically create a new column family; it must exist before use.
Batch size	The number of values per batch that is sent to HBase for tokenization. This is a performance-related parameter. The recommended default is 1000.
PDD Deletion Thread Count	The maximum number of threads that will be used to delete a PDD.

Property	Description
HBase Jar Paths	The additional HBase Jar files that are required for using Google Cloud Bigtable. The Jar files required will be specific for each setup. Each Jar file pathname must be defined as an absolute pathname and each separate pathname must be separated by `;`. This setting is optional. It is only required when using Google Cloud Bigtable.

When using an HBase Token Vault. the location of the client JAR on the Hadoop cluster nodes must be specified in the `yarn-site.xml` in `yarn.application.classpath`, in the Hadoop Cluster configuration.

If using HBase with Kerberos:

- Keytabs must be installed on all nodes connecting to HBase.
- The following properties must be added to `hbase-site.xml`:

```
<!-- User principal for HBase -->
<property>
  <name>hbase.kerberos.principal</name>
  <value>hbase@PRIVITAR.COM</value>
</property>
<!--- Location of keytab on nodes and Privitar -->
<property>
  <name>hbase.keytab.file</name>
  <value>/etc/security/keytabs/hbase.headless.keytab</value>
</property>
```

Note

Kerberos is not currently supported by the Google Cloud Bigtable Token Vault.

15.10.3. JDBC Token Vault

Only Oracle and PostgreSQL JBDC connections are currently supported.

When selecting JDBC Token Vault, the following settings are available:

Property	Description
URL	The JDBC URL of the RDBMS database.
Username	The username to authenticate with the database.
Password	The password to authenticate with the database.
JDBC Driver JAR Path	The path to the JDBC Driver on the Privitar filesystem. This driver will be used by the Spark Jobs running on the Hadoop cluster to connect to the RDBMS. This must therefore be set only when a Hadoop cluster is configured in the Environment.

Property	Description
KMS Key name	The name of the key used to encrypt the database credentials when they are shared with the Spark Jobs in the Hadoop Cluster. This name must be set when a Hadoop cluster is configured in the Environment, and it also requires a Key Management System (KMS) to have been configured for use with Privitar. For more information, see Key Management Environment Configuration.

Note

If you are using **CyberArk** to store user credentials, there will also be settings for **CyberArk descriptor queries**. The descriptor queries are used to retrieve the Token Vault username and password from CyberArk.

For more information about configuring CyberArk for use with the Privitar platform, see the separately provided *CyberArk Reference Guide*. (Please contact Privitar for further information about CyberArk integration.)

15.10.4. Amazon DynamoDB Token Vault

Choose DynamoDB and configure the settings in the table (double-click on the values to fill in the fields). The table is already populated with sensible default values, so there is no need to change any of the fields. Required fields are marked with an asterisk.

15.11. Importing and Exporting Configurations

Privitar provides the ability to save the definition of a Schema, a Policy, a Rule, or a Job to a portable file that can be transferred between systems. This allows a configuration to be developed on one instance of Privitar, and then transferred to one or more secondary instances.

This has several uses:

- Produce a set of configuration items on a **development** instance and deploy it to a **production** instance.
- Create a library of reference configurations for various scenarios and deploy them onto multiple instances.
- Create backups or 'known good' configuration restore points.

Note

The imported JSON file has to conform to the specific Privitar configuration format. It is not possible to import files from previous Privitar versions.

15.11.1. Importing Linked Configuration Items

When a Schema, Policy, Rule or Job is exported, enough information is included in the file to reconstruct the item in its entirety. This means that:

- Exporting a Policy also includes all of its associated Rule definitions, and a Schema definition.
- Exporting a Job also includes a Policy definition with all of its Rule definitions, and a Schema definition.

When importing, Privitar provides the choice of whether these linked configuration items should also be imported as new items, or whether existing compatible items should be used.

For example, consider importing a Policy. The file also contains a Schema definition. There are two ways the Policy can be imported:

- Create a new Policy and a new Schema.
- Create a new Policy, linked to an existing Schema that is compatible with the Policy.

These choices are presented when a Policy is imported.

15.11.2. Exporting a Configuration

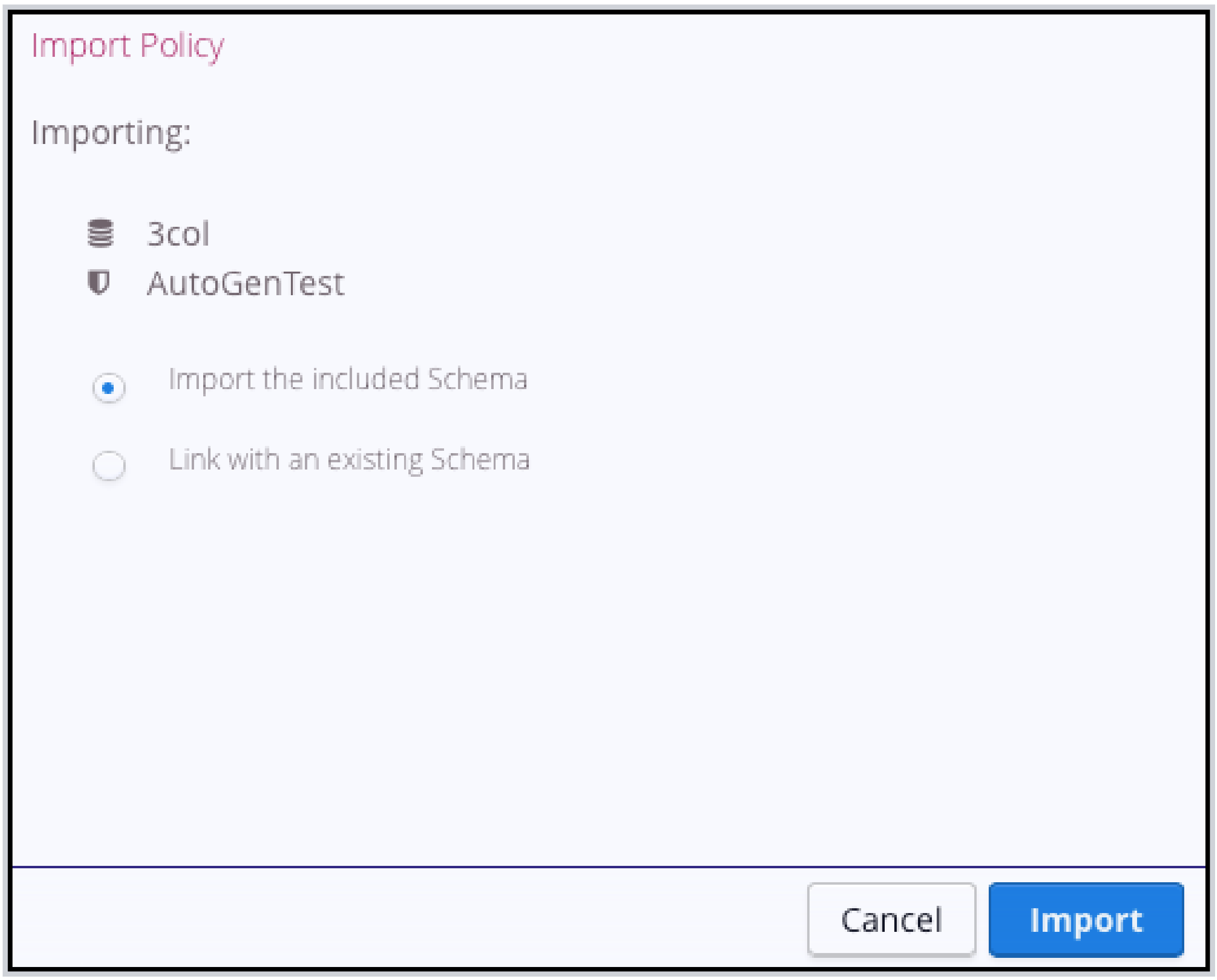
To export the configuration file for a Schema, Policy, Rule or Job:

1. Locate the item to be exported in the appropriate index listing.
2. Select the item using the selector in the first column of the index listing.
3. Select **Export** from the **Actions** list box.
4. The selected item will be downloaded to your local computer.

15.11.3. Importing a Configuration

To import the configuration file for a Schema, Policy, Rule or Job:

1. Use the navigation panel to access the index listing for the type of item to be imported.
2. Select **Import** from the **Actions** list box.
3. Either drag the file onto the dialog, or use the link to browse for the file locally.
4. Follow the steps on the **Import Policy** dialog box to indicate which items should be imported. For example, the dialog box for importing a Policy is:



Select whether any items that are associated with the imported item should be imported or linked with an existing item.

5. Click the **Import** button.

   The file is imported into Privitar.

   Imported items are distinguished from existing items in Privitar with similar names using a suffix that indicates the time of creation. For example:

   ```
   Employees Schema (imported 2017-10-21 14:36:25)
   ```

16. Superuser Administration

There are two main groups of tasks that are associated with the administration of the Privitar platform:

- Environment Administration tasks.
- Superuser Administration tasks.

This section describes **Superuser Administration** tasks. For more information about administering Environments in Privitar, see Environments Administration.

16.1. Superuser Administration

Superusers are overall system administrators for the Privitar platform and manage all aspects of Privitar once it has been fully installed and is operational.

A separate **Superuser** account is provided in Privitar. Once logged in as Superuser, a variety of tasks and settings are provided on the Privitar navigation panel:

- **Teams** are workspaces for groups of Users that collaborate together on a de-identification project.
- **Roles** define the custom user permissions of Privitar that can be assigned per Team.
- **Users** are user accounts for Privitar.
- **API Users** have access to the Privitar On Demand and Unveiler APIs, as well as the Privitar Automation API.
- **Settings** are general Privitar settings.
- **Metadata Attributes** define the metadata field configurations.
- **Cluster Types** define Privitar environments that use different Hadoop Cluster types.
- **Usage Reporting** tracks and logs information about daily usage of Privitar across all Teams.
- **Service Health** displays the health status of all connected Privitar services.
- **Audit** tracks and logs Privitar user and configuration activity.

For more information about assigning Superuser permissions to a user, see Managing Users.

16.2. What are Teams?

Privitar **Teams** represent workspaces for groups of Users and API Users that collaborate together on a de-identification project. Users and API Users are assigned specific Roles in their Teams. To be part of a Team, a User or API User has to have at least one Role in it. Users and API Users can be assigned to one or more Teams and have one or more Roles within a Team.

All Users within a Team (that is, all Team members), will be able to view all of the basic Privitar configuration objects (including Schemas, Policies, Rules, Jobs, Protected Data Domains) for that Team, independently of their Roles. And similarly, all API users within a Team will be able to read (GET) all basic Privitar configuration objects for that

Team, independently of their Roles. The specific actions (for example, creating, editing or deleting) that Team members can perform on these objects and other settings depend on the Users' Roles (that is, permissions), in that Team.

Only Rules as well as SecureLink destinations are shared with other Teams. That is, they can be viewed and used in Policies and Jobs by all other Teams. Any given Rule and SecureLink destination, however, can only be changed or deleted by the owning Team (the Team that created the object). All other Privitar configuration objects (Schemas, Policies, Jobs, Protected Data Domains, Environments) are not shared with other Teams.

If you are part of multiple Teams, you can switch between your Team spaces by clicking on the list box alongside your username in the top-right of any Privitar page, then choosing **Switch Teams** and selecting the Team of your choice.

To be able to create or edit Teams as well as to assign Users/API Users and Roles to a Team you have to be a Superuser.

16.3. Managing Teams

This section describes all the tasks that can be performed to manage Teams and to assign Users to Teams.

16.3.1. Creating a Team

To create a Team:

1. Select **Teams** from the Superuser navigation sidebar.
2. Click on **Create New Team**.

   The **Create New Team** dialog box is displayed.
3. Enter a name for the new team in the **Name** box and optionally complete the **Description** and **Contact** boxes for any additional information you want to add about the new Team.
4. Click on **Save** to create the new Team.

Note

A Superuser is not by default added to any Team. If you want the Superuser account to be a member of a Team, you have to assign the account to the Team workspace. For more information, see ???.

16.3.2. Editing a Team

To edit the details of a Team:

1. Select **Teams** from the Superuser navigation sidebar.
2. Click on the name of the Team to edit in the **Team** column.

   The **Edit Team** dialog box is displayed.

3. Edit the details of the Team.
4. Click on **Save** to update the details of the Team.

16.3.3. Deleting a Team

To delete a team:

1. Select **Teams** from the Superuser navigation sidebar.
2. Click on the name of the Team to edit in the **Team** column.

   The **Edit Team** dialog box is displayed.
3. Click on **Delete** to delete the selected Team.

   Before the Team is deleted, Privitar will conduct a series of checks for Privitar objects that have been created by members of the Team and have a dependency on that Team to remain in use:

   - If these checks reveal no dependencies, you will be asked to confirm the deletion. If you confirm, then the Team will be deleted.
   - If these checks reveal that there are dependencies such as Schemas and Policies that are owned by the Team, these will be displayed and you will be asked to confirm the deletion. If you confirm, then the Team will be deleted along with the listed Schemas and Policies.
   - If these checks reveal any critical dependencies, such as an Environment that is owned by the Team, then the deletion request will be denied and you will be provided with a list of the dependencies that must be deleted *before* the Team can be deleted.

   For more information on the dependencies that are checked and the necessary action that must be taken before a Team can be deleted, see Team Deletion Dependency Check.

16.3.4. Assigning (human) Users to a Team and Managing their Roles

Every User that is created in Privitar needs to be assigned to a Team and be given Roles within that team. To assign (human) Users to a Team and give them a Role within that Team:

1. Choose **Teams** from the Superuser navigation menu.
2. Click on the **Manage Team Members** button next to a Team.

   The Team name dialog box is displayed showing all the Roles in the Team together with the number of users assigned to each role. Clicking on the Role name in the **Role** column displays the users who have been assigned to that Role in the Team.
3. Click on the Role name in the **Role** column that you want to assign Users to.
4. Follow one of the steps depending on how users are managed in your Privitar setup:

   a. If a local database is used for identity management, click **Add User** to add new Users to the list.

   b. If LDAP/Active Directory is used for identity management, click **Add LDAP group** to add new User groups to the list.

   c. If SAML SSO is used for identity management, click **Add SAML group** to add new User groups to the list.

After any of these actions, a dialog box is displayed.

5. Select one or more Users/User groups listed in the dialog box and click on **Add** to add the selected User/User groups to the currently selected Role in the Team.
6. Repeat the previous three steps to assign other Roles.
7. Click on **Save** to update the Team with the new Users and assignments.

16.3.5. Assigning API Users to a Team and Managing their Roles

To assign API User to a Team and give them a Role within that Team:

1. Choose **Teams** from the Superuser navigation menu.
2. Click on the **Manage API Users** button next to a Team.

   The Team name dialog box is displayed showing all the Roles in the Team together with the number of users assigned to each role. Clicking on the Role name in the **Role** column displays the users who have been assigned to that Role in the Team.
3. Click on the Role name in the **Role** column that you want to assign Users to.
4. Follow one of the steps depending on how users are managed in your Privitar setup:
   - If a local database is used for identity management, click **Add User** to add new Users to the list.
   - If LDAP/Active Directory is used for identity management, click **Add LDAP group** to add new User groups to the list.
5. Select one or more Users/User groups listed in the dialog box and click on **Add** to add the selected User/User groups to the currently selected Role in the Team.
6. Repeat the previous three steps to assign other Roles.
7. Click on **Save** to update the Team with the new Users and assignments.

16.3.6. Team Deletion Dependency Check

The following table lists the dependency checks that are performed by Privitar in response to a request to delete a Team from Privitar, together with the advised action to take.

Check	Advised action to take
First check	The number of related Environments, Rules and Watermark investigations owned by the Team are displayed. You will be asked to delete these objects (Environments, Rules and Watermark investigations) before the Team can be deleted.
Does the Team own any Environment?	
Does the Team own any Rules?	
Are there any outstanding Watermark Investigations?	
Second check	The number of related Shared Environments are displayed together with the number of PDDs and Jobs that are dependent on this Shared Environment. (The details of each Shared Environment can be downloaded as a CSV file from a **Download** link.)
Does the Team own any PDDs? (Shared Environment)	

Check	Advised action to take
Does the Team own any Jobs? (Shared Environment)	You will be asked to delete these objects (PDDs and Jobs) before the Team can be deleted. (As the Environment is shared with other Teams it would not need to be deleted.)
Third check	All related Schemas and Policies will be displayed. (The details of the Policies and Schemas can be downloaded as a CSV file from a **Download** link.)
Does the Team own any Schemas?	
Does the Team own any Policies?	If you confirm the Team deletion request, then the Schemas and Policies will be deleted along with the Team.

16.4. Managing Roles

Privitar supports a range of User and API User actions such as Environment configuration, Policy definition and running of Jobs.

These actions are subject to Role-based access control and can be assigned on a per-Team basis. Roles define the actions that a User or API User is able to take in a given Team. That is, control the permissions of Users and API Users in a Team. While Privitar is installed with default Roles, these can be customized and new Roles, that is new combinations of permissions for specific actions, can be defined.

Note

All Users within a Team, that is all Team members, will be able to view all of the basic Privitar configuration objects (including Schemas, Policies, Rules, Jobs, Protected Data Domains) for that Team, independently of their Roles. And similarly, all API users within a Team will be able to read (GET) all basic Privitar configuration objects for that Team, independently of their Roles.

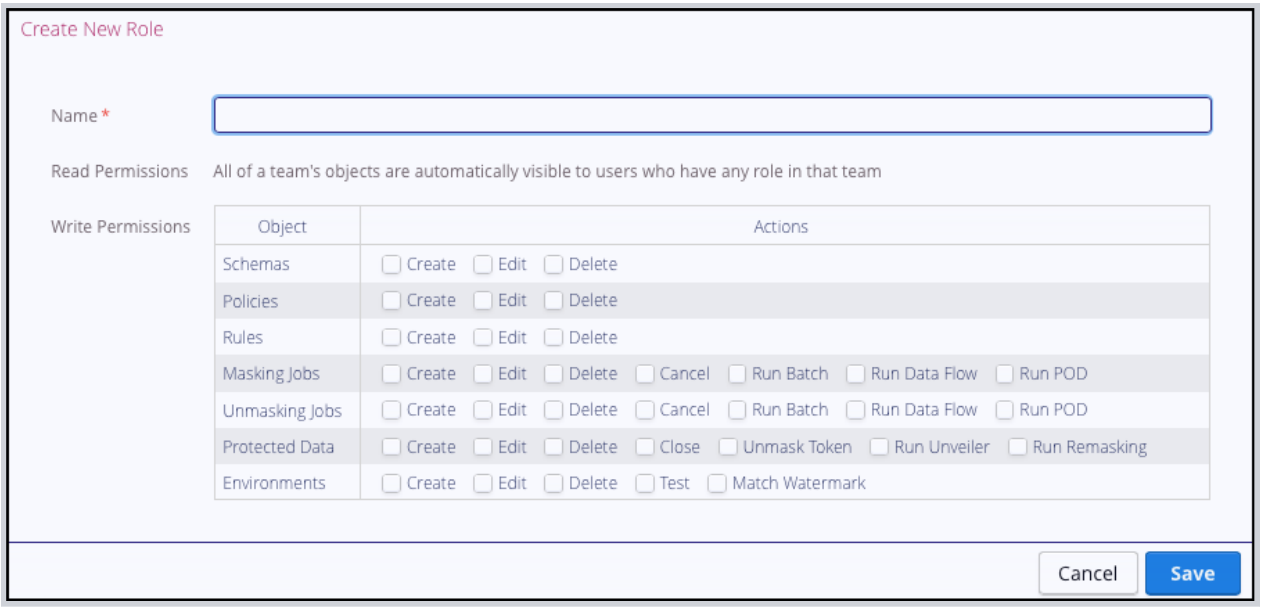
For a description of the default Roles that are supplied in Privitar, see Default Roles.

16.4.1. Creating or Editing a Role

To create or edit a Role:

1. Select **Roles** from the Superuser navigation sidebar.
2. Click on **Create New Role** to add a new Role (or click on the name of an existing role in the **Name** column to edit a Role).

   The **Role** dialog box is displayed showing all the permissions that are available (or have been assigned). For a new role:



Read Permissions: All of a team's objects are automatically visible to users who have any role in that team

Write Permissions:

Object	Actions
Schemas	Create, Edit, Delete
Policies	Create, Edit, Delete
Rules	Create, Edit, Delete
Masking Jobs	Create, Edit, Delete, Cancel, Run Batch, Run Data Flow, Run POD
Unmasking Jobs	Create, Edit, Delete, Cancel, Run Batch, Run Data Flow, Run POD
Protected Data	Create, Edit, Delete, Close, Unmask Token, Run Unveiler, Run Remasking
Environments	Create, Edit, Delete, Test, Match Watermark

3. Assign a unique name to identify the new Role.
4. Check (or uncheck) the specific permissions that you would like to attribute to this Role. For more information about the permissions, see Role Permissions.

16.4.2. Assigning Roles to Users

To assign Roles to Users in a given Team, see Managing Teams.

16.4.3. Deleting a Role

To delete a Role:

1. Select **Roles** from the SuperUser navigation sidebar.
2. Click on the Selector button alongside the name of the Role you wish to delete.
3. Select **Delete** from the **Actions** list box.
4. Click on **OK** to confirm the deletion of the Role.

Note

This action is irreversible and Users that previously had this Role and the associated Permissions will cease to have these permissions after the Role has been deleted.

16.5. Default Roles

The following table describes the default Roles that are provided in Privitar.

Role	Allowed Activities (in the given Team)
Admin	Create, test, edit and delete Environments (including supplying Hadoop Cluster details and managing token vaults, secret keys, Privitar on Demand settings as well as SecureLink configurations).

Role	Allowed Activities (in the given Team)
Author	Create, edit and delete Schemas, Rules and Policies.
	Create, edit and delete masking and unmasking Jobs (for all of Batch/Hadoop, Data Flow, Privitar on Demand).
	Create, edit, delete and close Protected Data Domains.
Operator	Run and cancel Batch masking Jobs (on an already defined Environment, according to a given privacy Policy and defined Job). For example, run and cancel Jobs on a Hadoop cluster.
	An Operator executes Jobs but cannot create or edit Jobs or Policies, unless they have additional Roles.
	The Operator Role also does not include Permissions to run Data Flow Jobs (these require the Data Flow Operator Role).
	Operators may, however, create Protected Data Domains, as this is an important precursor to running a Job.
Investigator	Re-identify/unmask a single field by entering a tokenized value in the unmasking interface and retrieving the original sensitive value. For security this activity requires an additional authentication by another distinct User with the Investigator role. Unmasking is only available on Hadoop Clusters.z
	Investigate a Watermark (through a Batch Job on a Hadoop Cluster).
Unmasker	The Unmasker role is required on User credentials to run and cancel Unmasking Jobs on an entire file, rather than an individual value. Unmasking is only available on Hadoop Clusters (through a Batch Job).
Data Flow Operator	The Data Flow Operator has Run Data Flow permissions that are required on user credentials used in a Data Flow pipeline configuration. That is, for Jobs on a data streaming processor such as NiFi, Kafka/Confluent or StreamSets.
	Note Only API users can operate Data Flow pipelines using either basic HTTP authentication or Mutual TLS authentication.

16.6. Role Permissions

Superusers can create custom Roles for Privitar Users. A custom Role consists of a custom combination of permission. These are assigned on a per-Team basis. This means that the permissions are only valid for the specific Teams a given User has this Role for. For more information on how to create and edit Roles, refer to Managing Roles.

The following table describes what type of permissions to perform certain actions on specific Privitar objects can be assigned to a given Role.

Object	Action	Description
Schemas	Create	Create new Schemas describing the input data.

Object	Action	Description
	Edit	Edit (that is, change) existing Schemas describing the input data. Note that editing a Schema could invalidate existing Policies and Jobs using it.
	Delete	Delete existing Schemas describing the input data. Note that deleting a Schema will invalidate existing Policies and Jobs using it.
Policies	Create	Create new Policies. That is, define which de-identification tranformations (Rules and Generalization strategies) to apply to a given Schema.
	Edit	Edit (that is, change) existing de-identification Policies. Note that editing a Policy could affect and invalidate existing Jobs and Protected Data Domains using it.
	Delete	Delete existing de-identification Policies. Note that deleting a Policy could affect and invalidate existing Jobs and Protected Data Domains using it.
Rules	Create	Create new Rules. That is, define specific de-identification tranformations (masking approaches) that can be used in Policies. Note that Rules, unlike other Privitar objects, are shared with and can be used by any Team, even if the User who created it does not belong to these.
	Edit	Edit existing Rules created by the given User's Team. Note that editing a Rule could affect and invalidate existing Policies, Jobs and Protected Data Domains that use it, including in other Teams.
	Delete	Delete existing Rules created by the given User's Team. Note that deleting a Rule will affect and invalidate existing Policies, Jobs and Protected Data Domains that use it, including in other Teams.
Masking Jobs	Create	Create new masking (that is, de-identification), Jobs of any type (Batch/Hadoop, Data Flow and Privitar on Demand Jobs) on an already defined Environment, according to a given privacy Policy. Note that running the defined masking Job requires additional permissions
	Edit	Edit existing masking Jobs of any type (Batch/Hadoop, Data Flow and Privitar on Demand Jobs). Note that editing a Job could invalidate or change its behavior for new data that gets processed by it.
	Delete	Delete existing masking Jobs of any type (Batch/Hadoop, Data Flow and Privitar on Demand Jobs). Note that deleting an existing Data Flow or Privitar on Demand Job will break the corresponding data pipeline.
	Cancel	Cancel (that is, terminate early) Batch/Hadoop masking Jobs that are still in progress of processing.
	Run Batch	Run (that is, start the processing of) defined Batch masking Jobs(i.e. de-identification Jobs on an Hadoop cluster).

Object	Action	Description
	Run Data Flow	Permit the execution of Data Flow masking Jobs. That is, Jobs on a data streaming processor such as NiFi, Kafka/ Confluent or StreamSets). This permission grants Data Flow permissions that are required on user credentials used in a Data Flow pipeline configuration using either basic HTTP authentication or Mutual TLS authentication.
	Run POD	Permit the execution of Privitar On Demand (POD) Masking Jobs. POD Masking jobs can only be run by API users using Mutual TLS authentication, so this permission is only applicable for API users.
Unmasking Jobs	Create	Create new unmasking Jobs for previously de-identified data files on an Environment. Running the Unmasking Job requires additional permissions.
	Edit	Edit existing Unmasking Jobs. Note that editing a Job could invalidate or change its behavior for new data that gets processed by it.
	Delete	Delete existing Unmasking Jobs.
	Cancel	Cancel (that is, terminate early), Unmasking Jobs that are still in progress of processing.
	Run Batch	Run (that is, start the processing of), defined unmasking (Batch) Jobs (that is, re-identification Jobs on an Hadoop cluster) on a given file. Note that there is a separate permission for unmasking single tokens rather than entire files (see below in **Protected Data**)
	Run Data Flow	Permit the execution of Unmasking Jobs on a data streaming processor such as Apache NiFi, Apache Kafka/ Confluent, or StreamSets.
	Run POD	Permit the execution of Privitar On Demand (POD) Unmasking Jobs. POD Unmasking jobs can only be run by API Users using Mutual TLS authentication, so this permission is only applicable for API Users.
Protected Data	Create	Create new Protected Data Domains (PDDs) in an existing Environment.
	Edit	Edit existing Protected Data Domains (PDDs). Note that during this process previously created PDD Metadata might be overwritten.
	Delete	Delete closed Protected Data Domains (PDDs). Note that this will delete the PDD metadata and Job history associated with it, but not the output data (except the tokens in the Token Vault).
	Close	Close (that is, lock), Protected Data Domains (PDDs). Closing a PDD will make it 'read-only' and prevent any new data from being added. The record of tokens (that is, the Token Vault) produced during Job runs in this PDD is discarded. Closing a PDD also prevents unmasking.
	Unmask Token	Unmask (that is, re-identify to the original/raw value), a single token value of a de-identified field. This operation is only possible on a Hadoop environment. Note that there are separate permissions for unmasking entire files (see Unmasking Jobs above).

Object	Action	Description
	Run Unvelier	Permit the execution of SecureLink Unveiler operations over Protected Data Domains (PDDs). Unveiler operations can only be run by API users using Mutual TLS authentication, so this permission is only applicable for API Users.
	Run Remasking	Permit the execution of ReMasking operations over Protected Data Domains (PDDs). Remasking operations can only be run by API Users using Mutual TLS authentication, so this permission is only applicable for API Users.
	Remove Token Mapping	Permit the removal of a token mapping in a Token Vault in order to support a *Right to be Removed* request. This operation can only be run by API Users using Mutual TLS authentication, so this permission is only applicable for API Users.
Environments	Create	Create new Environments, including supplying Hadoop Cluster details and managing token vaults, secret keys, Privitar on Demand settings as well as SecureLink configurations.
	Edit	Edit configurations of existing Environments, incl. Hadoop Cluster details and managing token vaults, secret keys and encryption, Privitar on Demand settings as well as SecureLink configurations. Note that changes could affect and invalidate other Privitar objects such as Jobs and Protected Data Domains, that use the given Environment.
	Delete	Delete Environments. Note that deleting an Environment will affect and invalidate other Privitar objects such as Jobs and Protected Data Domains, that use that Environment.
	Test	Test the Environment configuration. This test only applies to Hadoop Cluster Environments.
	Match Watermark	Investigate a file (in an Environment owned by your Team) and match a Watermark against it. This operation is only available through Batch Jobs on Hadoop Clusters.

16.7. Managing Users

Users in Privitar are defined as *human* users that operate on Privitar through the user interface, rather than through automation APIs. For information about API Users, see Managing API Users

The Privitar Platform can be configured to use a local user database, LDAP/Active Directory or SAML SSO for identity management of human UI Users.

This section describes managing users when Privitar is configured to use a local User database.

Note

For more information about managing users when Privitar is configured to use LDAP/Active Directory or SSO, contact your system administrator.

16.7.1. Creating/Editing Users

To create a new User or edit the details for an existing User:

1. Select **Users** from the Superuser navigation sidebar.
2. Click on **Create New User**.

   The **Create New User** dialog box is displayed.
3. Enter the details for the new User:
   - **Name** is the full display name of the User (first and last name).
   - **Username** is the unique name that will be used to login to Privitar.
   - **Email** is the email address. (This is optional.)
4. Select the **Account enabled** checkbox to ensure that the account is activated on Privitar.
5. Select the **Superuser** checkbox if you want the new User to have Superuser permissions.
6. Note down or copy the automatically generated Password displayed in the **Password** edit box. This temporary password will be required for the User's first login.

   You can also enter a new password, or click on **Generate** to create a new password.
7. Click on **Save** to create the new User.

The new User will need to be assigned to at least one Team (or have Superuser permissions) to be able to login to Privitar. For more information about how to add Users to a Team and assign Permissions/Roles to them, see Managing Teams.

Each new User will be asked to create a new password after their first login.

16.7.2. Disabling a User account

To disable a User account:

1. Select **Users** from the Superuser navigation sidebar.
2. Click on **Edit** in the **Actions** column alongside the name of the User.
3. Deselect the **Account enabled** checkbox.
4. Click on **Save**.

16.7.3. Resetting a User's password

To reset a User's password:

1. Select **Users** from the Superuser navigation sidebar.
2. Click on **Edit** in the **Actions** column alongside the name of the User.

3. Click on **Change Password**.

   The **Password** edit box is displayed.
4. Enter a new password, or click **Generate** to generate a random password.

   Note down or copy the Password. This temporary password will be required for the User's login.
5. Click on **Save**.

Following the reset of a password, the User will be asked to choose a new password the next time that they login.

16.7.4. Assigning or Removing SuperUsers

To assign Superuser permissions to a User account:

1. Select **Users** from the Superuser navigation sidebar.
2. Click on **Edit** in the **Actions** column alongside the name of the User.
3. Select the **Superuser** checkbox if you want the User to have Superuser permissions. Deselect the checkbox to remove Superuser permissions.
4. Click on **Save**.

16.8. Managing API Users

API Users are defined as application users that operate on Privitar through the automation APIs, rather than the UI. In addition to being used when calling Privitar Automation v3 APIs, API users are also required for:

- Running Data Flow Jobs, using basic HTTP authentication or Mutual TLS authentication.
- Making authorisation checks on HTTPS calls to the Privitar On Demand and SecureLink Unveiler APIs to perform Masking, Re-masking and Unveiling operations on Protected Data Domains (PDDs).

Privitar can be configured to use the Privitar local database or LDAP/Active Directory for identity management of API users.

This section describes managing API users when Privitar is configured to use a local user database.

Note

For more information about managing users when Privitar is configured to use LDAP/Active Directory or SSO, contact your system administrator.

For more information about managing Users, see Managing Users .

Note

API Users cannot login to the Privitar UI.

16.8.1. Creating/Editing API Users

To create a new API user, or edit the details of an existing API user:

1. Select **API Users** from the Superuser navigation sidebar.
2. Click on **Create New API User**.

   The **Create New API User** dialog box is displayed.
3. Enter the details for the new API User:
   - **Name** is the full display name of the API User (first and last name).
   - **Username** is the unique name that will be used to login to Privitar.
   - **Email** is the email address. (This is optional.)
   - **Common Name** is used for API authentication. Enter the details if Privitar is configured to use Mutual TLS.
4. Select the **Account enabled** checkbox to ensure that the account is activated on Privitar.
5. Select the **Superuser** checkbox if you want the new API User to have Superuser permissions.
6. Note down or copy the automatically generated Password displayed in the **Password** edit box. This temporary password will be required for the API User's first login.

   You can also enter a new password, or click on **Generate** to create a new password.

   > **Note**
   >
   > If you have entered a **Common Name**, you do not need to enter **Password** details. **Password** details are only required if you are using basic HTTP authentication.
7. Click on **Save** to create the new API User.

For more information about how to add API Users to a Team and assign Permissions/Roles to them, see Managing Teams.

16.8.2. Disabling an API User account

To disable an API User account:

1. Select **API Users** from the Superuser navigation sidebar.
2. Click on **Edit** in the **Actions** column alongside the name of the API User.
3. Deselect the **Account enabled** checkbox.
4. Click on **Save**.

16.8.3. Resetting an API User's password

To reset a User's password (if basic HTTP authentication is used):

1. Select **API Users** from the Superuser navigation sidebar.
2. Click on **Edit** in the **Actions** column alongside the name of the API User.

3. Click on **Change Password**.

   The **Password** edit box is displayed.
4. Enter a new password, or click **Generate** to generate a random password.

   Note down or copy the password. This temporary password will be required for the API User's login.
5. Click on **Save**.

16.8.4. Assigning or Removing Superusers

To assign Superuser permissions to an API user account:

1. Select **API Users** from the Superuser navigation sidebar.
2. Click on **Edit** in the **Actions** column alongside the name of the API User.
3. Select the **Superuser** checkbox if you want the User to have Superuser permissions. Deselect the checkbox to remove Superuser permissions.
4. Click on **Save**.

16.9. Managing permissions for POD and SecureLink Jobs

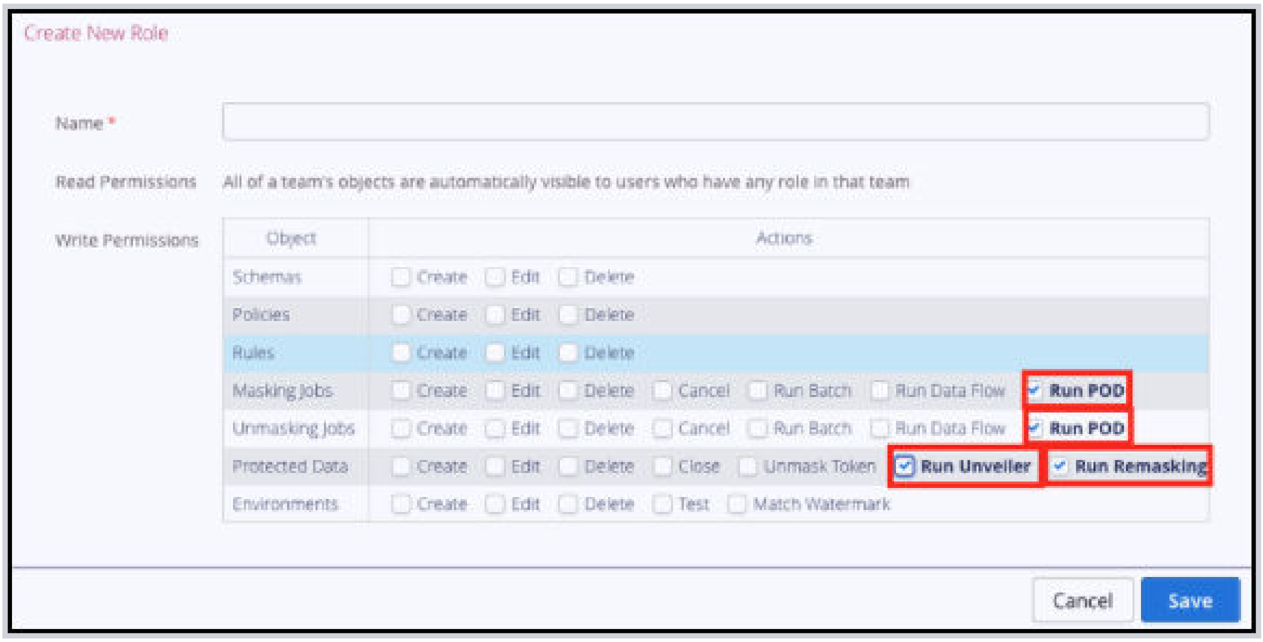
In order to run POD or SecureLink Unveiler jobs, an API user needs to be setup with mutual TLS authentication and be granted the relevant permissions to perform one or more of the operations that are provided for running POD and SecureLink. This includes:

- Masking
- Unmasking
- Remasking
- Unveiling

These permissions are managed on a per-Team basis. For example, if an API user is granted a role enabling them to run Masking POD jobs in a team (Team A), that user will be allowed to run any masking POD job in that team. (This also applies to Unmasking and Unveiling).

This means that in order to run any POD Masking Job which belongs to Team A, the API user needs to have been assigned a role in Team A containing the Run POD permission for Masking Jobs.

See the permissions highlighted in the **Create New Role** dialog box:



Create New Role

Name *

Read Permissions: All of a team's objects are automatically visible to users who have any role in that team

Write Permissions:

Object	Actions
Schemas	☐ Create ☐ Edit ☐ Delete
Policies	☐ Create ☐ Edit ☐ Delete
Rules	☐ Create ☐ Edit ☐ Delete
Masking Jobs	☐ Create ☐ Edit ☐ Delete ☐ Cancel ☐ Run Batch ☐ Run Data Flow ✓ **Run POD**
Unmasking Jobs	☐ Create ☐ Edit ☐ Delete ☐ Cancel ☐ Run Batch ☐ Run Data Flow ✓ **Run POD**
Protected Data	☐ Create ☐ Edit ☐ Delete ☐ Close ☐ Unmask Token ✓ **Run Unveiler** ✓ **Run Remasking**
Environments	☐ Create ☐ Edit ☐ Delete ☐ Test ☐ Match Watermark

To run Remasking POD jobs within a single team, the following permissions need to have been assigned:

- Run Remasking permission for Protected Data
- Run POD permission for Masking Jobs

To run Remasking POD jobs between two teams (Team A and Team B) the following permissions need to have been assigned:

- Run Remasking permission for Protected Data, for the team that owns the source POD Job (Team A).
- Run POD permission for Masking Jobs for the team that owns the destination POD Job (Team B).

For more information about Permissions for users of Privitar, see Role Permissions.

16.10. Managing General Settings

The Settings interface allows a Superuser to control various internal values and settings that the Privitar platform uses during operation.

To access the **Settings** page, click **Settings** on the Superuser Navigation sidebar.

To change a Setting:

1. Select **Settings** from the Superuser navigation sidebar.
2. Click on **Edit** in the **Actions** column alongside the name of the Setting.

   The **Setting** dialog box is displayed.
3. Edit the details for the Setting.
4. Click on **Save** to update the details for the Setting.

16.10.1. Settings description

The following table describes each of the Settings shown in the Settings page.

Setting	Description
CSV Delimiter	Default CSV delimiter character for new Schemas. May be overridden in individual Schema configurations. Unicode characters can be used. For example, `\u0009` for tabs.
CSV Escape Char	Default CSV escape character for new Schemas. May be overridden in individual Schema configurations. Unicode characters can be used. For example, `\u0002` for STX.
CSV Quote Char	Default CSV quote character for new Schemas. May be overridden in individual Schema configurations. Unicode characters can be used. For example, `\u0003` for ETX.
Default CSV Date Format	Default CSV Date Format for defining Date type columns in Schemas. May be overridden in individual Schema configurations. For more information about the date format, see Date and Timestamp formats.
Default CSV Timestamp Format	Default CSV Timestamp Format for defining Timestamp type columns in Schemas. May be overridden in individual Schema configurations. For more information about the timestamp format, see Date and Timestamp formats.
Default Job Output Type	Default HDFS Batch Job output type. The available options are: • `OVERWRITE` : If any existing files are encountered when running the Batch Job, the files will be overwritten. (This is the default option used for this setting.) • `STOP`: If any existing files are encountered when running the Batch Job, the files will **not** be overwritten. Instead, the Job Fails completely and an error message will be issued. This setting can be overridden on a per-Job basis by editing the **Overwriting Behavior** setting for an individual HDFS Batch Job.
Default NULL Value	Default NULL text for new Schemas. May be overridden in individual Schema configurations.
Event Broker Location	The location (URL) of the Event Broker. This location should be specified by a system administrator.

Setting	Description
Hive default job output type	This setting determines the default behavior for Hive table creation when running Hive Batch Jobs. The available options are: • `INSERT` : create the table if it does not exist. • `FAIL` : Attempt to create the table, and fail if it already exists. • `TRUNCATE` : Truncate (erase) the table before inserting into it, or create it if it does not exist. This setting can be overridden on a per-Job basis by editing an individual Hive Batch Job.
Instance ID	This ID is used internally by Privitar when creating names for various objects (such as filenames and YARN queues) to disambiguate these objects from similar objects created by other instances of Privitar that may be sharing the same cluster.
Processor Root	The location of the Privitar installation on the local filesystem. This location should be specified by a system administrator.
Spark Jar File	The location of the Spark Jar installation on the local filesystem. This location should be specified by a system administrator.
Watermark Digits Length	The number of digits used when generating Watermark signatures. The default value is 3. It is not recommended to change this setting without consulting Privitar.

16.11. What are Metadata Attributes

In the Privitar Platform, *metadata attributes* define how existing objects like PDDs can be augmented with additional fields. A metadata attribute is created to describe a value that can be added. Metadata attributes can be used to map the fields of a third-party application, like a data catalog, onto a Privitar object.

16.11.1. Metadata Attribute Mapping

Certain metadata attributes have special meaning to Privitar. To express this, when creating a metadata attribute in the Privitar user interface, it is possible to specify a *mapping* from the metadata attribute onto the special fields that Privitar provides. They can be used to map to metadata read from third-party data catalog applications or can also be used to map to other Privitar objects that have metadata associated with them.

The following list shows the available mapping options. At the moment, the only object in Privitar to support this feature is a PDD.

Mapping	Description	Type
Description	The metadata attribute contains the Privitar object description.	String
Expiration date	The metadata attribute contains the Privitar Object Expiration Date.	Date

When the mapping is enabled, then its type replaces the current type of the metadata attribute.

The mappings are *unique*.

The mapped metadata connects metadata attribute values to the Privitar platform. Once such an association is set, additional actions become allowed (for example, see Managing PDD Lifecycle).

The unmapped metadata contains arbitrary data chosen by the organization to represent data usage information on PDDs. Examples are *Approver*, *Restrictions*, etc.

16.12. Managing Metadata Attributes

With the Privitar Platform, users can change a set of metadata attributes for some Privitar objects. Only PDDs can benefit from this feature at the moment. Metadata attributes let an organization's policies for descriptive information on data use to be represented in Privitar. For example, records of *Approver*, *Restrictions*, and *Expiry Date* may be useful for an organization when creating and working with PDDs, and assessing the risks in their use.

Following the configuration of the set of metadata attributes, Privitar users can supply metadata values when creating a new Privitar object via API or the Privitar user interface. The metadata attributes are then displayed on the object page, obtainable via API, and are also shown when watermarks are extracted.

The attributes themselves have a *data type*, which determines the format of data that the attribute contains, such as Integer or Date, for example.

Note

Metadata attributes are unique. An attempt to create a duplicate will lead to an error message.

16.12.1. Creating or Editing Metadata Attributes

1. Select **Metadata Attributes** on the navigation bar
2. Click on **Create New Metadata Attribute**
3. On the **New Metadata Attribute** screen:
    a. On the top section of the screen, enter the **name** of the attribute you would like to create. You can also decide to untick **Enable Metadata Field**, so the metadata is not available, and therefore not displayed in the metadata list of Privitar objects.
    b. On the **Properties** section, you can enter a **Display Name**. This can be useful when the default name comes from an external system and is not meaningful enough to you. You can add a **description** to clarify the use of the metadata attribute. By default, all metadata are created with the **Value Type** String. Use the list box to change it to a more accurate type.

Note

If the type *Date* is selected, then you must enter a **date format**.

Example of date formats:

Format	Example
yyyy-MM-dd	2021-04-25
yyyy-MM-dd HH:mm:ss	2021-03-02 13:05:20

For the syntax of the date format, please see Date and Timestamp formats.

If the metadata attribute is going to be overwritten by a third-party application, then untick **Values are editable via the Privitar Platform UI**.

c. Select **Enable Mappings** if the metadata attribute should have special meaning to Privitar. You need to specify the **Entity** (currently only **Protected Data Domain** is available), which is the Privitar item that will use the metadata attribute, and the **mapping**. See Metadata attribute mappings for more information.

4. Click on **Save**. Now the metadata attribute will be provided every time a new object of the type defined in the **Entity**field is created.

16.12.2. Default Metadata Attributes

The following table describes the metadata fields that are enabled for any PDD created with the Privitar platform.

Field	Description
Description	A general purpose overview of the Privitar object. *This metadata attribute is enabled by default.*
Approver	The person who has approved the creation of this Privitar object. *This metadata attribute is enabled by default.*
Recipient	The intended recipient of the Privitar object. *This metadata attribute is enabled by default.*
Intended Use	A description of the intended use of the Privitar object.
Restrictions	A description of any restrictions that might apply to the Privitar object. For example, access may be subject to contractual or information security limitations.

Field	Description
Expiry Date	The date on which the Privitar object should cease to be used. Using the data after this date may constitute a risk to the organization. **Note** This date is for information only and has no automatic impact on the physical access to data or the PDD's open/closed state in Privitar.
Notes	Any additional notes about the Privitar object.

16.12.3. Data Types for Metadata Attributes

The following table lists available data types for custom metadata fields.

Type	Limits
String	A UTF-8 characters, max length 2000 bytes
Integer	A Signed 64-bit integer, between -2^{63} and a maximum value of 2^{63}-1
Float	A 64-bit precision floating point number
Boolean	A value is either *true* or *false*.
Date	The format of the date is specified using the *format string* entered on the Metadata Attribute Definition. This format is used for internal storage, and when the date metadata value is received via API. The time could also be specified along with the date.

16.13. Administering in a Hadoop Cluster

This section contains important details about administering the Privitar platform in a Hadoop cluster.

16.13.1. Supported Character Encodings

Privitar requires all input data to be in UTF-8 format. This is due to the current level of support in Spark.

16.13.2. Sensitive Information Leakage After YARN Abnormal Termination

As Spark jobs for Privitar are running in the cluster they write out temporary files to the directory(s) set by the YARN `yarn.nodemanager.local-dirs` parameter. Due to the nature of Privitar, these files may include sensitive information.

In the event that YARN terminates abnormally, such files may remain in YARN's working directory(s) and not be automatically cleaned-up.

The files must therefore be manually removed in this situation. To do this, delete cache files for Privitar jobs that failed to the YARN termination. The cache files are located under the sub-directory of the directory:

```
${yarn.nodemanager.local-dirs}/usercache/${username}/appcache/$
{applicationId}
```

This directory corresponds to the user that ran the job.

16.13.3. Configuring HDFS-level compression

HDFS-level compression is transparent to Privitar and requires only cluster-level configuration.

To enable HDFS-level compression, see the Cloudera HDFS Administration Guide.

16.13.4. Compression inside Avro files

To enable a compression codec inside Avro files, modify the `spark-overrides.properties` file in:

```
<processor-root>/config
```

as follows:

```
spark.sql.avro.compression.codec = uncompressed|snappy|deflate
spark.sql.avro.deflate.level = 5 // if using deflate
```

16.13.5. Internal Spark compression

Spark can use compression internally (for RDD partitions, broadcast variables and shuffle output). This is transparent to Privitar and requires only cluster-level configuration.

For more information about configuring compression in Spark, see the Apache Spark Configuration Guide .

16.14. Creating Hadoop Cluster Types

A number of different Hadoop Cluster types can be setup in Privitar so that a user can define Privitar Environments that use different types of Hadoop clusters.

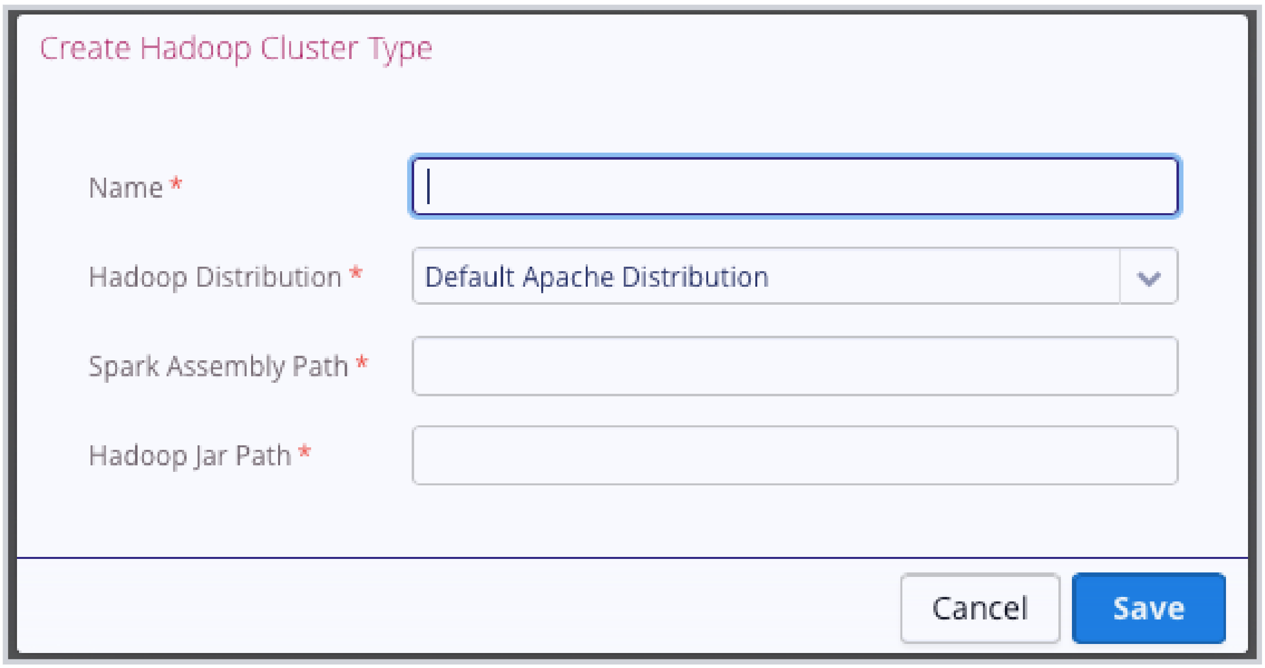
New Hadoop cluster types are created from the Superuser Cluster Types index page. For a new cluster type you need to specify the paths to the required jars for the Hadoop Cluster type.

To create a new cluster type:

1. Select **Cluster Types** from the Superuser navigation sidebar.

   The **Cluster Types** page is displayed.
2. Click on **Create new Hadoop Cluster Type**.

   The **Create Hadoop Cluster Type** dialog box is displayed:


Create Hadoop Cluster Type

Name *

Hadoop Distribution * Default Apache Distribution

Spark Assembly Path *

Hadoop Jar Path *

3. Enter a name for the Cluster in the **Name** box.
4. Choose the Hadoop Distribution in use from the **Hadoop Distribution** list box.
5. Enter the Spark assembly path in the **Spark Assembly Path** box.

   This path points to the location of the Spark installation on the local filesystem. (This should be specified by a system administrator.)
6. Enter the Hadoop Jar Path in the **Hadoop Jar Path** box.

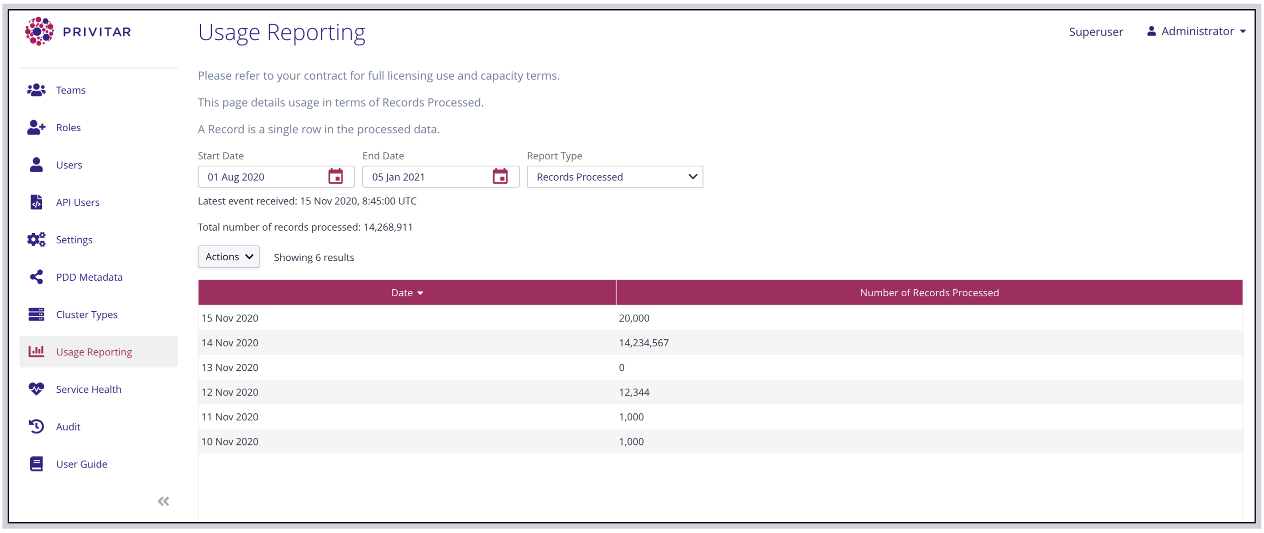
   This path points to the location of the Hadoop client libraries path on the local Privitar server. The value depends on the vendor and version of the Hadoop distribution. If Privitar is not installed on an edge node, the Hadoop jars have to be copied from the cluster to a local path.
7. Click on **Save** to save the new configuration.

   The **Cluster Types** page will be updated to show the new cluster.

16.15. Managing Service Usage

The **Usage Reporting** page tracks and logs information about daily usage of the Privitar platform. By default, the table displays usage data per day for the past month.

The page is available to Superusers in the Superuser Administration section of Privitar. For example:



Please refer to your contract for full licensing use and capacity terms.

This page details usage in terms of Records Processed.

A Record is a single row in the processed data.

Start Date: 01 Aug 2020
End Date: 05 Jan 2021
Report Type: Records Processed

Latest event received: 15 Nov 2020, 8:45:00 UTC

Total number of records processed: 14,268,911

Showing 6 results

Date	Number of Records Processed
15 Nov 2020	20,000
14 Nov 2020	14,234,567
13 Nov 2020	0
12 Nov 2020	12,344
11 Nov 2020	1,000
10 Nov 2020	1,000

Privitar usage can be quantified using two different metrics:

- As an aggregate number of Privacy Preserving Operations (PPOs) per day. A PPO is defined as any rule that has been applied to a field, but excludes:
  - A field in a column with a **Drop Column** rule applied.
  - A field in a column with a **Retain Value** rule applied

  Both of these rules are passing through the data and not performing any sort of masking operation, so they are not considered as PPO operations.
- As an aggregate number of records processed. A record is a single row in the processed dataset.

The **Report Type** list box determines the type of service usage that is displayed.

A single entry in the usage summary table corresponds to an aggregate view of either the (total) number of PPOs used or the (total) number of records processed, across all Teams that were set up on Privitar for that day of usage.

The page also contains information about:

- The **Latest event received**. That is, the latest event received that details the daily aggregate count of PPOs or records processed.
- The **Total number of PPOs** or the **Total number of records processed** for the current time period specified in **Start Date** and **End Date**.

A number of actions are available from this page that enable you to customise the view of the usage data:

- Click on **Date** in the usage table to sort the usage records in ascending order (oldest first) or descending (latest first) order.
- Click on the second column (**Number of Records Processed** or **Number of PPOs**) to sort the entries in ascending order (lowest number of PPOs or records first) or descending order (highest number of PPOs or records first)
- Use the **Start Date** and **End Date** date pickers to view the usage data for a specific time period. The table will be updated to show the usage data over the selected period.

The **Total number of PPOs** or **Total number of records** summary information is also updated to reflect the new time period.

For futher analysis of the usage data, you can also download a usage report for either the number of PPOs or the number of records processed. To download a CSV file of the current usage data displayed on the page, select **Actions > Download**. The downloaded file name and the column headings in the downloaded file will reflect the report type selected.

No records on this page can be edited or deleted.

16.16. Managing Service Health

The Service Health page displays status information about the general health of any services that are connected to the Privitar platform. The services that can be monitored are:

- NiFi
- Kafka
- StreamSets
- Privitar-On-Demand (POD)
- Privitar SDK

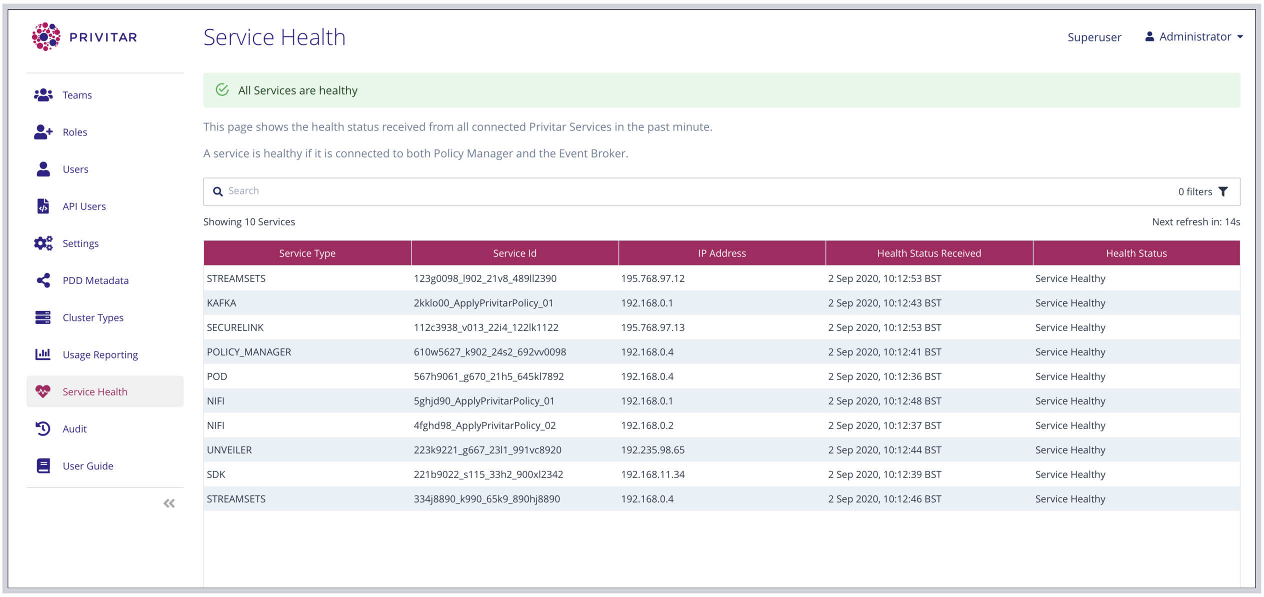
The status information that is shown refers to whether or not the service is able to connect to the **Privitar Event Broker** which is used for logging the usage of the system. (For more information about the Event Broker, see Managing Service Usage.)

The information displayed for each service is:

- Service Type - The service type.
- Service Id - The Id of the service.
- IP Address - The IP address of the service.
- Health Status Received - The time and date that the health status of the service was last received.
- Health Status - The status of the connection between the service and the Event Broker. This can be any of: **Service Healthy**, **Service Unavailable**, **Unable to Connect to Event Broker**.

The status information for each service is updated every 30 seconds.

The page is available to Superusers in the **Superuser Administration** section of Privitar. If all connected services are healthy, a green notification banner is displayed at the top of the page:



Service Health

All Services are healthy

This page shows the health status received from all connected Privitar Services in the past minute.

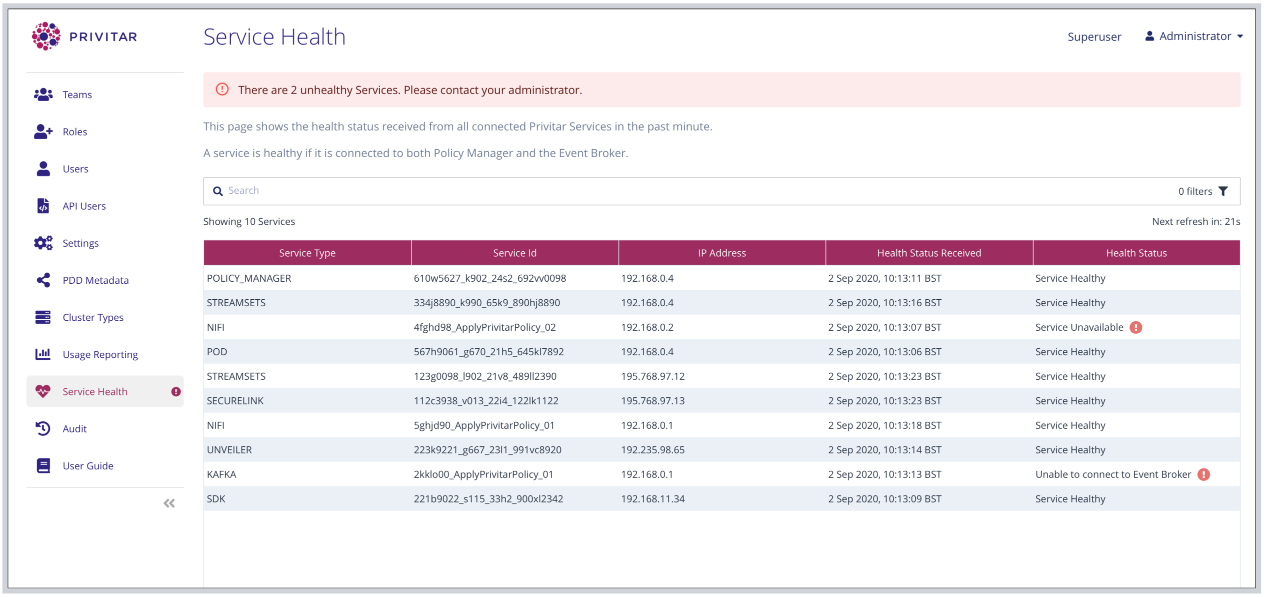
A service is healthy if it is connected to both Policy Manager and the Event Broker.

Showing 10 Services

Next refresh in: 14s

Service Type	Service Id	IP Address	Health Status Received	Health Status
STREAMSETS	123g0098_l902_21v8_489ll2390	195.768.97.12	2 Sep 2020, 10:12:53 BST	Service Healthy
KAFKA	2kklo00_ApplyPrivitarPolicy_01	192.168.0.1	2 Sep 2020, 10:12:43 BST	Service Healthy
SECURELINK	112c3938_v013_22i4_122lk1122	195.768.97.13	2 Sep 2020, 10:12:53 BST	Service Healthy
POLICY_MANAGER	610w5627_k902_24s2_692vv0098	192.168.0.4	2 Sep 2020, 10:12:41 BST	Service Healthy
POD	567h9061_g670_21h5_645kl7892	192.168.0.4	2 Sep 2020, 10:12:36 BST	Service Healthy
NIFI	5ghjd90_ApplyPrivitarPolicy_01	192.168.0.1	2 Sep 2020, 10:12:48 BST	Service Healthy
NIFI	4fghd98_ApplyPrivitarPolicy_02	192.168.0.2	2 Sep 2020, 10:12:37 BST	Service Healthy
UNVEILER	223k9221_g667_23l1_991vc8920	192.235.98.65	2 Sep 2020, 10:12:44 BST	Service Healthy
SDK	221b9022_s115_33h2_900xl2342	192.168.11.34	2 Sep 2020, 10:12:39 BST	Service Healthy
STREAMSETS	334j8890_k990_65k9_890hj8890	192.168.0.4	2 Sep 2020, 10:12:46 BST	Service Healthy

If there is a problem with any of the connected services, the notification banner will turn red and an error icon will be displayed alongside the unhealthy services:



Service Health

There are 2 unhealthy Services. Please contact your administrator.

This page shows the health status received from all connected Privitar Services in the past minute.

A service is healthy if it is connected to both Policy Manager and the Event Broker.

Showing 10 Services

Next refresh in: 21s

Service Type	Service Id	IP Address	Health Status Received	Health Status
POLICY_MANAGER	610w5627_k902_24s2_692vv0098	192.168.0.4	2 Sep 2020, 10:13:11 BST	Service Healthy
STREAMSETS	334j8890_k990_65k9_890hj8890	192.168.0.4	2 Sep 2020, 10:13:16 BST	Service Healthy
NIFI	4fghd98_ApplyPrivitarPolicy_02	192.168.0.2	2 Sep 2020, 10:13:07 BST	Service Unavailable ❗
POD	567h9061_g670_21h5_645kl7892	192.168.0.4	2 Sep 2020, 10:13:06 BST	Service Healthy
STREAMSETS	123g0098_l902_21v8_489ll2390	195.768.97.12	2 Sep 2020, 10:13:23 BST	Service Healthy
SECURELINK	112c3938_v013_22i4_122lk1122	195.768.97.13	2 Sep 2020, 10:13:23 BST	Service Healthy
NIFI	5ghjd90_ApplyPrivitarPolicy_01	192.168.0.1	2 Sep 2020, 10:13:18 BST	Service Healthy
UNVEILER	223k9221_g667_23l1_991vc8920	192.235.98.65	2 Sep 2020, 10:13:14 BST	Service Healthy
KAFKA	2kklo00_ApplyPrivitarPolicy_01	192.168.0.1	2 Sep 2020, 10:13:13 BST	Unable to connect to Event Broker ❗
SDK	221b9022_s115_33h2_900xl2342	192.168.11.34	2 Sep 2020, 10:13:09 BST	Service Healthy

An error icon will also be displayed on the Service Health navigation button so you can see the status of the services at any time that you are using Privitar.

It is possible to filter the entries shown in the table so that you can view just the services or attributes that you are interested in. For example, you might want to view the health of one specfic service, not all the services.

To apply a filter:

1. Click on the filter icon. The **Filter** dialog box is displayed.
2. Edit the dialog box to select the services or attributes that you want to display.
3. Click on **Apply**. The filter is applied to the table and filter buttons are displayed to indicate that a filter is active.

16.17. Managing the Audit log

The Audit log records the changes that have been made to the configuration of the Privitar platform together with specific events that have taken place, such as a Job run completion. In effect, it provides an activity history of the Privitar platform.

The Audit log is available to all users of the Privitar platform.

- If you log in as a *Superuser*, you will see the changes made by all the teams.
- If you connect as a standard user, all teams that you are a member of will be displayed. Further, system administration objects such as teams and users will not be displayed.

A Superusers is able to see logging information for all users and object types and shows the timestamp and initiating user for the following activities:

- User logins and failed logins.
- Creations, deletions, and other changes to the configuration of:
  - Schemas
  - Policies
  - Rules
  - Protected Data Domains
  - Jobs
  - Environments
  - Users
  - Roles
  - Teams
  - Settings
- Job Runs and completions.

Standard users will be able to see a subset of the log information. They will be able to see activities for the following object types:

- Creations, deletions, and other changes to the configuration of:
  - Schemas
  - Policies
  - Rules
  - Protected Data Domains
  - Jobs
  - Environments
- Job Runs and completions.

To view the Audit log:

1. Select **Audit** from the navigation bar.

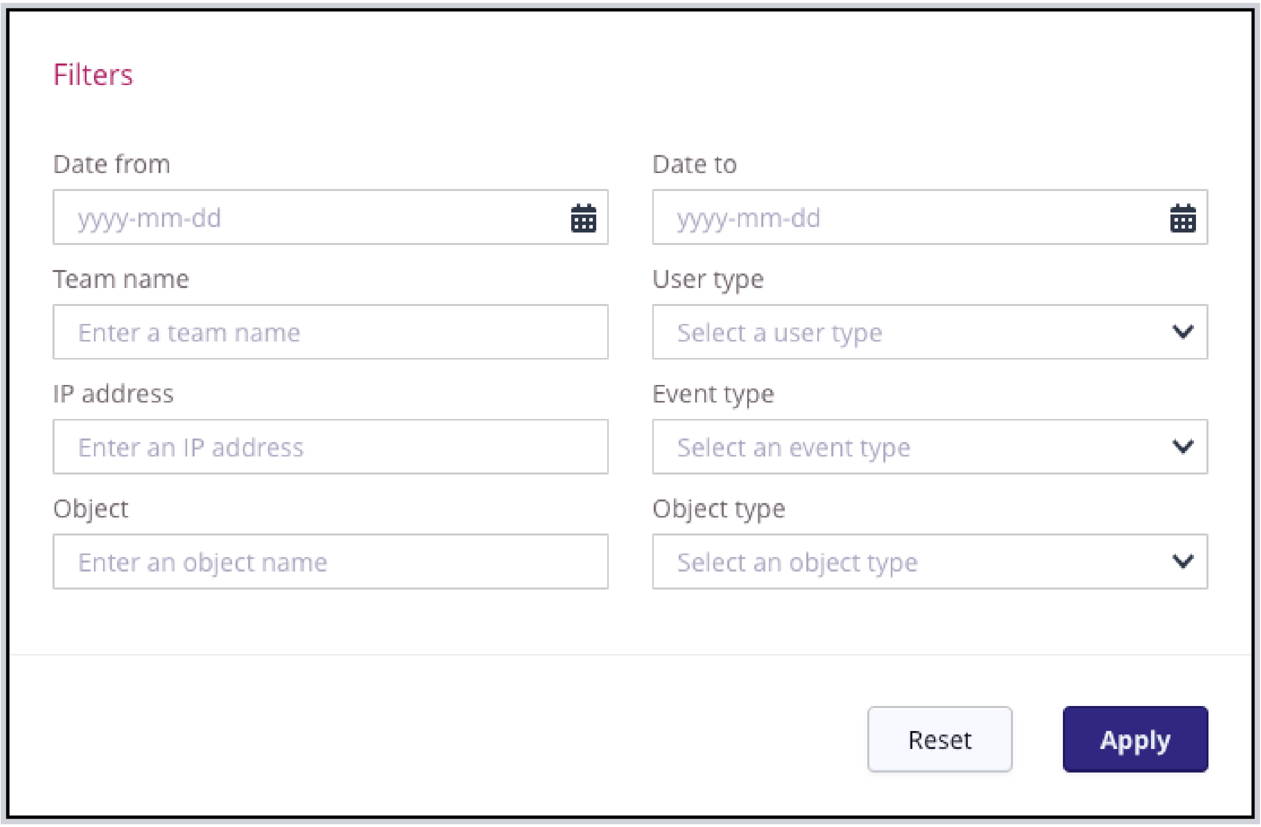
   The Audit page is displayed containing a list of events for changes that have been made on the Privitar platform.

2. Click on an event or change of interest to display more information.
3. Information on the event is displayed in the **Event Details** section.

To select a set of events based on one or more specific values:

1. Choose **Audit** from the navigation bar.
2. Click on the funnel icon on the top right corner of the screen.

   The **Filters** dialog box is displayed:



Filters

Date from: yyyy-mm-dd	Date to: yyyy-mm-dd
Team name: Enter a team name	User type: Select a user type
IP address: Enter an IP address	Event type: Select an event type
Object: Enter an object name	Object type: Select an object type

3. Select the Filters that you wish to apply to the log.

   The **Date From** date field is taken as the beginning of the provided date and **Date To** date field is the end of the provided date.

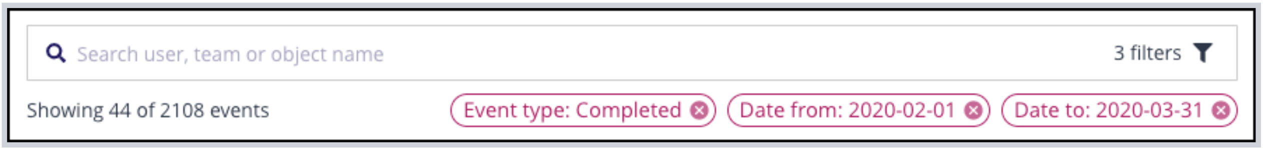
   Note

   If you select **Date From** to 2020-12-20, internally, it will be interpreted as 2020-12-20T00:00:00.000Z.

   If you select **Date To** to 2020-12-20, internally, it will be interpreted as 2020-12-20T23:59:59.999Z.

4. Click **Apply**.

   The audit log will show a subset of the events based on the applied filters. The filters are displayed above the log to indicate that the view is filtered:



To remove any of the filters, click on the **X** alongside the filter.

The audit log will be updated to reflect the change in the filter.

Some objects provide direct access to their audit logs via an **Audit Log** button located in the top-right corner of the page.

The following object types provide an **Audit Log** button:

- Schemas
- Policies
- Rules
- Jobs
- Environments

17. Reference

This section contains detailed reference information about various aspects of the Privitar platform:

- Data and Timestamp formats in CSV files; see Date and Timestamp formats.
- Batch data sources and supported types; Batch data sources and Supported Types.
- Data Types; Privitar Data Types.
- Token Vault Types; Token Vault Types.
- Multi-language support; see Privitar Multi-Language Support.
- Event Types generated by the Instrumentation system; see Event Types.

17.1. Date and Timestamp formats

When processing data containing a date or timestamp field, Privitar uses format conventions based on the formats defined in the Java.time.format class. For example, this convention is used when importing CSV files or outputting data to CSV files.

The Privitar Schema Date format for a Timestamp field of format:

```
24-12-2019 12:11:02.123
```

is:

```
dd-MM-yyyy HH:mm:ss.SSS
```

Note the following upper-case and lower-case distinctions in the formats:

- lower-case `m` is minutes, not months.
- upper-case `M` is month, not minutes.
- lower-case `s` is seconds, not fractions of seconds.
- upper-case `S` is fractions of seconds, not seconds.

Some more common example formats are shown in the table below:

Date / Time	Format	Description
Years	yyyy	Represents a full year, such as 1999.
	yy	Represents an abbreviated year, such as 99.
Months	MMMM	Represents a fully spelled out month name, such as `January`.
	MMM	Represents an abbreviated month name, such as `Jan`.
	MM	Represents a two-digit month (with 0-padding), such as `03` or `11`.
	M	Represents a one-digit or two-digit month (without 0-padding), such as 3 or 11.
Days	dd	Represents a two-digit day of the month (with 0-padding), such as `06` or `24`.

Date / Time	Format	Description
	d	Represents a one-digit or two-digit day of the month (without 0-padding), such as 6 or 24.
Hours	HH	Represents the hour of the day on a 24-hour clock, such as `04` or `10`.
Minutes	mm	Represents the minute of the hour (with 0-padding), such as `05` or `34`.
Seconds	ss	Represents the second of the minute (with 0-padding), such as `07` or `18`.
Fractions of a Second	SSS	Represents milliseconds, such as `003` or `121`.
	SSSSSS	Represents microseconds, such as `000003` or `432121`.
	SSSSSSSSS	Represents nanoseconds, such as `000000003` or `139648121`.

17.2. Batch data sources and Supported Types

This section describes the data file formats that can be processed by Privitar Batch Jobs.

17.2.1. Supported data formats

Privitar Batch Jobs can read files from HDFS in the following formats:

- **Avro**. For more information, see Avro file format support.
- **CSV**. For more information, see CSV file format support.
- **Parquet**. For more information, see Parquet file format support.

17.2.2. Working with Hive tables

The Privitar platform can read and process the data in Hive tables without changing the underlying data format.

When reading Hive tables where the underlying data is in CSV format, Privitar will copy over the same CSV delimiter settings into the output Hive table if a delimiter has been set. Otherwise, the cluster's default delimiter will be used.

For other CSV settings used in Hive tables, Privitar will use the cluster's default settings. An example of the typical CSV settings defined for a cluster is shown in the table below. But, check your local cluster configuration:

Setting	Cluster default
quoteChar	`"'",`
escapeChar	`"\"`

17.2.3. Working With Unsupported Data Types

The Privitar platform does not support all data types. However, it is often necessary to read unsupported data types and either output it un-modified, or remove it.

Privitar Schemas can achieve this by use of a Privitar type called `Other`. This is a placeholder for types that Privitar does not currently support.

Fields declared as `Other` in a Privitar Schema limit Policy options for that field to retaining and removing (dropping):

- If a *Retain Rule* is selected, the field is copied unchanged to the output. Tokenization and Generalization are not supported.
- If a *Drop Rule* is selected, the field will be suppressed entirely in the output.

When outputting data that contains passed-through `Other` fields, the behavior depends on the selected output format:

Output format	Behavior
Avro, Parquet	In the output data, the schema and data values for `Other` fields are written identically in the output as read in the input.
CSV	In the CSV output data, the value for `Other` fields is written in an undefined string format.

17.2.4. Avro file format support

Privitar can read and write data stored in Apache Avro format. (For more information about Avro file format, see https://avro.apache.org/.)

Avro specifies a schema definition syntax, and schemas are included in Avro-encoded data as a header. In order to load Avro data, a Privitar Schema compatible with the Avro schema must be created separately first.

This section describes how to define fields in a Privitar Schema such that it will be suitable for reading data conforming to a known Avro schema. It also details unsupported features of Avro schemas in Privitar.

Supported Avro data consists of a top-level record consisting of named fields, each of which has a typed value. The Avro schema allows the following data types for fields. The table also describes the level of support in Privitar. (For more information about Privitar data types, see Privitar Data Types.):

Avro Type	Description	Support in Privitar
`null`	No value.	Supported as Privitar `Other` type.
`boolean`	Binary value.	Supported as Privitar `Boolean` type.
`int`	int: 32-bit signed integer.	Supported as Privitar `Integer` type.
`long`	long: 64-bit signed integer.	Supported as Privitar `Long` type.
`float`	float: single precision (32-bit) IEEE 754 floating-point number.	Supported as Privitar `Float` type.
`double`	double: double precision (64-bit) IEEE 754 floating-point number.	Supported as Privitar `Double` type.
`bytes`	Sequence of 8-bit unsigned bytes.	Supported as Privitar `Other`.
`string`	Unicode character sequence.	Values of types string are mapped onto `Texttype` in Privitar.
`date`	Number of days from the unix epoch, 1 January 1970.	Supported as Privitar `Date`.

Avro Type	Description	Support in Privitar
`timestamp-millis`	Number of milliseconds from the unix epoch, 1 January 1970 00:00:00.000 UTC.	Supported as Privitar `Timestamp`. `timestamp-millis` are always written as/converted to microseconds (`timestamp-micros`), irrespective of the timestamp format of the input file (and even when using a *Retain Rule*).
`timestamp-micros`	Number of microseconds from the unix epoch, 1 January 1970 00:00:00.000000 UTC.	Supported as Privitar `Timestamp`. NiFi does not support timestamp-micros in 1.6.0, in 1.8.0 and later. It converts microseconds to milliseconds (`timestamp-millis`), therefore losing precision. This is outside of the control of the Privitar platform.
`Record`	Key/value pairs where the keys are drawn from a fixed set, and the values can be any type, for example a list.	Supported as Privitar `Other` type. When outputting to CSV, Records are output as a string of undefined format.
`Enum`	Single value drawn from a fixed set of strings.	`Enum`s are not supported (even when classifying them as Privitar schema type `Other`). When outputting to CSV, `Enum`s are output as a string.
`Array`	A list of values, either primitives or complex types. Values may be nested; for example, a list of lists.	Supported as Privitar `Other` type. When outputting to CSV, Arrays are output as a string of undefined format.
`Map`	Key/value pairs where the keys are arbitrary and the value type is fixed.	Supported as Privitar `Other` type. When outputting to CSV, Records are output as a string of undefined format.
`Union`	Unions are declared with a list of other types, for example int and string. Values must then conform to one of those types.	Supported as Privitar `Other` type, with the exception of the form `[null, primitive-type]` to indicate a nullable primitive value. All fields are nullable in Privitar Schemas, so this Union format is equivalent to a simple field of the correct Privitar type.
`Fixed`	A fixed value of a specified number of bytes.	Supported as Privitar `Other`.

Avro Type	Description	Support in Privitar
`Decimal`	Decimal of precision defined by underlying field.	Supported as Privitar `Other` type. Fields in an Avro schema with logicalType of `decimal` are always converted to type `fixed`, irrespective of the input type (and even when using a *Retain Rule*).

How to Read and Write a Table in Avro format

To read and write a table in Avro format, follow these steps. (For more information about Avro format support in Privitar, see Avro file format support.):

1. Create or infer a Privitar Schema for the Avro schema.

   The Privitar Schema must be created to match the Avro schema. If the schemas do not match, errors will be reported in the **Job Results** page when the Job is run.
2. Create a Policy.
3. Create a Batch Job.
4. On the Batch **Job** dialog box in the **Data Locations** tab, choose `Avro` from the **Type** list box for the relevant table. If required, select the settings icon to specify the **Namespace** and **Record Name** header fields. These are optionally included in Avro data to describe the Avro schema in a higher-level way.
5. Run the job. When the Job is run, the data will be read in and written back out using the Avro format.

Some things to note about how Privitar reads and writes Avro format data:

- The top-level name and namespace for the schema of the output Avro can be specified when a Batch Job is created that outputs to Avro.
- Privitar will retain the Avro schema after processing, except when using the Encrypt Rule and certain Generalization operations that output ranges or default strings since these will naturally require changing the data type (to string).

  When writing Avro files, all columns, including any fields that are originally defined as non-nullable (nullable=false) in the input, are automatically converted to be nullable.
- When reading Avro data, default values that are used for CSV related settings (that is, date formats) are ignored by Privitar.

17.2.5. CSV file format support

This section describes options for reading comma-separated value (CSV) formatted files in Privitar.

The CSV delimiter, Quote character and Escape Character settings can be configured as global default settings by Superuser in the Privitar Settings window. For more information, see Managing General Settings.

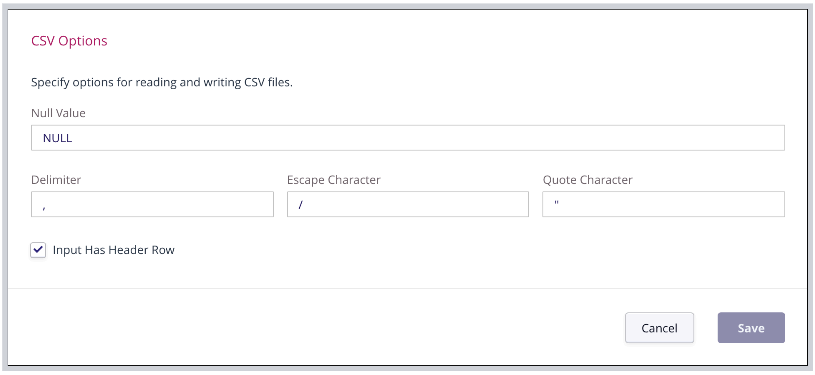
The options are described in the following table:

Setting	Description
Null Value	The input text that will be parsed to a missing (null) value. This value is case-sensitive.
Delimiter	The default character that is used to separate the fields in a row.
Escape Character	The character that starts an escape sequence. For example, you can escape a quote (using `\"` or `\'`) within a quoted string to include these characters literally without being interpreted as a quote. For example, `'It\'s OK'` is interpreted as `It's OK`. If you use an escape character outside of a quoted context, then it is not interpreted as an escape value but instead appears verbatim.
Quote Character	The character that is used to delimit text strings in the input.
CSV Header Options	When reading a CSV as an input file as part of a masking or unmasking job, you can declare whether the first row of the CSV file is a header row. When a header row is expected, make a selection to retain or drop header rows within the output. You control this behavior by choosing one of the following options: **Input files have no header row—**Every row in the input files is treated as a data record. **Retain header row—**The first row in each input file is interpreted as a header, which must match the column names of the input Schema. The headers will be retained in the output files. **Drop header row—**The first row in each input file is interpreted as a header. The headers will be dropped from the output files. Note that these options only apply to masking and unmasking jobs. When reading a CSV file to infer a Schema, the header row is used to name the columns in the Schema. In this case, check **Contains header row**. This is described within Creating a Schema from CSV.

How To Read a Table in CSV Format

To read a table in CSV format:

1. Create a Privitar Schema for the CSV file. (For more information about how to read Dates and Timestamps in CSV files specifically, see Date and Timestamp formats.)
2. Create a Policy.
3. Create a Batch Job.
4. On the Batch **Job** dialog in the **Data Locations** tab, choose **CSV** from the **Type** list box for the relevant table.
5. If required, change the CSV delimiter and quote settings. To do this:
   a. Select the **Data locations** tab from the Batch **Job** dialog box.
   b. Select the *Settings* icon alongside the relevant CSV table. The **CSV Options** dialog box is displayed.


CSV Options

Specify options for reading and writing CSV files.

Null Value: NULL

Delimiter: ,

Escape Character: /

Quote Character: "

[✓] Input Has Header Row

c. Edit the settings in the dialog box to define certain attributes of the CSV file that is being imported. (For more information about the settings, see CSV file format support.)
d. Select **Save** to save the changes.

6. Run the Job. When the job is run, the data will be read using the specified CSV format.

17.2.6. Parquet file format support

Privitar can read and write data stored in Apache Parquet format. (For more information about Parquet file format, see https://parquet.apache.org/.)

This section describes how to define fields in a Privitar Schema such that it will be suitable for reading data conforming to a known Parquet schema. It also details unsupported features of Parquet schemas in Privitar.

In the table below, Parquet types are listed, along with the type that should be selected in the Privitar Schema to ensure compatibility between the data file and the Privitar Policy definition. (For more information about Privitar data types, see Privitar Data Types:

Parquet Type	Description	Support in Privitar
`BOOLEAN`	1-bit Boolean	Supported as Privitar `Boolean` type.
`INT(8,signed)`	8-bit signed integer	Supported as Privitar `Byte` type.
`INT(16,signed)`	16-bit signed integer	Supported as Privitar `Short` type.
`INT32,INT(32,signed)`	32-bit signed integer	Supported as Privitar `Integer` type.
`INT64,INT(64,signed)`	64-bit signed integer	Supported as Privitar `Long` type.
`INT96,INT(96,signed)`	96-bit signed integer	Supported as Privitar `Timestamp` type.
`DOUBLE`	IEEE 64-bit floating point values	Supported as Privitar `Double` type.
`FLOAT`	IEEE 32-bit floating point values	Supported as Privitar `Float` type.
`UINT_8`	8-bit unsigned integer	Supported as Privitar `Other` type.

Parquet Type	Description	Support in Privitar
`UINT_16`	16-bit unsigned integer	
`UINT_32`	32-bit unsigned integer	
`UINT_64`	64-bit unsigned integer	
`DECIMAL`	Decimal of precision defined by underlying field.	Supported as Privitar `Other` type
`STRING`	UTF-8 encoded character string.	Supported as Privitar `Text` type
`DATE`	Number of days from the Unix epoch, 1 January 1970.	Supported as Privitar `Date` type
`TIMESTAMP(MILLIS)`	Number of milliseconds from the Unix epoch, 00:00:00.000 on 1 January 1970, UTC.	Supported as Privitar `Timestamp`. `TIMESTAMP(MILLIS` are always written as/converted to nanoseconds using the `int96` datatype (without any metadata), irrespective of the timestamp format of the input file (and even when using a *Retain Rule*). The default writing format can be overridden by modifying the `spark.sql.parquet.outputTimestampType` property, setting it to `TIMESTAMP_MILLIS`.
`TIMESTAMP(MICROS)`	Number of microseconds from the Unix epoch, 00:00:00.000000 on 1 January 1970, UTC.	Supported as Privitar `Timestamp`. `TIMESTAMP(MICROS)` are always written as/converted to nanoseconds using the `int96` datatype (without any metadata), irrespective of the timestamp format of the input file (and even when using a *Retain Rule*). The default writing format can be overridden by modifying the `spark.sql.parquet.outputTimestampType` property, setting it to `TIMESTAMP_MICROS`.
`TIMESTAMP(NANOS)`	Number of nanoseconds from the Unix epoch, 00:00 on 1 January 1970, UTC.	Supported as Privitar `Other` type.
`TIME(MILLIS)`	Number of milliseconds after midnight.	Supported as Privitar `Other` type.
`JSON`	UTF-8 encoded character string of valid JSON.	Supported as Privitar `Other` type.

Parquet Type	Description	Support in Privitar
`MAP`	A table of key-value pairs.	Supported as Privitar `Other` type.
`LIST`	A sequence of values.	Supported as Privitar `Other` type.
`ENUM`	One of a fixed set of values, for example a day of the week.	Supported as Privitar `Other` type.

How to Read and Write a Table in Parquet format

To read and write a table in Parquet format, follow these steps. (For more information about Parquet format support in Privitar, see Parquet file format support.):

1. Create or infer a Privitar Schema for the Parquet schema.

   The Privitar Schema must be created to match the Parquet schema. If the schemas do not match, errors will be reported in the **Job Results** page when the Job is run.
2. Create a Policy.
3. Create a Batch Job.
4. On the Batch **Job** dialog box in the **Data Locations** tab, choose `Parquet` from the **Type** list box for the relevant table.
5. Run the Job. When the job is run, the data will be read in and written back out using Parquet format.

Some things to note about how Privitar reads and writes Parquet format data:

- Privitar will retain the Parquet schema after processing, except when using the Encrypt Rule and certain Generalization operations that output ranges or default strings, since these will naturally require changing the data type (to `string`).

  When writing Parquet files, all columns, including any fields that are originally defined as non-nullable (`nullable=false`) in the input, are automatically converted to be nullable for compatibility reasons.
- When reading Parquet data, default values that are used for CSV related settings (that is, date formats) are ignored by Privitar.

17.3. Privitar Data Types

The Privitar platform strictly validates data types as per the Schema. If the defined Privitar Schema does not match the data types of the underlying input file, errors are expected during processing.

Columns in a Privitar Schema may be of the following data types:

Data type	Description
Text	A string value. That is, a sequence of Unicode characters.
Boolean	A binary value, True or False.
Byte	An 8-bit integer value.

Data type	Description
Short	A 16-bit integer value.
Integer	A 32-bit integer value.
Long	A 64-bit integer value.
Float	A 32-bit precision floating point value.
Double	An 64-bit precision floating point value.
Date	A date without any time reference.
Timestamp	An instant point in time made up of both a date and time component. Privitar accepts milliseconds, microseconds and INT96-based nanoseconds precision timestamps. **Note** NiFi does not support timestamp-micros in version 1.6.0, in version 1.8.0 and later. It converts microseconds to milliseconds, therefore losing precision. This is outside of the control of the Privitar platform.
Other	The Other type is used when data is loaded from a source format that contains types that Privitar does not support. For example, Avro files may contain nested structured types. When defining the Privitar Schema, any such fields may be marked as **Other**. This allows those fields to be passed through unchanged or dropped by specifying *Retain* or *Drop* rules on them in the Privitar Policy. Downstream systems consuming the Privitar output can still use those values if retained.

17.4. Privitar Multi-Language Support

The Privitar Platform supports the Masking and Unmasking of text in multiple languages. That is, data in different languages can be de-identified using the Privitar Masking Rules. For example, the Regular Text Expression Generator rule can be used to generate outputs in multiple languages.

The languages supported by Privitar are those that are encoded in UTF-8 and on the Basic Multilingual Plane (BMP). (For more information, see Basic Multilingual Plane.)

The following table lists the languages that are supported on the Privitar platform.

Note

There are some languages that may use some characters that are not on the BMP. (These languages are highlighted in the table.)

Language	Comments
Afrikaans	Fully supported
Albanian	Fully supported

Language	Comments
Arabic	Fully supported
Armenian	Fully supported
Basque	Fully supported
Bengali	Fully supported
Bosnian (Cyrillic)	Fully supported
Bosnian (Latin)	Fully supported
Bulgarian	Fully supported
Burmese	Fully supported
Catalan	Fully supported
Chinese (Simplified)	Partial support – some characters not on the BMP
Chinese (Traditional)	Partial support – some characters not on the BMP
Croatian	Fully supported
Czech	Fully supported
Danish	Fully supported
Dutch	Fully supported
English	Fully supported
Estonian	Fully supported
Farsi	Fully supported
Finnish	Fully supported
French	Fully supported
Georgian	Fully supported
German	Fully supported
Greek	Fully supported
Gujarati	Fully supported
Hebrew	Fully supported
Hindi	Fully supported
Hungarian	Fully supported
Icelandic	Fully supported
Indonesian	Fully supported
Irish	Fully supported
Italian	Fully supported
Japanese: Hiragana	Fully supported
Japanese: Katakana	Fully supported
Japanese: Kanji	Partial support – some characters not on the BMP
Kannada	Fully supported
Khmer	Fully supported
Korean	Fully supported
Latvian	Fully supported
Lithuanian	Fully supported
Luxembourgish	Fully supported
Macedonian	Fully supported

Language	Comments
Malay	Fully supported
Malayalam	Fully supported
Maltese	Fully supported
Marathi	Fully supported
Montenegrin	Fully supported
Norwegian	Fully supported
Polish	Fully supported
Portuguese (Brazil)	Fully supported
Portuguese (European)	Fully supported
Romanian (Cyrillic)	Fully supported
Romanian (Latin)	Fully supported
Romansh	Fully supported
Russian	Fully supported
Serbian (Cyrillic)	Fully supported
Serbian (Latin)	Fully supported
Slovak	Fully supported
Slovenian	Fully supported
Spanish	Fully supported
Swahili (Latin)	Fully supported
Swahili (Arabic)	Fully supported
Swedish	Fully supported
Tagalog	Fully supported
Tamil	Fully supported
Te reo	Fully supported
Telugu	Fully supported
Thai	Fully supported
Turkish	Fully supported
Ukrainian	Fully supported
Urdu	Fully supported
Vietnamese	Fully supported
Welsh	Fully supported
Xhosa	Fully supported
Zulu	Fully supported

17.5. Token Vault Types

The types of Token Vaults supported by Privitar are described in the table below:

Token Vault Type	Description	Requires Hadoop?	Jobs compatibility
HDFS	Hadoop File System. This Token Vault type can only be selected in environments with Hadoop Clusters. When closing a PDD in an Environment with an HDFS Token Vault, Privitar will start a Spark job to delete the corresponding Token Vault.	Yes	Batch (Hive/HDFS types only. Not AWS Glue.)
HBase (and Google Cloud Bigtable)	The HBase distributed Hadoop database. This Token Vault type can only be selected in environments with Hadoop Clusters but can also be accessed by pipelines running Data Flow Jobs. When closing a PDD in an Environment with an HBase Token Vault, Privitar will start a Spark job to delete the corresponding Token Vault.	Yes	Batch (Hive/HDFS types only. Not AWS Glue.) Data Flow On Demand
JDBC	JDBC-compatible relational database. Supported databases are Oracle and PostgreSQL. Depending on the Environment configuration, a JDBC Token Vault can be closed using Privitar or Privitar On Demand (POD)	No	Batch (Hive/HDFS types only. Not AWS Glue.) Data Flow On Demand
DynamoDB	DynamoDB is a NoSQL database service provided by Amazon.com as part of Amazon Web Services.	No	Batch (All types.) Data Flow On Demand

17.6. Event Types

This document contains details about the different event types captured by Privitar

17.6.1. List of Event Types

- DATA_FLOW_JOB_RUN

  This event is published when a data flow job is run
- DATA_FLOW_METRIC

  This event is published at a fixed interval and contains information about different data flow performance metrics
- DATA_FLOW_PROCESSOR_CLOSED

  This event is published whenever a data flow processor is closed
- DATA_FLOW_PROCESSOR_CREATED

  This event is published whenever a new data flow processor is created

- ENVIRONMENT_CREATED

  This event is published whenever a new environment is created
- ENVIRONMENT_DELETED

  This event is published whenever a environment is deleted
- ENVIRONMENT_UPDATED

  This event is published whenever a environment is updated
- FIELD_ID_AND_DATA_TYPE_USAGE_COUNT

  This event is published when a report is created about how many rules is column assigned to and how many times it is assigned
- INTEGRATION_START_UP

  This event is published whenever a data flow integration is started
- INTEGRITY

  This event is published when log files are rolled over as an integrity measure
- JOB_CREATED

  This event is published whenever a new job is created
- JOB_DELETED

  This event is published whenever a job is deleted
- JOB_RUN

  This event is published whenever a job is run
- JOB_UPDATED

  This event is published whenever a job is updated
- LOGON

  This event is published whenever a user logs on
- METRIC

  This event is published at a fixed interval and contains information about different metrics
- PDD_CLOSED

  This event is published whenever a Protected Data Domain is closed
- PDD_CREATED

  This event is published whenever a new Protected Data Domain is created
- PDD_DELETED

  This event is published whenever a Protected Data Domain is deleted
- PDD_UPDATED

  This event is published whenever a Protected Data Domain is updated
- POLICY_CREATED

  This event is published whenever a new policy is created
- POLICY_DELETED

  This event is published whenever a policy is deleted
- POLICY_MANAGER_STARTED

This event is published whenever policy manager starts up

- POLICY_UPDATED

  This event is published whenever a policy is updated
- PPO

  This event is published whenever a job that performs Privacy Preserving Operations is run. A Privacy Preserving Operation is any operation that is not a retain or drop.
- PROCESSOR_START_UP

  This event is published when the spark processor is started
- RECORDS_PROCESSED

  This event is published when a job is run, and it contains details about the number of records processed.
- REMOVE_TOKEN_MAPPING_FINISHED

  This event is published when a request to forget a user's identifying value has finished processing
- REMOVE_TOKEN_MAPPING_STARTED

  This event is published when there is a new request to forget a user's identifying value
- RULE_CREATED

  This event is published whenever a new rule is created
- RULE_DELETED

  This event is published whenever a rule is deleted
- RULE_PREVIEWED

  This event is published whenever a rule is previewed
- RULE_UPDATED

  This event is published whenever a rule is updated
- SCHEMA_CREATED

  This event is synthesized whenever a new schema is created
- SCHEMA_DELETED

  This event is published whenever a schema is deleted
- SCHEMA_UPDATED

  This event is published whenever a schema is updated
- SERVICES_STATUS

  This event is published frequently with information about the current state of connected Privitar services
- SERVICE_STATUS_CHANGED

  This event is published whenever a Privitar Service has a new health status
- STEP_REPORT

  This event is published when a job completes, and contains information about the job's performance from the step reports
- SYSTEM_INFORMATION

This event is published on start up and contains information about the system the service is running on

- SYSTEM_USER_CREATED

  This event is published whenever a new system user is created
- SYSTEM_USER_DELETED

  This event is published whenever a system user is deleted
- SYSTEM_USER_UPDATED

  This event is published whenever a system user is updated
- TEAM_CREATED

  This event is published whenever a new team is created
- TEAM_DELETED

  This event is published whenever a team is deleted
- TEAM_DELETION_REQUEST

  This event is published whenever a request to delete a team is made
- TEAM_MEMBERSHIP_ADDED

  This event is published whenever a new team member is added
- TEAM_MEMBERSHIP_REMOVED

  This event is published whenever a team member is removed
- TEAM_UPDATED

  This event is published whenever a team is updated
- UNIQUE_COLUMN_COUNT_PER_RULE

  This event is published when we count for the rule how many different combinations (Field name, Data Type) is it used
- USER_CREATED

  This event is published whenever a new user is created
- USER_UPDATED

  This event is published whenever a user is updated
- WATERMARK_EXTRACTION_END

  This event is published when a watermark extraction finished. This can be published multiple times for a single job
- WATERMARK_EXTRACTION_SUBMITTED

  This event is published when a watermark extraction result gets submitted for matching
- WATERMARK_INVESTIGATION_END

  This event is published when a watermark investigation job is completed
- WATERMARK_INVESTIGATION_START

  This event is published when a watermark investigation job is created

17.6.2. Additional Information

List of Metric Types

- **AVAILABLE_SYSTEM_MEMORY**

  The amount of memory available. It is recorded as a gauge, and the current value is extracted whenever the metric is published. The unit is bytes
- **BLIND_SECURELINK_POINTS**

  The time it takes to make a SecureLink blind request. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **CACHE_GET_TOKENS_FOR_VALUES**

  The time it takes to get tokens for a given list of values from the token vault cache. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **CACHE_GET_VALUES_FOR_TOKENS**

  The time it takes to get values for a given list of tokens from the token vault cache. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **CACHE_LOAD_TOKENS**

  The time it takes to load tokens into the token vault cache. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **CACHING_RULE_APPLY_MASK**

  The time it takes to mask a batch of records when utilising caching. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **CACHING_RULE_UNMASK**

  The time it takes to unmask a batch of records when utilising caching. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **CONSISTENT_MASK**

  The time it takes to consistently tokenise a batch of records. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **CONSISTENT_MASK_GENERATE_TOKENS**

  The time it takes to generate tokens for consistent tokenisation of a batch of records. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **CPU_LOAD**

  The total number of runnable entities queued and running on the processors, averaged over the past minute. It is recorded as a gauge, and the current value is extracted whenever the metric is published. The unit is processes per minute
- **DATA_FLOW_DEFINITION_REQUEST_ATTEMPT_HISTOGRAM**

  The number of Privitar data flow job definition request attempts. It is recorded as a histogram, and the metric is updated with every request attempt. The statistical characteristics of the stream of recorded values are computed whenever the metric is published. The unit is attempts
- **DATA_FLOW_DEFINITION_REQUEST_RETRIES**

  The rate at which Privitar data flow job definition requests are retried when there is an error. It is recorded using a meter, and the unit is retries per second.
- **DECRYPT_BLINDED_SECURELINK_POINTS**

  The time it takes to decrypt the given blinded SecureLink points. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **DYNAMO_ENSURE_TABLE_EXISTS**

  The time it takes to check if a DynamoDb token vault table exists and create one if it doesn't. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **DYNAMO_GET_VAULT_SIZE**

  The time it takes to calculate the size of a DynamoDb token vault. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **DYNAMO_TOKEN_TO_VALUE_DELETE_ITEM**

  The time it takes to delete an item from a DynamoDb token to value table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **DYNAMO_TOKEN_TO_VALUE_GET_ITEMS**

The time it takes to get a list of items from a DynamoDb token to value table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **DYNAMO_TOKEN_TO_VALUE_WRITE_ITEM**

  The time it takes to write an item to a DynamoDb token to value table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **DYNAMO_VALUE_TO_TOKEN_GET_ITEMS**

  The time it takes to get a list of items from a DynamoDb value to token table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **DYNAMO_VALUE_TO_TOKEN_WRITE_ITEM**

  The time it takes to write an item to a DynamoDb value to token table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **DYNAMO_VAULT_INTERNAL_SERVER_ERRORS**

  The rate at which internal server error exceptions are thrown during DynamoDb operations. It is recorded using a meter, and the unit is exceptions per second.

- **DYNAMO_VAULT_THROTTLED_DDL**

  The rate at which throttled exceptions are thrown during DynamoDb DDL operations. It is recorded using a meter, and the unit is exceptions per second.

- **DYNAMO_VAULT_THROTTLED_ITEMS**

  The rate at which throttled exceptions are thrown during DynamoDb non-DDL operations. It is recorded using a meter, and the unit is exceptions per second.

- **FREE_DISK_SPACE**

  The amount of free disk space. It is recorded as a gauge, and the current value is extracted whenever the metric is published. The unit is megabytes

- **GET_DATA_FLOW_JOB_CONFIG**

  The time it takes for a Privitar service to retrieve the data flow job definition. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **GET_VAULT_CONNECTION_FROM_CACHE**

  The time it takes to acquire a connection to the token vault from the cache. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **HBASE_TOKEN_TO_VALUE_GET_ITEM**

  The time it takes to retrieve an item from an HBase token to value table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **HBASE_TOKEN_TO_VALUE_WRITE_ITEM**

  The time it takes to write an item to an HBase value to token table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **HBASE_VALUE_TO_TOKEN_GET_ITEM**

  The time it takes to retrieve an item from an HBase value to token table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **HBASE_VALUE_TO_TOKEN_WRITE_ITEM**

  The time it takes to write an item to an HBase value to token table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **HBASE_VAULT_DELETE_UNUSED_TOKEN**

  The time it takes to delete unused tokens from an HBase token vault. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **HBASE_VAULT_GET_EXISTING_MAPPINGS_FOR_VALUES**

  The time it takes to get existing value to token mappings from an HBase value to token table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **HBASE_VAULT_GET_VALUES_FOR_TOKENS**

  The time it takes to retrieve values for a given list of tokens in an HBase token to value table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **INITIAL_VAULT_SIZE**

The total size(s) of the token vault(s). For non-combined token vaults, it is recorded as a list of gauges - one for each rule applied, otherwise it is recorded as a single gauge for the single combined vault. The current value(s) are extracted whenever the metric is published. The unit is bytes

- **INPUT_DATA_CARDINALITY**

  The cardinality of the input data. The cardinality of a column is the number of unique values in that column. It is recorded as a list of histograms - one for each column in the input data, and the metric is updated with every batch of records processed. The statistical characteristics of the streams of recorded values are computed whenever the metric is published. The unit is rows

- **INPUT_DATA_NUM_ROWS**

  The total number of input rows processed for each batch of records. It is recorded as a histogram, and the metric is updated with every batch of records processed. The statistical characteristics of the stream of recorded values are computed whenever the metric is published. The unit is rows

- **INTERMEDIARY_BLIND_REQUEST_ATTEMPT_HISTOGRAM**

  The number of Privitar Intermediary service blind request attempts. It is recorded as a histogram, and the metric is updated with every request attempt. The statistical characteristics of the stream of recorded values are computed whenever the metric is published. The unit is attempts

- **INTERMEDIARY_BLIND_REQUEST_RETRIES**

  The rate at which blind requests to the Privitar Intermediary service are retried when there is an error. It is recorded using a meter, and the unit is retries per second.

- **JDBC_PARALLEL_SCAN_QUEUE_OCCUPANCY**

  The ratio of the size of the batch parallel scan queue to its maximum capacity. It is recorded as a gauge, and the current value is extracted whenever the metric is published. The unit is size per capacity

- **JDBC_VAULT_NUM_BUSY_CONNECTIONS**

  The number of busy connections to the JDBC token vault. It is recorded as a gauge, and the current value is extracted whenever the metric is published. The unit is connections

- **JDBC_VAULT_NUM_CONNECTIONS**

  The number of connections to the JDBC token vault. It is recorded as a gauge, and the current value is extracted whenever the metric is published. The unit is connections

- **MASKING_PROCESSOR_THROUGHPUT**

  The rate at which input records are processed in a masking operation. It is recorded using a meter, and the unit is records per second.

- **MASK_BATCH**

  The time it takes to mask a batch of records. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **NIFI_MASK**

  The time it takes to run a NiFi masking job. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **NUMBER_OF_PPOS**

  The total number of privacy preserving operations performed for each batch of records. A PPO is any operation that is not a retain or a drop. It is recorded as a histogram, and the metric is updated with every batch of records processed. The statistical characteristics of the stream of recorded values are computed whenever the metric is published. The unit is PPOs

- **NUMBER_OF_VALUES_WRITTEN_TO_TOKEN_VAULT**

  The total number of values written to the token vault for each batch of records. It is recorded as a histogram, and the metric is updated with every batch of records processed. The statistical characteristics of the stream of recorded values are computed whenever the metric is published. The unit is values

- **ORACLE_TOKEN_TO_VALUE_GET_VAULT_SIZE**

  The time it takes to calculate the size of an Oracle token to value table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **ORACLE_VALUE_TO_TOKEN_GET_VAULT_SIZE**

  The time it takes to calculate the size of an Oracle value to token table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **ORACLE_VAULT_DELETE_UNUSED_TOKENS**

  The time it to delete unused tokens in an Oracle token vault. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **ORACLE_VAULT_GET_AND_RESOLVE_DUPLICATE_TOKENS**

  The time it takes to resolve duplicate tokens in an Oracle token vault. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **ORACLE_VAULT_GET_EXISTING_MAPPINGS_FOR_VALUES**

  The time it takes to get existing value to token mappings for a given list of values, in an Oracle token vault. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **ORACLE_VAULT_GET_VALUES_FOR_TOKENS**

  The time it takes to retrieve the values for a given list of tokens in an Oracle token vault. It is recorded using a timer, which behaves as a meter and a histogram combined.

The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **ORACLE_VAULT_GET_VALUE_TO_TOKEN_MAPPINGS**

  The time it takes to get value to token mappings from an Oracle token vault. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **ORACLE_VAULT_WRITE_TOKEN_TO_VALUE_MAP**

  The time it takes to write token to value mappings to an Oracle token vault. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **ORACLE_VAULT_WRITE_VALUE_TO_TOKEN_MAP**

  The time it takes to write value to token mappings to an Oracle token vault. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **POD_APPLY_POLICY**

  The time it takes to apply a Privitar policy for a Privitar On Demand job. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **POD_GET_MASKING_PROCESSOR**

  The time it takes to create a masking processor for a Privitar On Demand job. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **POD_GET_ORDINARY_UNMASKING_PROCESSOR**

  The time it takes to create an unmasking processor for a Privitar On Demand job. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **POD_GET_REMASKING_PROCESSOR**

  The time it takes to create a remasking processor for a Privitar On Demand job. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **POD_GET_SL_UNMASKING_PROCESSOR**

The time it takes to create a SecureLink unmasking processor for a Privitar On Demand job. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **POD_UNMASK**

  The time it takes to perform a Privitar On Demand unmasking request. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **POD_UNMASK_SECURELINK**

  The time it takes to perform a Privitar On Demand unmasking request for SecureLink. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **POSTGRESQL_TOKEN_TO_VALUE_GET_VAULT_SIZE**

  The time it takes to get the size of an PostgreSQL token to value table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **POSTGRESQL_VALUE_TO_TOKEN_GET_VAULT_SIZE**

  The time it takes to get the size of a PostgreSQL value to token table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **POSTGRESQL_VAULT_DELETE_UNUSED_TOKENS**

  The time it takes to delete unused tokens in a PostgreSQL token vault. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **POSTGRESQL_VAULT_GET_AND_RESOLVE_DUPLICATE_MAPPINGS**

  The time it takes to resolve duplicate mappings in a PostgreSQL token vault. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **POSTGRESQL_VAULT_GET_EXISTING_MAPPINGS_FOR_VALUES**

  The time it takes to get existing value to token mappings from a PostgreSQL value to token table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **POSTGRESQL_VAULT_GET_TOKEN_TO_VALUE_MAPPINGS**

  The time it takes to retrieve token to value mappings from a PostgreSQL token to value table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **POSTGRESQL_VAULT_GET_VALUES_FOR_TOKENS**

  The time it takes to retrieve the values for a given list of tokens in a PostgreSQL token vault. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **POSTGRESQL_VAULT_WRITE_TOKEN_TO_VALUE_MAP**

  The time it takes to write token to value mappings to a PostgreSQL token to value table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **POSTGRESQL_VAULT_WRITE_VALUE_TO_TOKEN_MAP**

  The time it takes to write value to token mappings to a PostgreSQL value to token table. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **PROCESSOR_BATCH_EXECUTE**

  The time it takes to execute an operation on a batch of records. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **PROCESS_RECORDS**

  The time it takes to process all the batches in the input data. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **READ_RECORD_BATCH**

  The time it takes to read a batch of input records. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **REBLIND_SECURELINK_POINTS**

  The time it takes to make a SecureLink reblind request. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **SECURELINK_MASK**

  The time it takes to mask a batch of records with SecureLink. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **SHARED_MASKING_PROCESSOR_THROUGHPUT**

  The rate at which input records are processed in a masking operation, if sharing infrastructure between processing threads. It is recorded using a meter, and the unit is records per second.

- **SHARED_UNMASKING_PROCESSOR_THROUGHPUT**

  The rate at which input records are processed in an unmasking operation, if sharing infrastructure between processing threads. It is recorded using a meter, and the unit is records per second.

- **STREAMSETS_MASK**

  The time it takes to process a batch of input records using StreamSets. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **SWAP_TOTAL**

  The total size of the swap partition It is recorded as a gauge, and the current value is extracted whenever the metric is published. The unit is bytes

- **THREAD_COUNT**

  The current number of live threads. It is recorded as a gauge, and the current value is extracted whenever the metric is published. The unit is threads

- **TOKENISATION_ATTEMPTS**

  The number of attempts it takes to tokenise a batch of raw values. It is recorded as a histogram, and the metric is updated with every batch of records processed. The statistical characteristics of the stream of recorded values are computed whenever the metric is published. The unit is attempts

- **UNMASKING_PROCESSOR_THROUGHPUT**

  The rate at which input records are processed in an unmasking operation. It is recorded using a meter, and the unit is records per second.

- **UNMASK_BATCH**

  The time it takes to unmask a batch of records. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

- **VALUES_TO_TOKENISE**

  The number of raw values to be tokenised in a batch. It is recorded as a histogram, and the metric is updated with every batch of records processed. The statistical characteristics of the stream of recorded values are computed whenever the metric is published. The unit is values

- **VAULT_CACHE_ENTRY_COUNT**

  The estimated number of values in the token vault cache. It is recorded as a gauge, and the current value is extracted whenever the metric is published. The unit is values
- **VAULT_CACHE_ENTRY_WEIGHT**

  The total size in bytes of the token vault cache. It is recorded as a gauge, and the current value is extracted whenever the metric is published. The unit is bytes
- **VAULT_CACHE_EVICTION_COUNT**

  The number of times an entry in the token vault cache was evicted. It is recorded as a gauge, and the current value is extracted whenever the metric is published. The unit is times
- **VAULT_CACHE_HIT_RATE**

  The ratio of token vault cache requests which were hits. It is recorded as a gauge, and the current value is extracted whenever the metric is published. The unit is hits per request
- **VAULT_CACHE_LOAD_COUNT**

  The number of times the token vault cache attempted to load a new value. It is recorded as a gauge, and the current value is extracted whenever the metric is published. The unit is times
- **VAULT_START**

  The time it takes to initialise the token vault tables. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds
- **VAULT_STORE_TOKENS_IF_UNUSED**

  The time it takes to store unused tokens in a token vault. It is recorded using a timer, which behaves as a meter and a histogram combined. The metric is updated each time the operation is performed. Both the rate and the statistical characteristics of the recorded times are computed whenever the metric is published. The unit is nanoseconds

17.6.3. DATA_FLOW_JOB_RUN

This event is published when a data flow job is run

Event Fields:

- **eventType**

  The type of the event

  Value: DATA_FLOW_JOB_RUN
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS

- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **uniqueId**

    The unique id of the job run
  - **creatorId**

    The id of the job run creator
  - **job**

    The parameters of the job associated with this job run

    See the parameters field of the 'JOB_CREATED' Event for more details
  - **startTimestamp**

    The UTC timestamp when the processing of the batch of records described by this event begins
  - **endTimestamp**

    The UTC timestamp when the processing of the batch of records described by this event ends
  - **totalRecords**

    The total number of successfully processed records
  - **totalBadRecords**

    The total number of bad records
  - **watermarked**

    Whether or not the job run includes watermarking
  - **count**

    The number of individual events that this event is an aggregation of
  - **processorId**

    The id of the data flow processor that created this event
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.4. DATA_FLOW_METRIC

This event is published at a fixed interval and contains information about different data flow performance metrics

Event Fields:

- **eventType**

  The type of the event

  Value: DATA_FLOW_METRIC
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **metricName**

    The name of the metric

    Example Values: JDBC_VAULT_NUM_CONNECTIONS, JDBC_VAULT_NUM_BUSY_CONNECTIONS

    See the 'List of Metric Types' section for more details
  - **unit**

    The unit of the metric
  - **metadata**

    The metadata that accompanies the metric

    Sub-fields:

    - **jobId**

      The id of the job the metric pertains to
    - **processorId**

      The id of the data flow processor that this metric pertains to
    - **ruleId**

      The id of the rule the metric pertains to
    - **environmentId**

      The id of the environment
    - **jobType**

      The type of the Job

      Example Values: BATCH, POD, DATAFLOW
    - **pipelineId**

      The id of the data flow pipeline
    - **tokenizationBatchSize**

      The maximum batch size when processing records
    - **tokenizationThreadCount**

      The maximum number of threads to use when processing records

  - **vaultType**

    The token vault type associated with the job

    Example Values: HDFS, JDBC, NONE
  - **vaultDbEngine**

    The token vault database engine, if one has been set up

    Example Values: Oracle, Postgres, DynamoDb, HBase
  - **vaultDbVersion**

    The token vault database version, if one has been set up
  - **vaultConfigProperties**

    Other configuration properties of the token vault associated with the job. Note: sensitive properties will be redacted
  - **diskId**

    The volume id for the disk
  - **numberOfColumns**

    The number of columns in the input data
  - **numberOfOptionalColumns**

    The number of optional columns in the input data
- **gaugeValues**

  A list of gauge values

  Sub-fields:
  - **specificMetricMetadataValues**

    The metadata associated with this gauge

    Sub-fields:
    - **columnId**

      The column id
    - **ruleId**

      The rule id
  - **value**

    The value of the gauge
- **histogramValues**

  A list of histogram details.

  Sub-fields:
  - **specificMetricMetadataValues**

    The metadata associated with this histogram

    Sub-fields:
    - **columnId**

      The column id
    - **ruleId**

    The rule id

  - **snapshotCount**

    The number of values contained in the histogram snapshot
  - **maxValue**

    The largest value in the histogram snapshot
  - **minValue**

    The smallest value in the histogram snapshot
  - **medianValue**

    The median value in the histogram snapshot
  - **meanValue**

    The mean value in the histogram snapshot
  - **stdev**

    The standard deviation of values in the histogram snapshot
  - **seventyFifthPercentile**

    The seventy fifth percentile of values in the histogram snapshot
  - **ninetyFifthPercentile**

    The ninety fifth percentile of values in the histogram snapshot
  - **ninetyEighthPercentile**

    The ninety eighth percentile of values in the histogram snapshot
  - **ninetyNinthPercentile**

    The ninety ninth percentile of values in the histogram snapshot
  - **ninetyNinePointNinthPercentile**

    The ninety nine point ninth percentile of values in the histogram snapshot
- **meter**

  The details of the rate of values

  Sub-fields:

  - **count**

    The number of values recorded
  - **meanRate**

    The mean rate of events over time
  - **oneMinuteRate**

    The moving average rate of events over the last one minute
  - **fiveMinuteRate**

    The moving average rate of events over the last five minutes
  - **fifteenMinuteRate**

    The moving average rate of events over the last fifteen minutes

- **uuid**

  The event unique identifier

- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.5. DATA_FLOW_PROCESSOR_CLOSED

This event is published whenever a data flow processor is closed

Event Fields:

- **eventType**

  The type of the event

  Value: DATA_FLOW_PROCESSOR_CLOSED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **jobId**

    The job id
  - **processorId**

    The id of the data flow processor that created this event
  - **jobType**

    The job type

    Example Values: BATCH, POD, DATAFLOW
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.6. DATA_FLOW_PROCESSOR_CREATED

This event is published whenever a new data flow processor is created

Event Fields:

- **eventType**

  The type of the event

  Value: DATA_FLOW_PROCESSOR_CREATED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **jobId**

    The job id
  - **processorId**

    The id of the data flow processor that created this event
  - **jobType**

    The job type

    Example Values: BATCH, POD, DATAFLOW
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.7. ENVIRONMENT_CREATED

This event is published whenever a new environment is created

Event Fields:

- **eventType**

  The type of the event

  Value: ENVIRONMENT_CREATED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

The event parameters

Sub-fields:

- **id**

  The environment id
- **uniqueId**

  The unique id of the environment
- **team**

  The team of the environment creator

  See the parameters field of the 'TEAM_CREATED' Event for more details
- **tokenVaultType**

  The token vault type, if one has been set up

  Example Values: NONE, HDFS, JDBC
- **useDerivedTokenisation**

  Whether or not the token vault, if set up, uses derived tokenisation
- **useVaultEncryption**

  Whether or not the token vault, if set up, uses vault encryption
- **kmsType**

  The key management system (kms) type associated with this environment, if any has been set up

  Example Values: NONE, HADOOP, JAVA_KEYSTORE
- **usePodToClosePdds**

  Whether or not to use Publisher on Demand to close protected data domains
- **hadoopDistributionType**

  The Hadoop distribution type

  Example Values: DEFAULT
- **hadoopClusterDataSource**

  The Hadoop cluster data source

  Example Values: HDFS, NATIVEFS
- **useKerberos**

  Whether or not to use kerberos for this environment
- **runAs**

  The type of user to run jobs as

  Example Values: PRIVITAR_USER, JOB_OPERATOR, SERVICE_USER
- **outputsToSentry**

  Whether or not Protected Data Domain paths are managed by Sentry
- **hiveEnabled**

  Whether or not Hive is enabled
- **sharees**

    The list of teams that share this environment

    See the parameters field of the 'TEAM_CREATED' Event for more details

  - **sharedWithAllTeams**

    Whether or not this environment is shared with all teams

- **uuid**

  The event unique identifier

- **applicationVersion**

  The version of the Privitar application sending the event

- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.8. ENVIRONMENT_DELETED

This event is published whenever a environment is deleted

Event Fields:

- **eventType**

  The type of the event

  Value: ENVIRONMENT_DELETED

- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS

- **timestamp**

  The UTC timestamp when the event was created

- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The environment id

  - **uniqueId**

    The unique id of the environment

  - **team**

    The team of the environment creator

    See the parameters field of the 'TEAM_CREATED' Event for more details

  - **tokenVaultType**

    The token vault type, if one has been set up

    Example Values: NONE, HDFS, JDBC

  - **useDerivedTokenisation**

    Whether or not the token vault, if set up, uses derived tokenisation

- **useVaultEncryption**

  Whether or not the token vault, if set up, uses vault encryption
- **kmsType**

  The key management system (kms) type associated with this environment, if any has been set up

  Example Values: NONE, HADOOP, JAVA_KEYSTORE
- **usePodToClosePdds**

  Whether or not to use Publisher on Demand to close protected data domains
- **hadoopDistributionType**

  The Hadoop distribution type

  Example Values: DEFAULT
- **hadoopClusterDataSource**

  The Hadoop cluster data source

  Example Values: HDFS, NATIVEFS
- **useKerberos**

  Whether or not to use kerberos for this environment
- **runAs**

  The type of user to run jobs as

  Example Values: PRIVITAR_USER, JOB_OPERATOR, SERVICE_USER
- **outputsToSentry**

  Whether or not Protected Data Domain paths are managed by Sentry
- **hiveEnabled**

  Whether or not Hive is enabled
- **sharees**

  The list of teams that share this environment

  See the parameters field of the 'TEAM_CREATED' Event for more details
- **sharedWithAllTeams**

  Whether or not this environment is shared with all teams

- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.9. ENVIRONMENT_UPDATED

This event is published whenever a environment is updated

Event Fields:

- **eventType**

  The type of the event

  Value: ENVIRONMENT_UPDATED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The environment id
  - **uniqueId**

    The unique id of the environment
  - **team**

    The team of the environment creator

    See the parameters field of the 'TEAM_CREATED' Event for more details
  - **tokenVaultType**

    The token vault type, if one has been set up

    Example Values: NONE, HDFS, JDBC
  - **useDerivedTokenisation**

    Whether or not the token vault, if set up, uses derived tokenisation
  - **useVaultEncryption**

    Whether or not the token vault, if set up, uses vault encryption
  - **kmsType**

    The key management system (kms) type associated with this environment, if any has been set up

    Example Values: NONE, HADOOP, JAVA_KEYSTORE
  - **usePodToClosePdds**

    Whether or not to use Publisher on Demand to close protected data domains
  - **hadoopDistributionType**

    The Hadoop distribution type

    Example Values: DEFAULT
  - **hadoopClusterDataSource**

    The Hadoop cluster data source

    Example Values: HDFS, NATIVEFS
  - **useKerberos**

  Whether or not to use kerberos for this environment
  - **runAs**

    The type of user to run jobs as

    Example Values: PRIVITAR_USER, JOB_OPERATOR, SERVICE_USER
  - **outputsToSentry**

    Whether or not Protected Data Domain paths are managed by Sentry
  - **hiveEnabled**

    Whether or not Hive is enabled
  - **sharees**

    The list of teams that share this environment

    See the parameters field of the 'TEAM_CREATED' Event for more details
  - **sharedWithAllTeams**

    Whether or not this environment is shared with all teams
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.10. FIELD_ID_AND_DATA_TYPE_USAGE_COUNT

This event is published when a report is created about how many rules is column assigned to and how many times it is assigned

Event Fields:

- **eventType**

  The type of the event

  Value: FIELD_ID_AND_DATA_TYPE_USAGE_COUNT
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **fieldUniqueId**

    Unique Id of Field being used.

  - **dataType**

    Datatype of Field being used.
  - **rulesCount**

    How many different Rules are assigned to it.
  - **ruleUniqueIdAndUsageCount**

    How many times a particular Rule has been assigned to such Field.

    Sub-fields:

    - **ruleUniqueId**

      Unique Id of Rule being used.
    - **usageCount**

      How many times this Rule has been assigned.
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.11. INTEGRATION_START_UP

This event is published whenever a data flow integration is started

Event Fields:

- **eventType**

  The type of the event

  Value: INTEGRATION_START_UP
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **integrationVersion**

    The version of the integration that is starting up
  - **javaVersion**

    The version of java that the integration is running in
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.12. INTEGRITY

This event is published when log files are rolled over as an integrity measure

Event Fields:

- **eventType**

  The type of the event

  Value: INTEGRITY
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **previousHash**

    The hash contained in the integrity event of the previous log file
  - **hash**

    Ensures integrity of the log files
  - **rolls**

    Number of times we have rolled the log file
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.13. JOB_CREATED

This event is published whenever a new job is created

Event Fields:

- **eventType**

The type of the event

Value: JOB_CREATED

- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS

- **timestamp**

  The UTC timestamp when the event was created

- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The job id

  - **uniqueId**

    The unique id of the job

  - **team**

    The team of the job creator

    See the parameters field of the 'TEAM_CREATED' Event for more details

  - **creatorId**

    The id of the job creator

  - **hasVaults**

    Whether or not the job is configured with token vaults

  - **runAsServiceUser**

    The service user to run this job as

    Sub-fields:

    - **id**

      The service user id

    - **uniqueId**

      The service user unique id

  - **badRecordHandling**

    The failure mechanism when there are bad records

    Example Values: FAIL_WHEN_ALL_RECORDS_BAD

  - **badRecordsThreshold**

    The maximum amount of bad records allowed before failing the job

  - **jobPurpose**

    The purpose of the job

  - **cachingEnabled**

    Whether or not caching is enabled

  - **outputType**

The job output mechanism e.g whether or not to overwrite existing data

Example Values: OVERWRITE

- **objectType**

  The object type
- **sparkOverrides**

  The details of the spark overrides

  Sub-fields:

  - **sparkExecutors**

    The number of spark executors
  - **sparkExecCores**

    The number of cores per spark executor
  - **sparkExecMem**

    The amount of memory per spark executor
  - **sparkDriverMem**

    The amount of memory given to the spark driver
  - **yarnExecMemOverhead**

    The executor's yarn memory overhead
- **protectedDataDomain**

  The parameters of the protected data domain associated with this job

  See the parameters field of the 'PDD_CREATED' Event for more details
- **policies**

  The parameters of the policies associated with this job

  See the parameters field of the 'POLICY_CREATED' Event for more details
- **environment**

  The details of the job environment

  See the parameters field of the 'ENVIRONMENT_CREATED' Event for more details
- **secureLinkDestination**

  The securelink destination
- **tables**

  The input tables

  Sub-fields:

  - **id**

    The job run table id
  - **uniqueId**

    The job run table's unique id
  - **dataFormat**

    The job run table's data format

    Example Values: CSV, Avro, Hive, Parquet

- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.14. JOB_DELETED

This event is published whenever a job is deleted

Event Fields:

- **eventType**

  The type of the event

  Value: JOB_DELETED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The job id
  - **uniqueId**

    The unique id of the job
  - **team**

    The team of the job creator

    See the parameters field of the 'TEAM_CREATED' Event for more details
  - **creatorId**

    The id of the job creator
  - **hasVaults**

    Whether or not the job is configured with token vaults
  - **runAsServiceUser**

    The service user to run this job as

    Sub-fields:

    - **id**

      The service user id

  - **uniqueId**

    The service user unique id

- **badRecordHandling**

  The failure mechanism when there are bad records

  Example Values: FAIL_WHEN_ALL_RECORDS_BAD

- **badRecordsThreshold**

  The maximum amount of bad records allowed before failing the job

- **jobPurpose**

  The purpose of the job

- **cachingEnabled**

  Whether or not caching is enabled

- **outputType**

  The job output mechanism e.g whether or not to overwrite existing data

  Example Values: OVERWRITE

- **objectType**

  The object type

- **sparkOverrides**

  The details of the spark overrides

  Sub-fields:

  - **sparkExecutors**

    The number of spark executors

  - **sparkExecCores**

    The number of cores per spark executor

  - **sparkExecMem**

    The amount of memory per spark executor

  - **sparkDriverMem**

    The amount of memory given to the spark driver

  - **yarnExecMemOverhead**

    The executor's yarn memory overhead

- **protectedDataDomain**

  The parameters of the protected data domain associated with this job

  See the parameters field of the 'PDD_CREATED' Event for more details

- **policies**

  The parameters of the policies associated with this job

  See the parameters field of the 'POLICY_CREATED' Event for more details

- **environment**

  The details of the job environment

  See the parameters field of the 'ENVIRONMENT_CREATED' Event for more details

  - **secureLinkDestination**

    The securelink destination
  - **tables**

    The input tables

    Sub-fields:

    - **id**

      The job run table id
    - **uniqueId**

      The job run table's unique id
    - **dataFormat**

      The job run table's data format

      Example Values: CSV, Avro, Hive, Parquet
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.15. JOB_RUN

This event is published whenever a job is run

Event Fields:

- **eventType**

  The type of the event

  Value: JOB_RUN
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The job run id
  - **creatorId**

    The id of the job run creator

- **requestId**

  The id of the job run request
- **sparkApplicationId**

  The id of the spark application
- **status**

  The job run status

  Example Values: FAILED, SUCCESS, PENDING
- **submittedFrom**

  Where the job was submitted from

  Example Values: DASHBOARD, API
- **totalRecords**

  The total number of successfully processed records
- **totalBadRecords**

  The total number of bad records
- **job**

  The parameters of the job associated with this job run

  See the parameters field of the 'JOB_CREATED' Event for more details
- **uniqueId**

  The unique id of the job run
- **protectedDataDomainId**

  The id of the protected data domain associated with this job run
- **watermarked**

  Whether or not the job run includes watermarking
- **type**

  The job run type

  Example Values: MASKING, DELETING_VAULTS
- **killStarted**

  Whether the process to kill the job run has started
- **serviceUserId**

  The id of the service user running the job
- **requestedExecutors**

  The requested executors
- **provisionedExecutors**

  The provisioned executors
- **startTimestamp**

  The UTC timestamp when the job starts
- **endTimestamp**

  The UTC timestamp when the job ends

  - **executionStartTimestamp**

    The UTC timestamp when the job execution starts
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.16. JOB_UPDATED

This event is published whenever a job is updated

Event Fields:

- **eventType**

  The type of the event

  Value: JOB_UPDATED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The job id
  - **uniqueId**

    The unique id of the job
  - **team**

    The team of the job creator

    See the parameters field of the 'TEAM_CREATED' Event for more details
  - **creatorId**

    The id of the job creator
  - **hasVaults**

    Whether or not the job is configured with token vaults
  - **runAsServiceUser**

    The service user to run this job as

    Sub-fields:

  - **id**

    The service user id
  - **uniqueId**

    The service user unique id
- **badRecordHandling**

  The failure mechanism when there are bad records

  Example Values: FAIL_WHEN_ALL_RECORDS_BAD
- **badRecordsThreshold**

  The maximum amount of bad records allowed before failing the job
- **jobPurpose**

  The purpose of the job
- **cachingEnabled**

  Whether or not caching is enabled
- **outputType**

  The job output mechanism e.g whether or not to overwrite existing data

  Example Values: OVERWRITE
- **objectType**

  The object type
- **sparkOverrides**

  The details of the spark overrides

  Sub-fields:
  - **sparkExecutors**

    The number of spark executors
  - **sparkExecCores**

    The number of cores per spark executor
  - **sparkExecMem**

    The amount of memory per spark executor
  - **sparkDriverMem**

    The amount of memory given to the spark driver
  - **yarnExecMemOverhead**

    The executor's yarn memory overhead
- **protectedDataDomain**

  The parameters of the protected data domain associated with this job

  See the parameters field of the 'PDD_CREATED' Event for more details
- **policies**

  The parameters of the policies associated with this job

  See the parameters field of the 'POLICY_CREATED' Event for more details
- **environment**

    The details of the job environment

    [See the parameters field of the 'ENVIRONMENT_CREATED' Event for more details]

  - **secureLinkDestination**

    The securelink destination

  - **tables**

    The input tables

    Sub-fields:

    - **id**

      The job run table id

    - **uniqueId**

      The job run table's unique id

    - **dataFormat**

      The job run table's data format

      Example Values: CSV, Avro, Hive, Parquet

- **uuid**

  The event unique identifier

- **applicationVersion**

  The version of the Privitar application sending the event

- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.17. LOGON

This event is published whenever a user logs on

Event Fields:

- **eventType**

  The type of the event

  Value: LOGON

- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS

- **timestamp**

  The UTC timestamp when the event was created

- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The user id

  - **uniqueId**

    The unique identifier of the user
  - **userType**

    The user type

    Example Values: API, UI
  - **userManagementProvider**

    The user management provider

    Example Values: INTERNAL, LDAP, SSO_SAML
  - **superuser**

    Whether or not the user is a superuser
  - **logonPermitted**

    Whether or not the user is allowed to log on
  - **objectType**

    The object type
  - **enabled**

    Whether or not the user is enabled
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.18. METRIC

This event is published at a fixed interval and contains information about different metrics

Event Fields:

- **eventType**

  The type of the event

  Value: METRIC
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

- **metricName**

  The name of the metric

  Example Values: JDBC_VAULT_NUM_CONNECTIONS, JDBC_VAULT_NUM_BUSY_CONNECTIONS

  See the 'List of Metric Types' section for more details
- **unit**

  The unit of the metric
- **metadata**

  The metadata that accompanies the metric

  Sub-fields:

  - **jobId**

    The id of the job the metric pertains to
  - **processorId**

    The id of the data flow processor that this metric pertains to
  - **ruleId**

    The id of the rule the metric pertains to
  - **environmentId**

    The id of the environment
  - **jobType**

    The type of the Job

    Example Values: BATCH, POD, DATAFLOW
  - **pipelineId**

    The id of the data flow pipeline
  - **tokenizationBatchSize**

    The maximum batch size when processing records
  - **tokenizationThreadCount**

    The maximum number of threads to use when processing records
  - **vaultType**

    The token vault type associated with the job

    Example Values: HDFS, JDBC, NONE
  - **vaultDbEngine**

    The token vault database engine, if one has been set up

    Example Values: Oracle, Postgres, DynamoDb, HBase
  - **vaultDbVersion**

    The token vault database version, if one has been set up
  - **vaultConfigProperties**

    Other configuration properties of the token vault associated with the job. Note: sensitive properties will be redacted

- **diskId**

  The volume id for the disk
- **numberOfColumns**

  The number of columns in the input data
- **numberOfOptionalColumns**

  The number of optional columns in the input data

- **gaugeValues**

  A list of gauge values

  Sub-fields:

  - **specificMetricMetadataValues**

    The metadata associated with this gauge

    Sub-fields:

    - **columnId**

      The column id
    - **ruleId**

      The rule id
  - **value**

    The value of the gauge
- **histogramValues**

  A list of histogram details.

  Sub-fields:

  - **specificMetricMetadataValues**

    The metadata associated with this histogram

    Sub-fields:

    - **columnId**

      The column id
    - **ruleId**

      The rule id
  - **snapshotCount**

    The number of values contained in the histogram snapshot
  - **maxValue**

    The largest value in the histogram snapshot
  - **minValue**

    The smallest value in the histogram snapshot
  - **medianValue**

    The median value in the histogram snapshot
  - **meanValue**

    The mean value in the histogram snapshot

        - **stdev**

          The standard deviation of values in the histogram snapshot
        - **seventyFifthPercentile**

          The seventy fifth percentile of values in the histogram snapshot
        - **ninetyFifthPercentile**

          The ninety fifth percentile of values in the histogram snapshot
        - **ninetyEighthPercentile**

          The ninety eighth percentile of values in the histogram snapshot
        - **ninetyNinthPercentile**

          The ninety ninth percentile of values in the histogram snapshot
        - **ninetyNinePointNinthPercentile**

          The ninety nine point ninth percentile of values in the histogram snapshot
    - **meter**

      The details of the rate of values

      Sub-fields:

        - **count**

          The number of values recorded
        - **meanRate**

          The mean rate of events over time
        - **oneMinuteRate**

          The moving average rate of events over the last one minute
        - **fiveMinuteRate**

          The moving average rate of events over the last five minutes
        - **fifteenMinuteRate**

          The moving average rate of events over the last fifteen minutes
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.19. PDD_CLOSED

This event is published whenever a Protected Data Domain is closed

Event Fields:

- **eventType**

  The type of the event

  Value: PDD_CLOSED

- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The protected data domain id
  - **uniqueId**

    The unique id of the protected data domain
  - **team**

    The team of the protected data domain creator

    See the parameters field of the 'TEAM_CREATED' Event for more details
  - **watermarked**

    Whether or not the protected data domain is watermarked
  - **objectType**

    The object type
  - **environment**

    The details of the protected data domain environment

    See the parameters field of the 'ENVIRONMENT_CREATED' Event for more details
  - **metadata**

    The metadata of the protected data domain
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.20. PDD_CREATED

This event is published whenever a new Protected Data Domain is created

Event Fields:

- **eventType**

  The type of the event

  Value: PDD_CREATED

- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:
  - **id**

    The protected data domain id
  - **uniqueId**

    The unique id of the protected data domain
  - **team**

    The team of the protected data domain creator

    See the parameters field of the 'TEAM_CREATED' Event for more details
  - **watermarked**

    Whether or not the protected data domain is watermarked
  - **objectType**

    The object type
  - **environment**

    The details of the protected data domain environment

    See the parameters field of the 'ENVIRONMENT_CREATED' Event for more details
  - **metadata**

    The metadata of the protected data domain
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.21. PDD_DELETED

This event is published whenever a Protected Data Domain is deleted

Event Fields:

- **eventType**

  The type of the event

  Value: PDD_DELETED

- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The protected data domain id
  - **uniqueId**

    The unique id of the protected data domain
  - **team**

    The team of the protected data domain creator

    See the parameters field of the 'TEAM_CREATED' Event for more details
  - **watermarked**

    Whether or not the protected data domain is watermarked
  - **objectType**

    The object type
  - **environment**

    The details of the protected data domain environment

    See the parameters field of the 'ENVIRONMENT_CREATED' Event for more details
  - **metadata**

    The metadata of the protected data domain
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.22. PDD_UPDATED

This event is published whenever a Protected Data Domain is updated

Event Fields:

- **eventType**

  The type of the event

  Value: PDD_UPDATED

- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The protected data domain id
  - **uniqueId**

    The unique id of the protected data domain
  - **team**

    The team of the protected data domain creator

    See the parameters field of the 'TEAM_CREATED' Event for more details
  - **watermarked**

    Whether or not the protected data domain is watermarked
  - **objectType**

    The object type
  - **environment**

    The details of the protected data domain environment

    See the parameters field of the 'ENVIRONMENT_CREATED' Event for more details
  - **metadata**

    The metadata of the protected data domain
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.23. POLICY_CREATED

This event is published whenever a new policy is created

Event Fields:

- **eventType**

  The type of the event

  Value: POLICY_CREATED

- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **policy**

    The parameters of the policy

    Sub-fields:

    - **id**

      The policy id
    - **uniqueId**

      The unique id of the policy
    - **team**

      The team of the policy creator

      See the parameters field of the 'TEAM_CREATED' Event for more details
    - **creatorId**

      The id of the policy creator
    - **deleted**

      Whether or not the policy is deleted
    - **objectType**

      The object type
    - **schema**

      The parameters of the schema associated with this policy

      See the parameters field of the 'SCHEMA_CREATED' Event for more details
    - **autoGeneralisationParams**

      The details of any automatic generalisation operations

      Sub-fields:

      - **requiredAutoK**

        The minimum cluster size
      - **requiredAutoL**

        The L diversity
      - **requiredAutoC**

        The C ratio
      - **requiredLBins**

        The number of L bins

- **quasiIdentifyingAutoGenColumns**

  The quasi identifying columns

  Sub-fields:

  - **id**

    The column id

  - **uniqueId**

    The column's unique id

  - **dataType**

    The type of data in the column

    Example Values: Text, Boolean, Float, Double

  - **optional**

    Whether the data is always present in the column

  - **pathElements**

    The hierarchical schema path elements

    Sub-fields:

    - **index**

      The path element index

    - **type**

      The path element type

      Example Values: ARRAY, STRUCT

- **autoGenSensitiveColumns**

  The sensitive columns

  Sub-fields:

  - **id**

    The column id

  - **uniqueId**

    The column's unique id

  - **dataType**

    The type of data in the column

    Example Values: Text, Boolean, Float, Double

  - **optional**

    Whether the data is always present in the column

  - **pathElements**

    The hierarchical schema path elements

    Sub-fields:

    - **index**

      The path element index

        - **type**

          The path element type

          Example Values: ARRAY, STRUCT

  - **autoGenInterestingColumns**

    The interesting columns

    Sub-fields:

    - **id**

      The column id

    - **uniqueId**

      The column's unique id

    - **dataType**

      The type of data in the column

      Example Values: Text, Boolean, Float, Double

    - **optional**

      Whether the data is always present in the column

    - **pathElements**

      The hierarchical schema path elements

      Sub-fields:

      - **index**

        The path element index

      - **type**

        The path element type

        Example Values: ARRAY, STRUCT

  - **autoGenConstraints**

    The generalisation constraints

    See the parameters field of the 'RULE_CREATED' Event for more details

- **manualGeneralisationParams**

  The details of any manual generalisation operations

  Sub-fields:

  - **requiredManualK**

    The minimum cluster size

  - **quasiIdentifyingManualGenColumns**

    The quasi identifying columns

    Sub-fields:

    - **id**

      The column id

    - **uniqueId**

      The column's unique id

        - **dataType**

          The type of data in the column

          Example Values: Text, Boolean, Float, Double
        - **optional**

          Whether the data is always present in the column
        - **pathElements**

          The hierarchical schema path elements

          Sub-fields:

          - **index**

            The path element index
          - **type**

            The path element type

            Example Values: ARRAY, STRUCT
    - **manualGenConstraints**

      The generalisation constraints

      [See the parameters field of the 'RULE_CREATED' Event for more details]
- **masks**

  The details of any masking operations

  Sub-fields:

  - **rule**

    The parameters of this rule

    [See the parameters field of the 'RULE_CREATED' Event for more details]
  - **columns**

    The columns that this rule are applied to in the parent policy

    Sub-fields:

    - **id**

      The column id
    - **uniqueId**

      The column's unique id
    - **dataType**

      The type of data in the column

      Example Values: Text, Boolean, Float, Double
    - **optional**

      Whether the data is always present in the column
    - **pathElements**

      The hierarchical schema path elements

      Sub-fields:

        - **index**

          The path element index
        - **type**

          The path element type

          Example Values: ARRAY, STRUCT
    - **originalMaskingPolicy**

      The parameters of the original masking policy

      See the policy field inside the parameters field of the 'POLICY_CREATED' Event for more details
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.24. POLICY_DELETED

This event is published whenever a policy is deleted

Event Fields:

- **eventType**

  The type of the event

  Value: POLICY_DELETED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:
    - **policy**

      The parameters of the policy

      Sub-fields:
        - **id**

          The policy id
        - **uniqueId**

          The unique id of the policy

- **team**

  The team of the policy creator

  See the parameters field of the 'TEAM_CREATED' Event for more details
- **creatorId**

  The id of the policy creator
- **deleted**

  Whether or not the policy is deleted
- **objectType**

  The object type
- **schema**

  The parameters of the schema associated with this policy

  See the parameters field of the 'SCHEMA_CREATED' Event for more details
- **autoGeneralisationParams**

  The details of any automatic generalisation operations

  Sub-fields:

  - **requiredAutoK**

    The minimum cluster size
  - **requiredAutoL**

    The L diversity
  - **requiredAutoC**

    The C ratio
  - **requiredLBins**

    The number of L bins
  - **quasiIdentifyingAutoGenColumns**

    The quasi identifying columns

    Sub-fields:

    - **id**

      The column id
    - **uniqueId**

      The column's unique id
    - **dataType**

      The type of data in the column

      Example Values: Text, Boolean, Float, Double
    - **optional**

      Whether the data is always present in the column
    - **pathElements**

      The hierarchical schema path elements

      Sub-fields:

    - **index**

      The path element index
    - **type**

      The path element type

      Example Values: ARRAY, STRUCT
- **autoGenSensitiveColumns**

  The sensitive columns

  Sub-fields:
  - **id**

    The column id
  - **uniqueId**

    The column's unique id
  - **dataType**

    The type of data in the column

    Example Values: Text, Boolean, Float, Double
  - **optional**

    Whether the data is always present in the column
  - **pathElements**

    The hierarchical schema path elements

    Sub-fields:
    - **index**

      The path element index
    - **type**

      The path element type

      Example Values: ARRAY, STRUCT
- **autoGenInterestingColumns**

  The interesting columns

  Sub-fields:
  - **id**

    The column id
  - **uniqueId**

    The column's unique id
  - **dataType**

    The type of data in the column

    Example Values: Text, Boolean, Float, Double
  - **optional**

    Whether the data is always present in the column

        - **pathElements**

          The hierarchical schema path elements

          Sub-fields:

          - **index**

            The path element index

          - **type**

            The path element type

            Example Values: ARRAY, STRUCT

    - **autoGenConstraints**

      The generalisation constraints

      See the parameters field of the 'RULE_CREATED' Event for more details

- **manualGeneralisationParams**

  The details of any manual generalisation operations

  Sub-fields:

  - **requiredManualK**

    The minimum cluster size

  - **quasiIdentifyingManualGenColumns**

    The quasi identifying columns

    Sub-fields:

    - **id**

      The column id

    - **uniqueId**

      The column's unique id

    - **dataType**

      The type of data in the column

      Example Values: Text, Boolean, Float, Double

    - **optional**

      Whether the data is always present in the column

    - **pathElements**

      The hierarchical schema path elements

      Sub-fields:

      - **index**

        The path element index

      - **type**

        The path element type

        Example Values: ARRAY, STRUCT

  - **manualGenConstraints**

    The generalisation constraints

        See the parameters field of the 'RULE_CREATED' Event for more details

    - **masks**

      The details of any masking operations

      Sub-fields:

      - **rule**

        The parameters of this rule

        See the parameters field of the 'RULE_CREATED' Event for more details

      - **columns**

        The columns that this rule are applied to in the parent policy

        Sub-fields:

        - **id**

          The column id

        - **uniqueId**

          The column's unique id

        - **dataType**

          The type of data in the column

          Example Values: Text, Boolean, Float, Double

        - **optional**

          Whether the data is always present in the column

        - **pathElements**

          The hierarchical schema path elements

          Sub-fields:

          - **index**

            The path element index

          - **type**

            The path element type

            Example Values: ARRAY, STRUCT

  - **originalMaskingPolicy**

    The parameters of the original masking policy

    See the policy field inside the parameters field of the 'POLICY_CREATED' Event for more details

- **uuid**

  The event unique identifier

- **applicationVersion**

  The version of the Privitar application sending the event

- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.25. POLICY_MANAGER_STARTED

This event is published whenever policy manager starts up

Event Fields:

- **eventType**

  The type of the event

  Value: POLICY_MANAGER_STARTED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **jdkVersion**

    The current JDK version
  - **databaseEngine**

    The name of the configuration database engine

    Example Values: postgres, oracle, mysql
  - **databaseVersion**

    The database version
  - **ssoProvider**

    The name of the single sign-on (SSO) provider, if SSO is enabled.
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.26. POLICY_UPDATED

This event is published whenever a policy is updated

Event Fields:

- **eventType**

  The type of the event

  Value: POLICY_UPDATED

- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **policy**

    The parameters of the policy

    Sub-fields:

    - **id**

      The policy id
    - **uniqueId**

      The unique id of the policy
    - **team**

      The team of the policy creator

      See the parameters field of the 'TEAM_CREATED' Event for more details
    - **creatorId**

      The id of the policy creator
    - **deleted**

      Whether or not the policy is deleted
    - **objectType**

      The object type
    - **schema**

      The parameters of the schema associated with this policy

      See the parameters field of the 'SCHEMA_CREATED' Event for more details
    - **autoGeneralisationParams**

      The details of any automatic generalisation operations

      Sub-fields:

      - **requiredAutoK**

        The minimum cluster size
      - **requiredAutoL**

        The L diversity
      - **requiredAutoC**

        The C ratio
      - **requiredLBins**

        The number of L bins

- **quasiIdentifyingAutoGenColumns**

  The quasi identifying columns

  Sub-fields:

  - **id**

    The column id

  - **uniqueId**

    The column's unique id

  - **dataType**

    The type of data in the column

    Example Values: Text, Boolean, Float, Double

  - **optional**

    Whether the data is always present in the column

  - **pathElements**

    The hierarchical schema path elements

    Sub-fields:

    - **index**

      The path element index

    - **type**

      The path element type

      Example Values: ARRAY, STRUCT

- **autoGenSensitiveColumns**

  The sensitive columns

  Sub-fields:

  - **id**

    The column id

  - **uniqueId**

    The column's unique id

  - **dataType**

    The type of data in the column

    Example Values: Text, Boolean, Float, Double

  - **optional**

    Whether the data is always present in the column

  - **pathElements**

    The hierarchical schema path elements

    Sub-fields:

    - **index**

      The path element index

        - **type**

          The path element type

          Example Values: ARRAY, STRUCT

  - **autoGenInterestingColumns**

    The interesting columns

    Sub-fields:

    - **id**

      The column id

    - **uniqueId**

      The column's unique id

    - **dataType**

      The type of data in the column

      Example Values: Text, Boolean, Float, Double

    - **optional**

      Whether the data is always present in the column

    - **pathElements**

      The hierarchical schema path elements

      Sub-fields:

      - **index**

        The path element index

      - **type**

        The path element type

        Example Values: ARRAY, STRUCT

  - **autoGenConstraints**

    The generalisation constraints

    See the parameters field of the 'RULE_CREATED' Event for more details

- **manualGeneralisationParams**

  The details of any manual generalisation operations

  Sub-fields:

  - **requiredManualK**

    The minimum cluster size

  - **quasiIdentifyingManualGenColumns**

    The quasi identifying columns

    Sub-fields:

    - **id**

      The column id

    - **uniqueId**

      The column's unique id

        - **dataType**

          The type of data in the column

          Example Values: Text, Boolean, Float, Double

        - **optional**

          Whether the data is always present in the column

        - **pathElements**

          The hierarchical schema path elements

          Sub-fields:

            - **index**

              The path element index

            - **type**

              The path element type

              Example Values: ARRAY, STRUCT

    - **manualGenConstraints**

      The generalisation constraints

      See the parameters field of the 'RULE_CREATED' Event for more details

- **masks**

  The details of any masking operations

  Sub-fields:

    - **rule**

      The parameters of this rule

      See the parameters field of the 'RULE_CREATED' Event for more details

    - **columns**

      The columns that this rule are applied to in the parent policy

      Sub-fields:

        - **id**

          The column id

        - **uniqueId**

          The column's unique id

        - **dataType**

          The type of data in the column

          Example Values: Text, Boolean, Float, Double

        - **optional**

          Whether the data is always present in the column

        - **pathElements**

          The hierarchical schema path elements

          Sub-fields:

        - **index**

          The path element index
        - **type**

          The path element type

          Example Values: ARRAY, STRUCT
    - **originalMaskingPolicy**

      The parameters of the original masking policy

      See the policy field inside the parameters field of the 'POLICY_CREATED' Event for more details
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.27. PPO

This event is published whenever a job that performs Privacy Preserving Operations is run. A Privacy Preserving Operation is any operation that is not a retain or drop.

Event Fields:

- **eventType**

  The type of the event

  Value: PPO
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

    - **numberOfPpos**

      The total number of privacy preserving operations performed
    - **jobId**

      The job id
    - **jobType**

      The job type

      Example Values: BATCH, POD, DATAFLOW

- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.28. PROCESSOR_START_UP

This event is published when the spark processor is started

Event Fields:

- **eventType**

  The type of the event

  Value: PROCESSOR_START_UP
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **javaVersion**

    The version of java that the processor is running in
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.29. RECORDS_PROCESSED

This event is published when a job is run, and it contains details about the number of records processed.

Event Fields:

- **eventType**

  The type of the event

  Value: RECORDS_PROCESSED

- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **numberOfRecordsProcessed**

    The total number of records processed
  - **jobId**

    The job id
  - **jobType**

    The job type

    Example Values: BATCH, POD, DATAFLOW
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.30. REMOVE_TOKEN_MAPPING_FINISHED

This event is published when a request to forget a user's identifying value has finished processing

Event Fields:

- **eventType**

  The type of the event

  Value: REMOVE_TOKEN_MAPPING_FINISHED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **pddRunUniqueId**

    The unique id of the protected data domain run. In the case of removing token mappings, this is also the id of the removal request, and is used to send the progress heartbeats to Policy Manager
  - **ruleUniqueId**

    The id of the rule to remove the value from
  - **rule**

    The rule to remove the value from

    See the parameters field of the 'RULE_CREATED' Event for more details
  - **environment**

    The token vault's environment

    See the parameters field of the 'ENVIRONMENT_CREATED Event for more details
  - **deploymentId**

    The token vault's deployment id
  - **pddUniqueId**

    The id of the PDD to remove the value from
  - **pdd**

    The PDD to remove the value from

    See the parameters field of the 'PDD_CREATED' Event for more details
  - **userUniqueId**

    The id of the user that started the remove token mapping request.
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.31. REMOVE_TOKEN_MAPPING_STARTED

This event is published when there is a new request to forget a user's identifying value

Event Fields:

- **eventType**

  The type of the event

  Value: REMOVE_TOKEN_MAPPING_STARTED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

The UTC timestamp when the event was created

- **parameters**

  The event parameters

  Sub-fields:

  - **pddRunUniqueId**

    The unique id of the protected data domain run. In the case of removing token mappings, this is also the id of the removal request, and is used to send the progress heartbeats to Policy Manager

  - **ruleUniqueId**

    The id of the rule to remove the value from

  - **rule**

    The rule to remove the value from

    See the parameters field of the 'RULE_CREATED' Event for more details

  - **environment**

    The token vault's environment

    See the parameters field of the 'ENVIRONMENT_CREATED Event for more details

  - **deploymentId**

    The token vault's deployment id

  - **pddUniqueId**

    The id of the PDD to remove the value from

  - **pdd**

    The PDD to remove the value from

    See the parameters field of the 'PDD_CREATED' Event for more details

  - **userUniqueId**

    The id of the user that started the remove token mapping request.

- **uuid**

  The event unique identifier

- **applicationVersion**

  The version of the Privitar application sending the event

- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.32. RULE_CREATED

This event is published whenever a new rule is created

Event Fields:

- **eventType**

  The type of the event

  Value: RULE_CREATED

- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The rule id
  - **uniqueId**

    The unique identifier of the rule
  - **team**

    The team of the rule creator

    See the parameters field of the 'TEAM_CREATED' Event for more details
  - **rule**

    The rule type

    Example Values: RETAIN, DROP_COLUMN
  - **consistent**

    Whether or not the rule is consistent
  - **retainNull**

    Whether or not the rule is retains null
  - **reversible**

    Whether or not the rule is reversible
  - **sharedWithAllTeams**

    Whether or not the rule is shared with all teams
  - **objectType**

    The object type
  - **properties**

    The rule properties. Note: Sensitive properties will be redacted.
  - **sharees**

    The list of teams that share this rule

    See the parameters field of the 'TEAM_CREATED' Event for more details
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

The instanceId that emitted the event. Used to attribute events from the same service

17.6.33. RULE_DELETED

This event is published whenever a rule is deleted

Event Fields:

- **eventType**

  The type of the event

  Value: RULE_DELETED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The rule id
  - **uniqueId**

    The unique identifier of the rule
  - **team**

    The team of the rule creator

    See the parameters field of the 'TEAM_CREATED' Event for more details
  - **rule**

    The rule type

    Example Values: RETAIN, DROP_COLUMN
  - **consistent**

    Whether or not the rule is consistent
  - **retainNull**

    Whether or not the rule is retains null
  - **reversible**

    Whether or not the rule is reversible
  - **sharedWithAllTeams**

    Whether or not the rule is shared with all teams
  - **objectType**

    The object type
  - **properties**

    The rule properties. Note: Sensitive properties will be redacted.

  - **sharees**

    The list of teams that share this rule

    See the parameters field of the 'TEAM_CREATED' Event for more details
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.34. RULE_PREVIEWED

This event is published whenever a rule is previewed

Event Fields:

- **eventType**

  The type of the event

  Value: RULE_PREVIEWED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The rule id
  - **uniqueId**

    The unique identifier of the rule
  - **team**

    The team of the rule creator

    See the parameters field of the 'TEAM_CREATED' Event for more details
  - **rule**

    The rule type

    Example Values: RETAIN, DROP_COLUMN
  - **consistent**

    Whether or not the rule is consistent
  - **retainNull**

  Whether or not the rule is retains null

  - **reversible**

    Whether or not the rule is reversible

  - **sharedWithAllTeams**

    Whether or not the rule is shared with all teams

  - **objectType**

    The object type

  - **properties**

    The rule properties. Note: Sensitive properties will be redacted.

  - **sharees**

    The list of teams that share this rule

    See the parameters field of the 'TEAM_CREATED' Event for more details

- **uuid**

  The event unique identifier

- **applicationVersion**

  The version of the Privitar application sending the event

- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.35. RULE_UPDATED

This event is published whenever a rule is updated

Event Fields:

- **eventType**

  The type of the event

  Value: RULE_UPDATED

- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS

- **timestamp**

  The UTC timestamp when the event was created

- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The rule id

  - **uniqueId**

    The unique identifier of the rule

  - **team**

    The team of the rule creator

    See the parameters field of the 'TEAM_CREATED' Event for more details
  - **rule**

    The rule type

    Example Values: RETAIN, DROP_COLUMN
  - **consistent**

    Whether or not the rule is consistent
  - **retainNull**

    Whether or not the rule is retains null
  - **reversible**

    Whether or not the rule is reversible
  - **sharedWithAllTeams**

    Whether or not the rule is shared with all teams
  - **objectType**

    The object type
  - **properties**

    The rule properties. Note: Sensitive properties will be redacted.
  - **sharees**

    The list of teams that share this rule

    See the parameters field of the 'TEAM_CREATED' Event for more details
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.36. SCHEMA_CREATED

This event is synthesized whenever a new schema is created

Event Fields:

- **eventType**

  The type of the event

  Value: SCHEMA_CREATED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS

- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The schema id
  - **uniqueId**

    The unique identifier of the schema
  - **team**

    The team of the schema creator

    [See the parameters field of the 'TEAM_CREATED' Event for more details]()
  - **creatorId**

    The id of the schema creator
  - **inferred**

    Whether or not the schema was inferred
  - **objectType**

    The object type
  - **tables**

    The list of tables in the schema

    Sub-fields:

    - **id**

      The table id
    - **uniqueId**

      The table's unique id
    - **fields**

      The columns in the table

      Sub-fields:

      - **id**

        The column id
      - **uniqueId**

        The column's unique id
      - **dataType**

        The type of data in the column

        Example Values: Text, Boolean, Float, Double
      - **optional**

        Whether the data is always present in the column
      - **pathElements**

The hierarchical schema path elements

Sub-fields:

- **index**

  The path element index
- **type**

  The path element type

  Example Values: ARRAY, STRUCT

- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.37. SCHEMA_DELETED

This event is published whenever a schema is deleted

Event Fields:

- **eventType**

  The type of the event

  Value: SCHEMA_DELETED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The schema id
  - **uniqueId**

    The unique identifier of the schema
  - **team**

    The team of the schema creator

    See the parameters field of the 'TEAM_CREATED' Event for more details
  - **creatorId**

    The id of the schema creator

  - **inferred**

    Whether or not the schema was inferred
  - **objectType**

    The object type
  - **tables**

    The list of tables in the schema

    Sub-fields:

    - **id**

      The table id
    - **uniqueId**

      The table's unique id
    - **fields**

      The columns in the table

      Sub-fields:

      - **id**

        The column id
      - **uniqueId**

        The column's unique id
      - **dataType**

        The type of data in the column

        Example Values: Text, Boolean, Float, Double
      - **optional**

        Whether the data is always present in the column
      - **pathElements**

        The hierarchical schema path elements

        Sub-fields:

        - **index**

          The path element index
        - **type**

          The path element type

          Example Values: ARRAY, STRUCT
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.38. SCHEMA_UPDATED

This event is published whenever a schema is updated

Event Fields:

- **eventType**

  The type of the event

  Value: SCHEMA_UPDATED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The schema id
  - **uniqueId**

    The unique identifier of the schema
  - **team**

    The team of the schema creator

    See the parameters field of the 'TEAM_CREATED' Event for more details
  - **creatorId**

    The id of the schema creator
  - **inferred**

    Whether or not the schema was inferred
  - **objectType**

    The object type
  - **tables**

    The list of tables in the schema

    Sub-fields:

    - **id**

      The table id
    - **uniqueId**

      The table's unique id
    - **fields**

      The columns in the table

      Sub-fields:

        - **id**

          The column id
        - **uniqueId**

          The column's unique id
        - **dataType**

          The type of data in the column

          Example Values: Text, Boolean, Float, Double
        - **optional**

          Whether the data is always present in the column
        - **pathElements**

          The hierarchical schema path elements

          Sub-fields:

          - **index**

            The path element index
          - **type**

            The path element type

            Example Values: ARRAY, STRUCT

- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.39. SERVICES_STATUS

This event is published frequently with information about the current state of connected Privitar services

Event Fields:

- **eventType**

  The type of the event

  Value: SERVICES_STATUS
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **numberOfServices**

    The total number of currently connected services
  - **numberOfHealthyServices**

    The total number of currently healthy services
  - **services**

    A list of other details about the currently connected services

    See the parameters field of the 'SERVICE_STATUS_CHANGED' Event for more details
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.40. SERVICE_STATUS_CHANGED

This event is published whenever a Privitar Service has a new health status

Event Fields:

- **eventType**

  The type of the event

  Value: SERVICE_STATUS_CHANGED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **status**

    The current health status of this service

    Example Values: SERVICE_HEALTHY, SERVICE_UNAVAILABLE
  - **integrationId**

    The id of this service
  - **integrationType**

    The service integration type

    Example Values: NIFI, KAFKA, POD

- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.41. STEP_REPORT

This event is published when a job completes, and contains information about the job's performance from the step reports

Event Fields:

- **eventType**

  The type of the event

  Value: STEP_REPORT
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **fileDataSourceStepReportParameters**

    The details from the file data source step report

    Sub-fields:

    - **commonParameters**

      Parameters common to all step report types

      Sub-fields:

      - **startTime**

        The time at which the step began
      - **endTime**

        The time at which the step finished
      - **errorCount**

        The number of errors that occurred during the step
    - **commonDataSourceParameters**

      Parameters common to all data source step report types

      Sub-fields:

        - **goodRowCount**

          The total number of good rows across all input files
        - **badRowCount**

          The total number of bad rows across all input files
        - **fileSizeTotal**

          The total size in bytes of all input files
    - **fileCountPerTable**

      A list of the number of files in each table
    - **memorySizePerTable**

      A list of the memory size in bytes of all files in each table
    - **totalCorruptedRows**

      The total number of corrupted rows across all tables
    - **totalInvalidColumnCounts**

      The total number of invalid columns across all tables
    - **totalErrorsPerFile**

      A list of the number of errors per file
- **hiveDataSourceStepReportParameters**

  The details from the hive data source step report

  Sub-fields:
    - **commonParameters**

      Parameters common to all step report types

      Sub-fields:
        - **startTime**

          The time at which the step began
        - **endTime**

          The time at which the step finished
        - **errorCount**

          The number of errors that occurred during the step
    - **commonDataSourceParameters**

      Parameters common to all data source step report types

      Sub-fields:
        - **goodRowCount**

          The total number of good rows across all input files
        - **badRowCount**

          The total number of bad rows across all input files
        - **fileSizeTotal**

          The total size in bytes of all input files
    - **filterTypeOccurrenceCount**

    The number of times each filter type was used in the job run

- **filePartitionStepReportParameters**

  The details of the file partitioning step report

  Sub-fields:

  - **commonParameters**

    Parameters common to all step report types

    Sub-fields:

    - **startTime**

      The time at which the step began
    - **endTime**

      The time at which the step finished
    - **errorCount**

      The number of errors that occurred during the step
  - **commonPartitionParameters**

    Parameters common to all partitioning step report types

    Sub-fields:

    - **totalPartitionCount**

      The total number of partitions across all files
    - **totalOriginalPartitionCount**

      The total number of original partitions across all files
    - **repartitionSize**

      The size in bytes after repartitioning
  - **totalOriginalPartitionSize**

    The total size of all original partitions
- **hivePartitionStepReportParameters**

  The details of the hive partitioning step report

  Sub-fields:

  - **commonParameters**

    Parameters common to all step report types

    Sub-fields:

    - **startTime**

      The time at which the step began
    - **endTime**

      The time at which the step finished
    - **errorCount**

      The number of errors that occurred during the step
  - **commonPartitionParameters**

    Parameters common to all partitioning step report types

    Sub-fields:

    - **totalPartitionCount**

      The total number of partitions across all files

    - **totalOriginalPartitionCount**

      The total number of original partitions across all files

    - **repartitionSize**

      The size in bytes after repartitioning

- **maskingStepReportParameters**

  The details of the masking step report

  Sub-fields:

  - **commonParameters**

    Parameters common to all step report types

    Sub-fields:

    - **startTime**

      The time at which the step began

    - **endTime**

      The time at which the step finished

    - **errorCount**

      The number of errors that occurred during the step

  - **tableReportParameters**

    Report information for each table processed

    Sub-fields:

    - **goodRowCount**

      The number of good rows in the table

    - **badRowCount**

      The number of bad rows in the table

    - **columnParameters**

      Report information for each column in the table

      Sub-fields:

      - **badRowCount**

        The number of bad rows in the column

      - **nullRowCount**

        The number of null rows in the column

      - **goodRowCount**

        The number of good rows in the column

      - **ruleType**

        The type of the rule applied to the column

  - **vaultReportParameters**

    Report information for each each vault used during processing

    Sub-fields:

    - **ruleId**

      The id of the rule the vault applies to
    - **ruleType**

      The type of the rule the vault applies to
    - **vaultItemCount**

      The number of items in the vault
    - **vaultSizeBytes**

      The size of the vault in bytes
- **unmaskingStepReportParameters**

  The details of the unmasking step report

  Sub-fields:

  - **commonParameters**

    Parameters common to all step report types

    Sub-fields:

    - **startTime**

      The time at which the step began
    - **endTime**

      The time at which the step finished
    - **errorCount**

      The number of errors that occurred during the step
  - **tableReportParameters**

    Report information for each table processed

    Sub-fields:

    - **goodRowCount**

      The number of good rows in the table
    - **badRowCount**

      The number of bad rows in the table
    - **columnParameters**

      Report information for each column in the table

      Sub-fields:

      - **badRowCount**

        The number of bad rows in the column
      - **nullRowCount**

        The number of null rows in the column

    - **goodRowCount**

      The number of good rows in the column
    - **ruleType**

      The type of the rule applied to the column
- **manualGeneralisationStepReportParameters**

  The details of the manual generalisation step report

  Sub-fields:
  - **commonParameters**

    Parameters common to all step report types

    Sub-fields:
    - **startTime**

      The time at which the step began
    - **endTime**

      The time at which the step finished
    - **errorCount**

      The number of errors that occurred during the step
  - **tableReportParameters**

    Report information for each table processed

    Sub-fields:
    - **goodRowCount**

      The number of good rows in the table
    - **badRowCount**

      The number of bad rows in the table
    - **columnParameters**

      Report information for each column in the table

      Sub-fields:
      - **badRowCount**

        The number of bad rows in the column
      - **nullRowCount**

        The number of null rows in the column
      - **goodRowCount**

        The number of good rows in the column
      - **ruleType**

        The type of the rule applied to the column
  - **rowsRetained**

    The number of rows retained
  - **rowsDropped**

    The number of rows dropped

- **automaticGeneralisationStepReportParameters**

  The details of the automatic generalisation step report

  Sub-fields:

  - **commonParameters**

    Parameters common to all step report types

    Sub-fields:

    - **startTime**

      The time at which the step began

    - **endTime**

      The time at which the step finished

    - **errorCount**

      The number of errors that occurred during the step

  - **tableReportParameters**

    Report information for each table processed

    Sub-fields:

    - **ruleTypes**

      A list of all rule types used in the table

    - **rowCount**

      The number of rows in the table

  - **rowCount**

    The number of rows processed

- **dataSinkStepReportParameters**

  The details of the output step report

  Sub-fields:

  - **commonParameters**

    Parameters common to all step report types

    Sub-fields:

    - **startTime**

      The time at which the step began

    - **endTime**

      The time at which the step finished

    - **errorCount**

      The number of errors that occurred during the step

  - **totalOutputFileSize**

    The total size in bytes of all output files

  - **totalOutputRows**

    The total number of rows across all output files

- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.42. SYSTEM_INFORMATION

This event is published on start up and contains information about the system the service is running on

Event Fields:

- **eventType**

  The type of the event

  Value: SYSTEM_INFORMATION
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **memoryMb**

    Total installed memory
  - **cpuCores**

    Number of cores
  - **cpuClockSpeed**

    Max clock speed of each core
  - **diskSpaceMb**

    Size of the disks on the machine
  - **operatingSystem**

    Operating System running the application
  - **maxOpenFileDescriptors**

    maximum open file descriptors
  - **instanceId**

    The instance Id that published the event
- **uuid**

  The event unique identifier

- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.43. SYSTEM_USER_CREATED

This event is published whenever a new system user is created

Event Fields:

- **eventType**

  The type of the event

  Value: SYSTEM_USER_CREATED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The system user id
  - **uniqueId**

    The unique identifier of the system user
  - **deleted**

    Whether or not the system user has been deleted
  - **objectType**

    The object type
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.44. SYSTEM_USER_DELETED

This event is published whenever a system user is deleted

Event Fields:

- **eventType**

  The type of the event

  Value: SYSTEM_USER_DELETED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The system user id
  - **uniqueId**

    The unique identifier of the system user
  - **deleted**

    Whether or not the system user has been deleted
  - **objectType**

    The object type
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.45. SYSTEM_USER_UPDATED

This event is published whenever a system user is updated

Event Fields:

- **eventType**

  The type of the event

  Value: SYSTEM_USER_UPDATED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created

- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The system user id
  - **uniqueId**

    The unique identifier of the system user
  - **deleted**

    Whether or not the system user has been deleted
  - **objectType**

    The object type
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.46. TEAM_CREATED

This event is published whenever a new team is created

Event Fields:

- **eventType**

  The type of the event

  Value: TEAM_CREATED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The team id
  - **uniqueId**

    The unique id of the team
  - **objectType**

  The object type

- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.47. TEAM_DELETED

This event is published whenever a team is deleted

Event Fields:

- **eventType**

  The type of the event

  Value: TEAM_DELETED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The team id
  - **uniqueId**

    The unique id of the team
  - **objectType**

    The object type
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.48. TEAM_DELETION_REQUEST

This event is published whenever a request to delete a team is made

Event Fields:

- **eventType**

  The type of the event

  Value: TEAM_DELETION_REQUEST
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **uniqueId**

    The unique id of the team
  - **numberOfPolicies**

    The number of policies owned by the team at the time of the request
  - **numberOfSchemas**

    The number of schemas owned by the team at the time of the request
  - **numberOfRules**

    The number of rules owned by the team at the time of the request
  - **numberOfEnvironments**

    The number of environments owned by the team at the time of the request
  - **numberOfInvestigations**

    The number of investigations owned by the team at the time of the request
  - **intent**

    The intent of the deletion request. Either to validate or commit the deletion

    Example Values: COMMIT, VALIDATE
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.49. TEAM_MEMBERSHIP_ADDED

This event is published whenever a new team member is added

Event Fields:

- **eventType**

  The type of the event

Value: TEAM_MEMBERSHIP_ADDED

- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS

- **timestamp**

  The UTC timestamp when the event was created

- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The team id

  - **uniqueId**

    The unique id of the team

  - **objectType**

    The object type

- **uuid**

  The event unique identifier

- **applicationVersion**

  The version of the Privitar application sending the event

- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.50. TEAM_MEMBERSHIP_REMOVED

This event is published whenever a team member is removed

Event Fields:

- **eventType**

  The type of the event

  Value: TEAM_MEMBERSHIP_REMOVED

- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS

- **timestamp**

  The UTC timestamp when the event was created

- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The team id
  - **uniqueId**

    The unique id of the team
  - **objectType**

    The object type
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.51. TEAM_UPDATED

This event is published whenever a team is updated

Event Fields:

- **eventType**

  The type of the event

  Value: TEAM_UPDATED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The team id
  - **uniqueId**

    The unique id of the team
  - **objectType**

    The object type
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.52. UNIQUE_COLUMN_COUNT_PER_RULE

This event is published when we count for the rule how many different combinations (Field name, Data Type) is it used

Event Fields:

- **eventType**

  The type of the event

  Value: UNIQUE_COLUMN_COUNT_PER_RULE
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **ruleUniqueId**

    Unique Id of the Rule that is being used.
  - **uniqueFieldIdDataTypeCombinations**

    Count of for how many different combinations of (Field name, Data Type) the Rule is being used.
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.53. USER_CREATED

This event is published whenever a new user is created

Event Fields:

- **eventType**

  The type of the event

  Value: USER_CREATED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

The UTC timestamp when the event was created

- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The user id

  - **uniqueId**

    The unique identifier of the user

  - **userType**

    The user type

    Example Values: API, UI

  - **userManagementProvider**

    The user management provider

    Example Values: INTERNAL, LDAP, SSO_SAML

  - **superuser**

    Whether or not the user is a superuser

  - **logonPermitted**

    Whether or not the user is allowed to log on

  - **objectType**

    The object type

  - **enabled**

    Whether or not the user is enabled

- **uuid**

  The event unique identifier

- **applicationVersion**

  The version of the Privitar application sending the event

- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.54. USER_UPDATED

This event is published whenever a user is updated

Event Fields:

- **eventType**

  The type of the event

  Value: USER_UPDATED

- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS

- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **id**

    The user id
  - **uniqueId**

    The unique identifier of the user
  - **userType**

    The user type

    Example Values: API, UI
  - **userManagementProvider**

    The user management provider

    Example Values: INTERNAL, LDAP, SSO_SAML
  - **superuser**

    Whether or not the user is a superuser
  - **logonPermitted**

    Whether or not the user is allowed to log on
  - **objectType**

    The object type
  - **enabled**

    Whether or not the user is enabled
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.55. WATERMARK_EXTRACTION_END

This event is published when a watermark extraction finished. This can be published multiple times for a single job

Event Fields:

- **eventType**

  The type of the event

  Value: WATERMARK_EXTRACTION_END

- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **watermarkExtractionJobId**

    The watermark investigation job ID
  - **recordsProcessed**

    Number of records processed at time of publishing
  - **runtimeInMilliseconds**

    Time an operation took in milliseconds
  - **successful**

    Boolean indication if the operation was a success
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.56. WATERMARK_EXTRACTION_SUBMITTED

This event is published when a watermark extraction result gets submitted for matching

Event Fields:

- **eventType**

  The type of the event

  Value: WATERMARK_EXTRACTION_SUBMITTED
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **watermarkExtractionJobId**

    The watermark investigation job ID
  - **recordsProcessed**

    Number of records processed at time of publishing
  - **runtimeInMilliseconds**

    Time an operation took in milliseconds
  - **successful**

    Boolean indication if the operation was a success
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.57. WATERMARK_INVESTIGATION_END

This event is published when a watermark investigation job is completed

Event Fields:

- **eventType**

  The type of the event

  Value: WATERMARK_INVESTIGATION_END
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:
  - **watermarkExtractionJobId**

    The watermark investigation job ID
  - **recordsProcessed**

    Number of records processed at time of publishing
  - **runtimeInMilliseconds**

    Time an operation took in milliseconds
  - **successful**

    Boolean indication if the operation was a success

- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service

17.6.58. WATERMARK_INVESTIGATION_START

This event is published when a watermark investigation job is created

Event Fields:

- **eventType**

  The type of the event

  Value: WATERMARK_INVESTIGATION_START
- **eventSource**

  The source of the event

  Example Values: POLICY_MANAGER, NIFI, KAFKA, SDK, POD, STREAMSETS
- **timestamp**

  The UTC timestamp when the event was created
- **parameters**

  The event parameters

  Sub-fields:

  - **watermarkExtractionJobId**

    The watermark investigation job ID
  - **recordsProcessed**

    Number of records processed at time of publishing
  - **runtimeInMilliseconds**

    Time an operation took in milliseconds
  - **successful**

    Boolean indication if the operation was a success
- **uuid**

  The event unique identifier
- **applicationVersion**

  The version of the Privitar application sending the event
- **instanceId**

  The instanceId that emitted the event. Used to attribute events from the same service