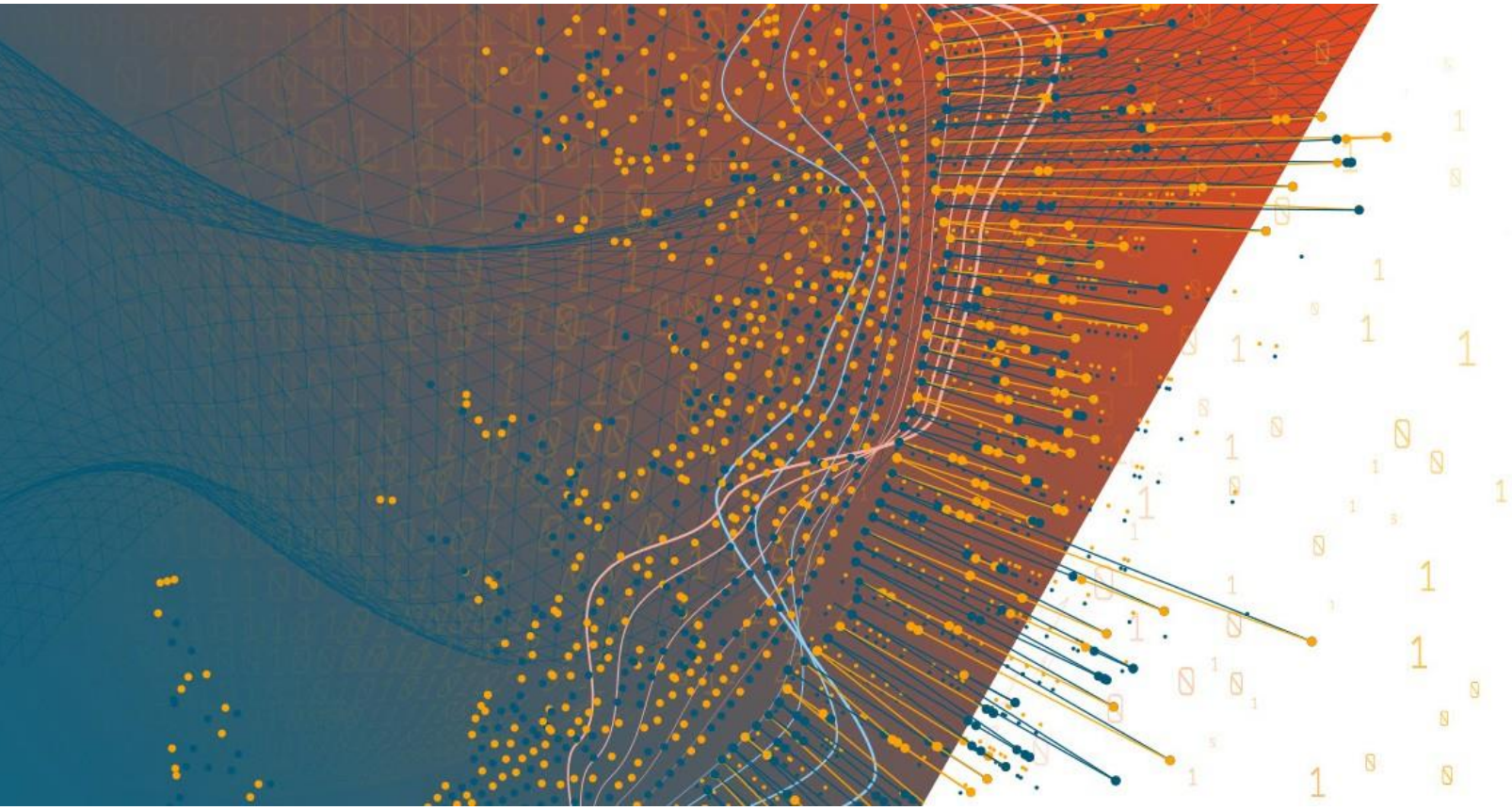


Altair Panopticon™ v2022.1

STREAMS SERVER INSTALLATION AND REFERENCE GUIDE

TABLE OF CONTENTS

[1] INTRODUCTION 1

Acronyms 1
Terminology 1
Overview 1
- Panopticon Streams Applications 2
- Panopticon Streams Operators 3
  - Panopticon Streams Inputs 3
  - Panopticon Streams Outputs 4

System Requirements 4
- System Hardware Requirements 5
  - Development / Test 5
  - Small Scale Deployment 5
  - Medium Scale Deployment 5
  - Large Scale Deployment 5

[2] GETTING STARTED 6

Setting Up Confluent Kafka Enterprise on a Local Machine Using Docker 6
Setting Up ZooKeeper, Kafka, and Schema Registry 7
- Additional Notes on Setting Up the Schema Registry 8

Setting Up Panopticon Streams 9
- Open JDK 11+ Dependencies 13
  - Background 13
  - Installation 14
  - Zip File Content 14
- Importing the Bundle of Example Applications 15
- Configuration of the Client Properties 15

Licensing 17
- Using Altair Units License in Panopticon Streams 18
- Using the Hosted Altair Units License in Panopticon Streams Server 19

Migration to Streams Server 2021.2 from an Older version 21
1. Copy All Content 22
2. Delete Old Content 22
3. One-time Conversion 22
4. Applications, Data Sources, and Data files 22
5. Do Not Make Changes on Both Servers 23
6. Post-migration Cleanup 23

Upgrade 23

[3] AUTHENTICATION 25

Introduction 25

Mapping Users to Roles 25
Token 26
Tomcat Realm 27
Tomcat User Base 28
Tomcat Memory Configuration for Windows 28
Tomcat Memory Configuration for Linux 29
LDAP 29
Active Directory 30
Windows Authentication 31
SAML 31
OAuth 2.0 32
Example 33
Filter 33
Creating a Custom Filter 34
Header 36

[4] PCLI: COMMAND UTILITIES FOR PANOPTICON 37

Export Data Sources 37
Parameters 37
Example 1: Export data sources from a workbook 37
Example 2: Export data sources from all workbooks example 37

[5] USING ALTAIR PANOPTICON STREAMS 38

Connecting to or Disconnecting from the CEP Engine 38
Connecting to the CEP Engine 40
Disconnecting from the CEP Engine: 40

[6] MANAGING THE STREAMS SYSTEM 41

Setting the Server Metrics Publisher 44
Viewing and Managing Kafka Properties 45
Reloading Configurations 46
Logging/Monitoring 47
View Logs 47
Set File Logging Level 49
Scheduling Task to Clear Topic Data 50
Modify a Scheduled Task 52

[7] AUTHORIZATION 53

Secure Access 53
Creating Folders 53
Creating Folders on the Applications Tab 54
Creating Folders on the Data Sources Tab 55

Adding Groups and Users with Allowed Authorization ... 57
Adding Groups and Users with Denied Access ... 60
Creating Subfolders ... 62
Updating Folder or Subfolder Properties ... 65

[8] MANAGING APPLICATIONS ... 67

Uploading Applications ... 68
Folders and Applications Display View ... 72
Importing an Application Bundle ... 74
Exporting an Application or Folder Bundle ... 78
Sorting the List of Applications ... 79
Searching for Applications ... 80
Renaming Applications or Folders ... 81
Viewing Application History and Republishing ... 82
Viewing and Managing Application Topic Usages ... 84
Clearing the Data In an Application Topic ... 85
Clearing the Schema in an Application Topic ... 86
Sorting Application Topics ... 86
Viewing the Application Data Sources Usage ... 87
Moving Applications ... 88
Copying Applications ... 89
Downloading an Application ... 90
Deleting an Application ... 91
Deleting Applications Using the Toolbar ... 91
Deleting Applications Using the Context Menu ... 91
Creating a New Application ... 92
Adding an Operator ... 95
Adding an Input Operator ... 96
Example ... 99
Adding An Aggregation Operator ... 99
Example ... 105
Building the Expression ... 105
Using the Expression Builder ... 106
Supported Aggregation Functions ... 108
Converting Timestamp to/from Integer ... 110
Adding a Branch Operator ... 110
Example ... 112
Example 2 ... 112
Adding a Calculation Operator ... 112
Supported Operators and Calculation Functions ... 116
Supported Operators ... 116
Supported Calculation Functions ... 118
Example ... 125
Adding a Conflate Operator ... 125
Example ... 127
Adding an External Input ... 127
Adding a Filter Operator ... 132

Example ... 133
Adding a Join Operator ... 133
Example ... 137
Fixing Broken Joins ... 137
Adding a Metronome Input Operator ... 142
Example ... 144
Adding a Python Transform Operator ... 144
Example ... 148
Additional Best Practice Recommendations in Using Python with Panopticon ... 148
Adding a Rank Operator ... 149
Example ... 153
Adding a Rekey Operator ... 153
Example ... 155
Adding a REST Transform Operator ... 155
Adding an R Transform Operator ... 160
Additional Best Practice Recommendations in Using R with Panopticon ... 163
Adding a Scatter Operator ... 163
Example ... 165
Adding a Table to Stream Operator ... 166
Example ... 168
Adding a Union Operator ... 168
Adding an Output Operator ... 169
Adding Application-specific Properties ... 171
Example ... 172
Saving an Application ... 173
Editing an Application ... 174
Validating and Fixing Application Issues ... 175
Starting an Application ... 176
Starting an Application on the Applications Tab ... 177
Starting an Application on the Application Page ... 179
Stopping an Application ... 183
Stopping an Application on the Applications Tab ... 183
Stopping an Application on the Application Page ... 184

[9] MANAGING DATA SOURCES ... 186

Uploading Data Sources ... 187
Folders and Data Sources Display View ... 191
Sorting the List of Data Sources ... 193
Searching for Data Sources ... 195
Renaming Data Sources or Folders ... 196
Viewing Application Usages ... 197
Moving Data Sources ... 199
Copying Data Sources ... 200
Downloading a Data Source ... 202
Deleting Data SourceS ... 202
Creating a Data Source ... 203
Common Data Source Settings ... 207

Selecting and Defining the Data Connector File Source ... 207
Defining the Message Type in Data Sources ... 212
Saving or Loading Column Definitions in the Data Sources ... 214
Defining Real-time Settings ... 216
Using the Data Source Toolbar ... 217
Date/Time Key Elements ... 219
Creating Email Output Connector ... 220
Creating InfluxDB Output Connector ... 221
Creating JDBC Database Output Connector ... 222
Creating Apache Kafka Output Connector ... 223
Creating Kx kdb+ Output Connector ... 224
Creating a MQTT Output Connector ... 225
Creating a REST Output Connector ... 226
Creating Text Output Connector ... 229
Creating ActiveMQ Input Data Source ... 230
Creating Amazon Kinesis – Data Streams Data Source ... 233
Creating AMPS Input Data Source ... 236
Creating Cassandra Input Data Source ... 239
Creating Elasticsearch 6.x Input Data Source ... 241
Creating Elasticsearch 7.x Input Data Source ... 242
Elasticsearch Connectors Dependency Installation ... 244
Creating Google Analytics Input Data Source ... 244
Creating Google Cloud Pub/Sub Input Data Source ... 246
Creating an InfluxDB Input Data Source ... 250
Creating JDBC Database Input Data Source ... 251
Creating JDBC (Beta) Input Data Source ... 256
Creating JDBC Database – Streaming Input Data Source ... 261
Creating a JSON Input Data Source ... 264
Creating Apache Kafka Input Data Source ... 265
Creating Kx kdb+ Input Data Source ... 269
Kx kdb+ - Deferred Sync Query ... 273
Creating Kx kdb+tick Input Data Source ... 273
Creating ksqlDB Input Data Source ... 277
Creating ksqlDB - Streaming Input Data Source ... 278
Creating Livy Spark Input Data Source ... 280
Creating MongoDB Input Data Source ... 282
Row-Wise Array Expansion ... 285
Column-Wise Array Expansion ... 285
Bson-Wise Array Expansion ... 285
Creating MQTT Input Data Source ... 286
Creating MS Excel (Legacy) Input Data Source ... 288
Creating MS Excel (xlsx) Input Data Source ... 290
Creating OneTick Input Data Source ... 292
Creating OneTick CEP Input Data Source ... 294
Creating OneTick Cloud Input Data Source ... 296
Creating Python Input Data Source ... 298
Creating RabbitMQ Input Data Source ... 301
Creating Rserve Input Data Source ... 305

Creating Shakti Beta Input Data Source 306
Creating Solace Input Data Source 306
Creating Splunk Input Data Source 309
Creating Stream Simulator Input Data Source 311
Creating StreamBase Input Data Source 315
Creating StreamBase LiveView Input Data Source 317
Creating Text Input Data Source 318
Creating WebSocket Input Data Source 320
Creating Web Data Input Data Source 322
Creating XML Input Data Source 327
Modifying Data Sources 329

[10] MANAGING DATA PRODUCERS 330

Refresh Data Producers 331
Starting or Stopping Data Producers 331

[11] MONITORING ENGINE METRICS AND APPLICATION TOPICS 332

Managing Topics 334
Filter Topics 335
Sorting the List of Topics 335
Moving to Other Topics List Pages 335

[12] MANAGING PARAMETERS 336

Adding Parameters 336
Modifying Parameters 337
Deleting Parameters 338
Refresh Parameters 338
Sorting the List of Parameters 338

[13] EXAMPLE APPLICATIONS 339

[14] PANOPTICON RESOURCES 341

[APPENDIX] 342

Properties: Streams 342

[1] INTRODUCTION

Fundamental to understanding Panopticon Streams are these acronyms and terminologies:

ACRONYMS

Component	Description
CEP	Complex Event Processing
PCLI	Panopticon Command-line Interface

TERMINOLOGY

Component	Description
Apache Kafka or Kafka	Used for building the real-time data pipelines and streaming applications. It is horizontally scalable, fault-tolerant, fast and runs in production in thousands of companies.
Apache ZooKeeper or ZooKeeper	A centralized service for maintaining configuration information, naming, providing both distributed synchronization and group services.
Confluent	The free, open-source streaming platform based on Apache Kafka. The Confluent Platform is the complete streaming platform for large-scale distributed environments. Unlike a traditional messaging system or streaming processing API, Confluent Enterprise enables your interfaces to be connected to anywhere in the world and help make decisions with all your internal systems in real-time.
Schema registry	Part of the Confluent distribution package. Stores a versioned history of all schemas and allows the evolution of schemas according to the configured compatibility settings. Also provides a plug-in to clients that handles schema storage and retrieval for messages that are sent in Avro format.
Panopticon Streams	The name of the Panopticon CEP platform.

OVERVIEW

Event processing is a method of tracking and analyzing streams of information of an event, and eventually deriving a conclusion from what transpired. CEP is an event processing method which combines data from multiple sources to infer events or patterns that may demonstrate unusual activities or anomalies, consequently requiring immediate action.

The CEP engine provided by Panopticon is named **Panopticon Streams** and it is built to work with different CEP engines. However, for this version, it will only support Kafka.

Kafka is a distributed streaming platform that lets you publish and subscribe to streams of records. Each record consists of a **key**, a **value**, and a **timestamp** and stores streams of records in categories called **topics**. Kafka is mainly used for two reasons:

- Building real-time streaming data pipelines that reliably get data between systems or applications
- Building real-time streaming applications that transform or react to the streams of the data

  Refer to https://kafka.apache.org/intro.html for more information.

Panopticon Streams enables you to create streaming data pipelines which both transforms and reacts to streaming data. Aside from Kafka, it is also using ZooKeeper and Schema Registry that are provided by Confluent. ZooKeeper is a key component when using Kafka since it allows the configuration and management of clusters in the Kafka servers. The Schema Registry stores a versioned history of all schemas used by Kafka and provides a RESTful interface for storing and retrieving Avro schemas.

Panopticon Streams Applications

The main task of Panopticon Streams is to execute and manage streams **applications**. An application describes how data should be piped, transformed, and processed. Applications consist of a set of **inputs**, **operators**, and **outputs** and is described or constructed in an XML file.

It can be viewed as a directed graph with a set of nodes (or operators) and a set of edges (or streams) that are interconnected with each other.

Component	Description
ID	The ID of the application config. It should be the same with the filename when loading an application config from the system.
operators	A list of operators (actions and functions).
Streams	A list of streams that describe the connection and the flow between operators.
properties	Application-specific defined properties.

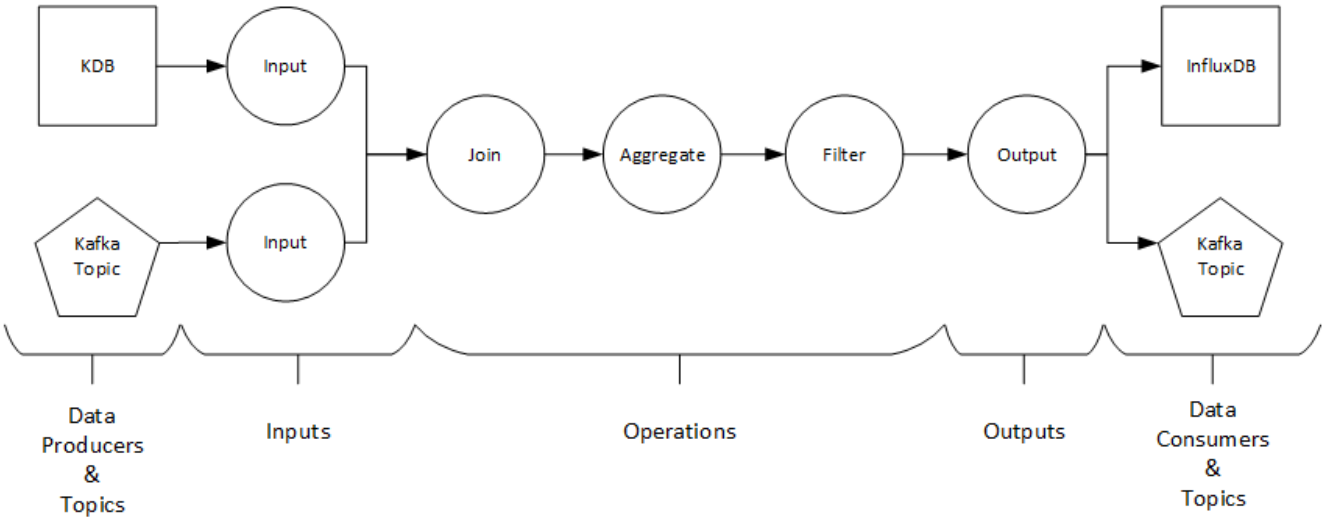


Figure 1-1. Panopticon Streams Framework

An application can either use **Kafka topics** or **data producers**, which generate data from a data source. The data producer also demonstrates to be the connection between the Panopticon Streams framework and the Panopticon core.

The Panopticon core has data connectors such as Kx kdb+, OneTick, and MS Excel that serve as data sources. Just like the application, the data source is also constructed or described in an XML file.

An application refers to a data source through its ID (or filename). There are several ways to create a data source of an application:

- Export data source with the PCLI tool

  The PCLI tool extracts the already defined data sources in workbooks and saves them as CEP data sources.

- Using Panopticon Streams

Panopticon Streams Operators

An **operator** is a single task responsible for processing the data and publishing it as an output. Currently, Panopticon Streams supports the following operators:

- Aggregation
- Branch
- Calculation
- Conflate
- External Input
- Filter
- Input
- Join
- Metronome
- Rank
- Rekey
- Scatter
- To_stream
- Output
- Union
- Python Transform
- REST Transform
- R Transform

Each operator produces one or more output streams that can be connected and defined as input streams for other operators.

Panopticon Streams Inputs

Panopticon Streams engine allows the combination of multiple data sources and their definition as input channels. The data sources are referred to within Panopticon Streams as **inputs**. The data produced by each input can be processed by one or more operators.

Panopticon Streams Outputs

An **output** produces and publishes streams towards a Kafka topic or a **data consumer**. A data consumer is the opposite of a data producer. It consumes the data produced from an output in Panopticon Streams and publishes the data to a data source.

The most common approach is to publish the data to a Kafka topic which eventually can be consumed or used by Panopticon Real Time or other platforms that support Kafka.

Currently, Panopticon Streams supports publishing of the output data to the following data sources:

- Email
- InfluxDB
- JDBC Databases
- Apache Kafka
- Kx kdb+
- Rest
- Text

SYSTEM REQUIREMENTS

Panopticon Streams Server is supported on these operating systems:

- Linux
- Windows 10 (64-bit) – For Development Environments Only
- Windows Server 2012 (64-bit)
- Windows Server 2016 (64-bit)

Panopticon Streams Server also requires:

- Oracle Java SE 8, Oracle Java SE 11, Open JDK 8, and Open JDK 11 are supported after installing the dependency files that are distributed with Panopticon Streams Server

> **NOTE** **Unzip the contents of the dependency package file provided by Panopticon into the `TOMCAT_HOME/lib` folder to be able to run Altair Panopticon software on JRE 8 and Open JDK 8.**

- Apache Tomcat 9.0.x

> **NOTE** **Panopticon Streams Server does not support Tomcat 7.x, Tomcat 8.0.x, or Tomcat 8.5.x.**

Panopticon Streams Server is supported for deployment on the following cloud providers:

- Amazon Web Services (AWS)
- Microsoft Azure
- Google Cloud Platform

- Oracle Cloud

Supported browsers include the latest version of:

- Google Chrome
- Safari

System Hardware Requirements

Development / Test

- 1 x Dual Core CPU (Hyper Threaded to 4 Cores/Threads)
- 8GB RAM
- 4GB Disk (Available)
- In Memory Caching limited to available Server RAM

Small Scale Deployment

- 1 x Quad Core CPU Or Equivalent (Hyper Threaded to 8 Cores/Threads)
- 16GB RAM
- 4GB Disk (Available)
- In Memory Caching limited to available Server RAM

Medium Scale Deployment

- 4 x Quad Core CPU Or Equivalent (Hyper Threaded to 32 Cores/Threads)
- 32GB RAM
- 4GB Disk (Available)
- In Memory Caching limited to available Server RAM

Large Scale Deployment

- 8 x Quad Core CPU Or Equivalent (Hyper Threaded to 64 Cores/Threads)
- 64GB RAM
- 4GB Disk (Available)
- In Memory Caching limited to available Server RAM

[2] GETTING STARTED

Running Panopticon Streams can either be done with:

- a Dockerized Kafka (Confluent Kafka Enterprise platform)
- a local cluster that includes Zookeeper, Kafka, and Schema Registry

Follow the steps below corresponding to the platform you are using.

SETTING UP CONFLUENT KAFKA ENTERPRISE ON A LOCAL MACHINE USING DOCKER

Steps:

1. Install Docker.

   If you install on a Windows machine, you can use Docker Desktop for Windows, see:

   https://docs.docker.com/desktop/windows/install/

   NOTE **If you run Docker on Windows, we recommend that you use the WSL 2 backend. If you choose to use Hyper-V instead, make sure that you select Linux containers, and increase the available memory to at least 8 GB.**

2. Download the Docker Compose script from Confluent:

   https://github.com/confluentinc/cp-all-in-one/raw/v6.0.1/cp-all-in-one/docker-compose.yml

   Save this in a convenient location, such as next to the Panopticon Streams directory.

3. Start the Confluent platform from a command prompt in the same directory as the YML file:

   ```
   docker compose up -d
   ```

   This script defines a number of services that together make up the full Confluent platform. Panopticon Streams really only needs zookeeper, broker, and schema-registry.

4. To verify that everything is working, run `docker compose ps`, and make sure that all services are listed with status **running**.

   Once the Confluent services are up, start Tomcat and Panopticon Streams to build, deploy, and execute your applications.

SETTING UP ZOOKEEPER, KAFKA, AND SCHEMA REGISTRY

NOTE **Windows is currently not a supported platform for running Confluent Kafka, ensure that your OS is on the list of supported operating systems: https://docs.confluent.io/4.0.0/installation/installing_cp.html#system-requirements**

Before proceeding, you must install and setup the following prerequisites:

- ❑ Java JDK 64-bit, version 1.7 or later
- ❑ System Environment variable JAVA_HOME set to the Java JDK 64-bit

Steps:

1. Download one of the Confluent Kafka archives from http://confluent.io/download.
2. Extract the contents of the archive to a new location.
3. Below are the top-level folders of the archive:

```
confluent-3.1.1/bin/         # Driver scripts for starting/stopping services
confluent-3.1.1/etc/         # Configuration files
confluent-3.1.1/share/java/  # Jars
```

4. Start the ZooKeeper, Kafka and Schema Registry processes in the correct order. Make sure the previous process has been started before continuing to the next one.

   - Start ZooKeeper

     ```
     $ ./bin/zookeeper-server-start ./etc/kafka/zookeeper.properties
     ```

   - Start the Kafka broker

     ```
     $ ./bin/kafka-server-start ./etc/kafka/server.properties
     ```

   - Start Schema Registry

     ```
     $ ./bin/schema-registry-start ./etc/schema-registry/schema-registry.properties
     ```

   When these three processes have been started, you can now connect Panopticon Streams to your local Kafka cluster to execute and deploy your applications.

   For more details, refer to the Confluent Kafka Installation-and Quick Start guides:

   - https://docs.confluent.io/3.1.1/installation.html
   - https://docs.confluent.io/3.1.1/quickstart.html

NOTE

When connecting to a Kafka broker on a separate machine, exposing different IP addresses internally and externally, you need to configure `KAFKA_ADVERTISED_LISTENERS`. This is typically the case when running Kafka in a Docker container.

The symptoms of the missing configuration are:

- **Panopticon Streams can connect to ZooKeeper and the Kafka Broker**
- **No data is written to topics**

In `[Kafka]/etc/kafka/server.properties`, uncomment `advertised.listeners` and replace "your.host.name" with the externally exposed host name or IP address.

```
# Hostname and port the broker will advertise to producers
and consumers. If not set,
# it uses the value for "listeners" if configured.
Otherwise, it will use the value
# returned from java.net.InetAddress.getCanonicalHostName().
advertised.listeners=PLAINTEXT://your.host.name:9092
```

When using the Confluent Docker image, you can pass the `KAFKA_ADVERTISED_LISTENERS` as a parameter:

```
docker run -d --restart=always \

--net=confluent \
--name=kafka \
-p 9092:9092 \
-e KAFKA_ZOOKEEPER_CONNECT=zookeeper:2181 \
-e
KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://your.host.name:9092 \
-e KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR=1 \
confluentinc/cp-kafka:5.1.0
```

Additional Notes on Setting Up the Schema Registry

It is recommended to turn off the compatibility checking in schema registry when used with Panopticon Streams.

To do this, set the Avro compatibility level to **NONE** (as mentioned below) in the `schema-registry.properties` file.

Then there are three cases depending on how Kafka is deployed:

- On Windows from the ZIP file from Panopticon. Already turned off by default.
- On Linux manually deployed ("bare metal"). Add the following line to …`etc/schema-registry/schema-registry.properties`

  `Avro.compatibility.level=`**`NONE`**

- With Docker Compose using the Confluent images

  Add the following line to the environment section of the schema-registry service in `docker-compose.yml`

  `SCHEMA_REGISTRY_AVRO_COMPATIBILITY_LEVEL: 'NONE'`

SETTING UP PANOPTICON STREAMS

Follow the steps and guidelines below to install Panopticon Streams on Windows.

NOTE **If you need to upgrade your previously installed Panopticon Streams, proceed to the Upgrade section.**

Steps:

1. Extract the contents of `PanopticonStreamsWAR_<version>.zip` file to a new location.

   This zip file will contain the following files and folder:

   - `streams.war`
   - `streams.xml`
   - `Examples.apz`
   - `CustomMessageParserExample.zip`
   - `CustomTransformExample.zip`
   - `OpenJDK11Dependencies.zip`
   - OpenJDK11Dependencies_README.txt
   - `User_License.rtf`
   - Panopticon Streams Reference Guide
   - examples folder with sample data files and CEP applications and data sources

2. Create the `AppData` folder (i.e., **streamsserverdata**) and ensure that the user account **Local Service** running Tomcat has read/write and execute permissions to this folder.

   Example: `c:\streamsserverdata`

3. Specify the license type that will be used. Use any of the following license types:

   - Volume License file (**PanopticonLicense.xml**) that must be copied to the designated `AppData` folder.
   - Altair Units license. Refer to Using Altair Units License in Panopticon Streams for more information.
   - Hosted Altair Units license. Refer to Using the Hosted Altair Units in Panopticon Streams Server for more information.

4. Copy the extracted `streams.xml` file into the Tomcat config folder (`\Apache Software Foundation\Tomcat 9.0\conf\Catalina\localhost`). This file contains the following information:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/streams">
      <Environment name="PanopticonAppData" override="false"
type="java.lang.String" value="c:\streamsserverdata" />
</Context>
```

NOTE

- **Instead of setting the path of the environment variable `PanopticonAppData` on the `streams.xml` file, you can do so on the System Environment Variables. For example:**

Variable	New Value
PanopticonAppData	`c:\panopticonstreamsdata`

- **If the directory path is set in both an environment variable and in the `streams.xml` file, the value set in the XML file will take precedence.**

- **Starting with 21.2, the `DatawatchVDDAppData` is replaced with `PanopticonAppData` as the specifier for Panopticon application data directory. You can still use `DatawatchVDDAppData` as a fallback, but going forward, `PanopticonAppData` should be used.**

5. Copy the `streams.war` file into the Tomcat `webapps` folder (`\Apache Software Foundation\Tomcat 9.0\webapps`).

6. Edit the existing `tomcat-users.xml` file which is available in the Tomcat config folder (`\Apache Software Foundation\Tomcat 9.0\conf`) and add the entry:

```
<role rolename="user"/>
<role rolename="designer"/>
<role rolename="admin"/>
<user username="viewer" password="viewer" roles="user" />
<user username="designer" password="designer" roles="designer" />
<user username="admin" password="admin" roles="admin" />
```

For more complex authentication and user directory options, see section [3] Authentication.

IMPORTANT

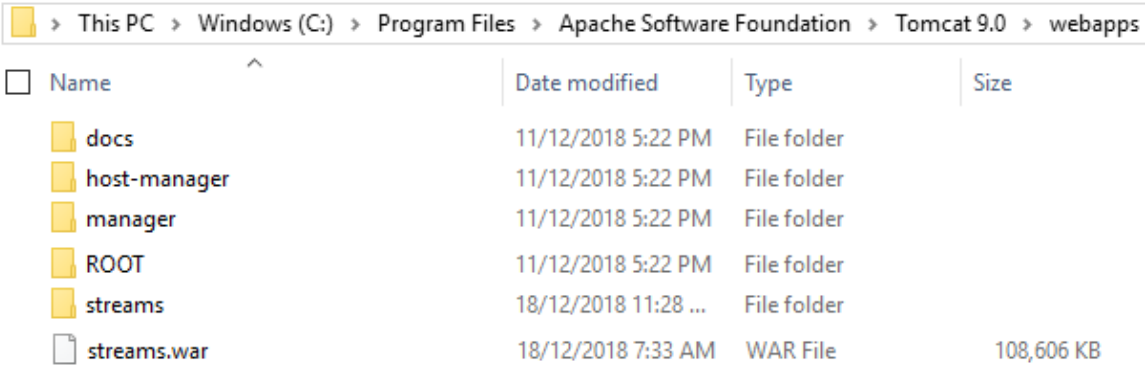
- **Before proceeding to step 7, ensure the Tomcat `temp` folder (e.g., `(\Apache Software Foundation\Tomcat 9.0\temp`) is available.**

- **You can opt to choose a different `temp` folder with the CATALINA_TMPDIR environment variable. For example:**

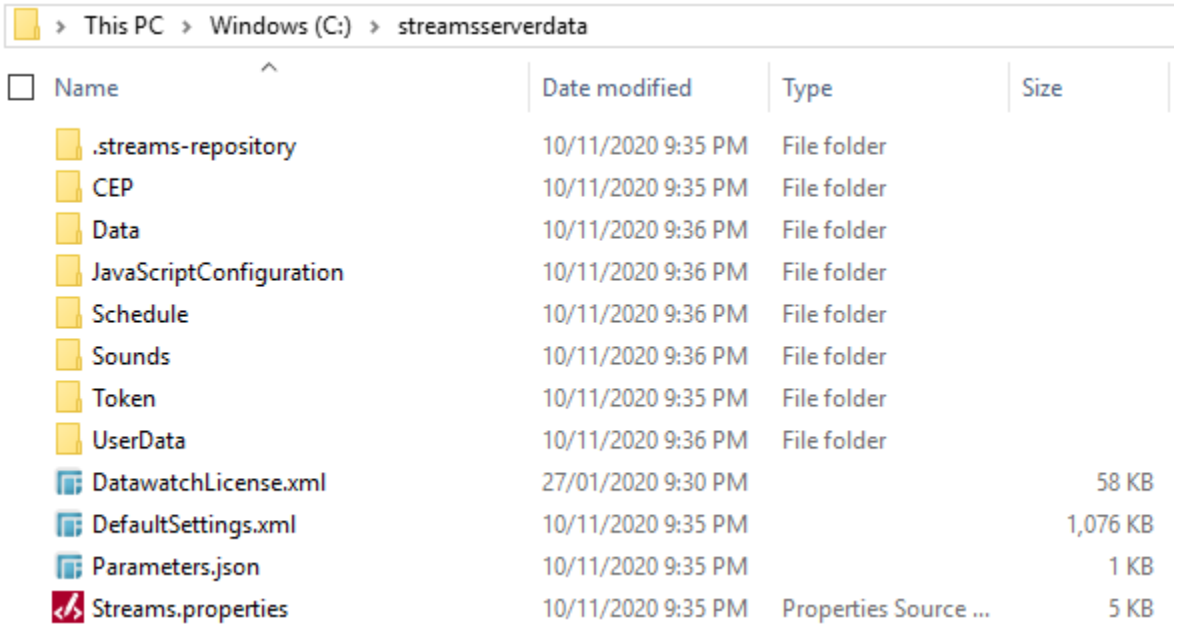
Variable	Value
CATALINA_TMPDIR	`C:\tomcat\dev\temp`

7. Start Tomcat to deploy the `.war` file.

The `streams` folder is extracted in the Tomcat `webapps` folder:



Also, the `.streams-repository`, `CEP`, `Data`, `JavaScriptConfiguration`, `Schedule`, `Sounds`, `Token`, and `UserData` folders are generated in the `streamsserverdata` folder along with the `Streams.properties`, `DefaultSettings.xml` and `Parameters.json` files:

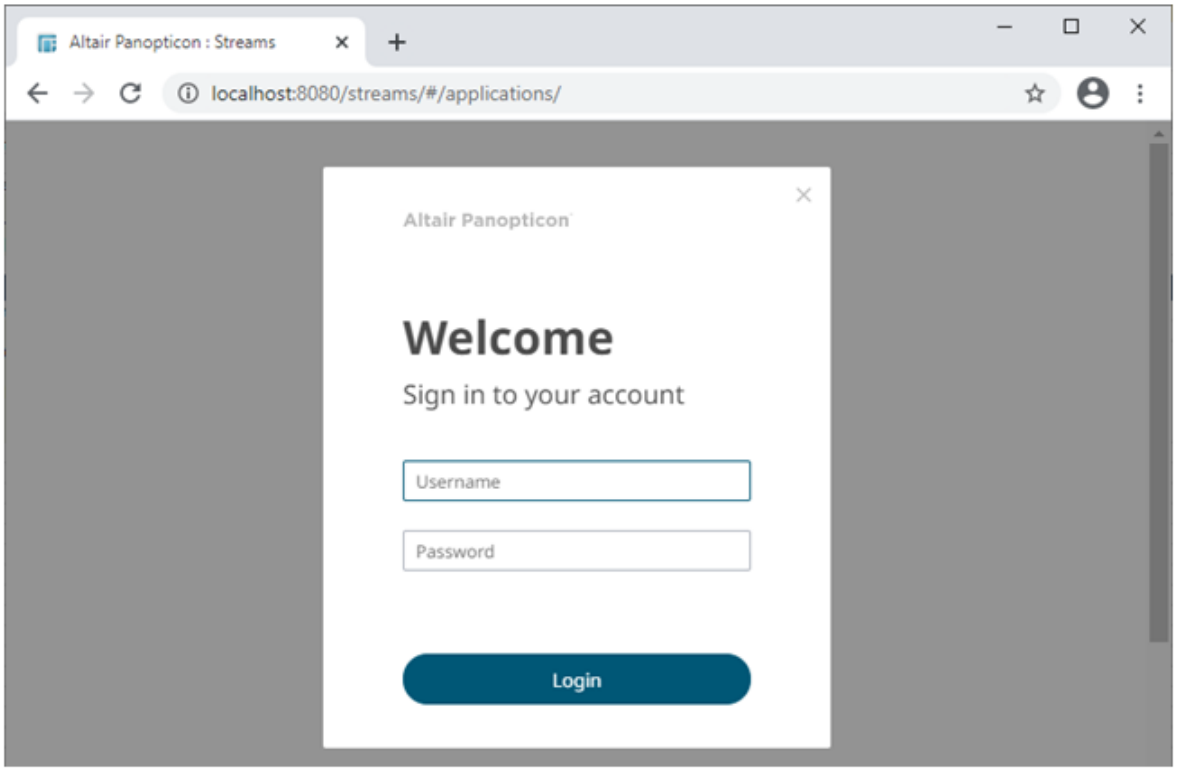


8. You should now be able to log on to Panopticon Streams using the following URL:

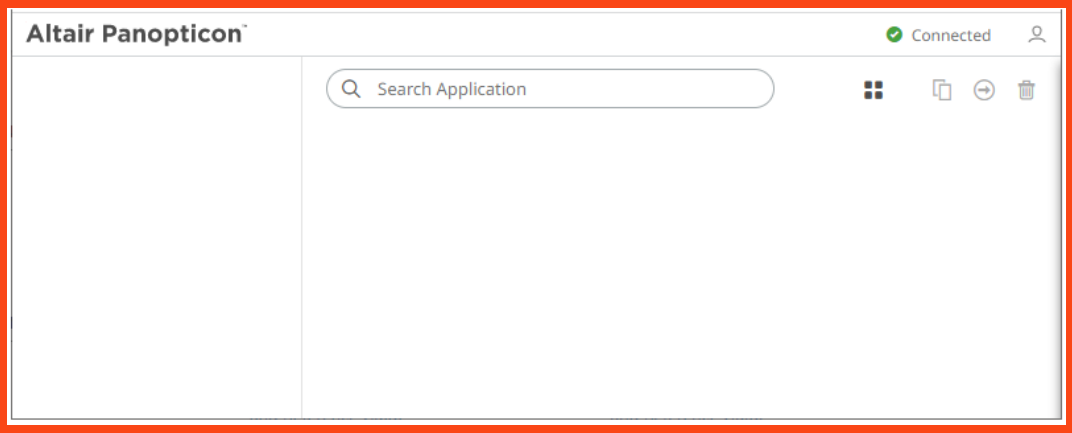
   `[Host Name]:[Port]/[Name of your application]`

   For example:

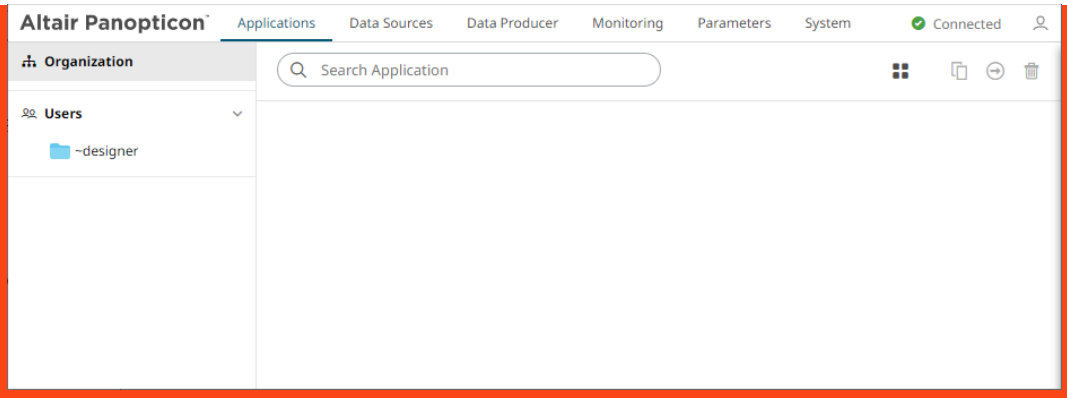
   `http://localhost:8080/streams`



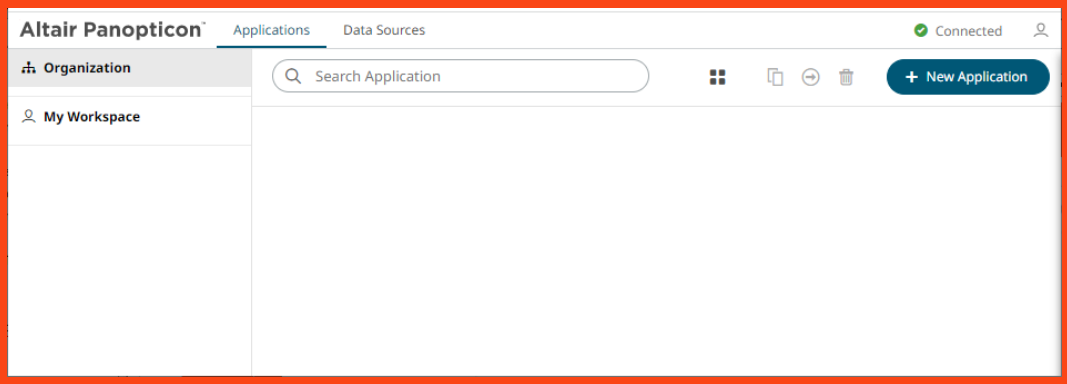
NOTE **Panopticon Streams Server supports different user roles. By default, all users are assigned the VIEWER role. For example, logging on using the viewer user added in step 6, Panopticon Streams Server will only display:**



To have full access to all the services, the user is required to have an ADMINISTRATOR role.



A user with a DESIGNER role can create or upload applications and data sources:



For more information on how to set up the user groups and map them to the user roles, refer to Mapping Users to Roles for more information.

Open JDK 11+ Dependencies

`The AltairPanopticonStreamsWAR_<version number>.zip` file includes `OpenJDK11Dependencies.zip` which contains necessary dependencies for running Altair Panopticon software on Open JDK 11 and up.

The overview, installation, and list of the contents of `OpenJDK11Dependencies.zip` are provided and discussed in the `OpenJDK11Dependencies_README.txt` file.

Background

In Java 9, a number of Java EE modules were marked for deprecation, and subsequently removed completely from Java 11.

With missing Java EE dependencies, the typical exceptions would include `NoClassDefFoundError` exceptions being thrown for `javax/xml/bind` classes.

```
Exception in thread "main" java.lang.NoClassDefFoundError:
javax/xml/bind/JAXBException
    at monitor.Main.main(Main.java:27)
Caused by: java.lang.ClassNotFoundException: javax.xml.bind.JAXBException
    at
java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinClassLoader.java:582
)
```

```
    at
java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(ClassLoaders.java:
185)
    at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:496)
    ... 1 more
```

In order to support deployment on either Java 1.8 or Open JDK 11+, we have packaged the necessary Java EE dependencies separately for simple installation in Tomcat.

Installation

Do the following to make the dependencies available to the JVM and the Altair Panopticon server:

1. Stop Tomcat.
2. Unzip the contents of OpenJDK11Dependencies.zip into the TOMCAT_HOME/lib folder.
3. Start Tomcat.

Zip File Content

- Jakarta XML Binding API (jakarta.xml.bind-api), version 2.3.2
  - jakarta.xml.bind-api-2.3.2.jar
  - jakarta.activation-api-1.2.1.jar
- JAXB Runtime (jaxb-runtime), version 2.3.2
  - jakarta.xml.bind-api-2.3.2.jar
  - txw2-2.3.2.jar
  - istack-commons-runtime-3.0.8.jar
  - jakarta.activation-api-1.2.1.jar
  - stax-ex-1.8.1.jar
  - jakarta.activation-api-1.2.1.jar
  - jakarta.xml.bind-api-2.3.2.jar
  - FastInfoset-1.2.16.jar
  - jakarta.activation-api-1.2.1.jar
- Jakarta SOAP Implementation (saaj-impl), version 1.5.1
  - saaj-impl-1.5.1.jar
  - jakarta.xml.bind-api-2.3.2.jar
  - jakarta.activation-api-1.2.1.jar
  - jakarta.xml.soap-api-1.4.1.jar
  - mimepull-1.9.11.jar
  - stax-ex-1.8.1.jar
- Java API for XML Web Services (jaxws-api), version 2.3.1
  - jaxws-api-2.3.1.jar

- jaxb-api-2.3.1.jar
- javax.activation-api-1.2.0.jar
- javax.xml.soap-api-1.4.0.jar
- javax.annotation-api-1.3.2.jar

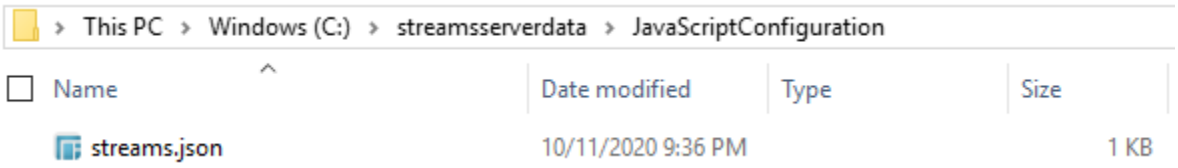
Importing the Bundle of Example Applications

The `AltairPanopticonStreamsWAR_<version number>.zip` file includes the bundle file (`Examples.apz`) of the example applications and their associated data sources and data files.

Follow the instructions in Importing an Application Bundle to import this bundle to Panopticon Streams Server.

Configuration of the Client Properties

Starting with version 2020.1, Panopticon Streams Server generates a `streams.json` configuration file in the `JavaScriptConfiguration` directory of the `AppData` folder (i.e., **c:\streamsserverdata**).



The default content of the `streams.json` file has the following objects/names:

```
{
  "baseUrl" : ".",
  "hideAuthenticationButton" : false,
}
```

NOTE **In the JSON files, a dot in the name (e.g., name1.name2) is used to denote a nested object structure:**

```
{
  "name1": {
    "name2": ...
  }
}
```

In the `streams.json` file, you can control the configuration of the following objects/names:

Object/Name	`baseUrl`
Description	Location of Panopticon Streams Server.
Default Value	**"."**
Required	Yes
Object/Name	`automaticReconnectOnServerDisconnect`
Description	If set to **true**, then the real time connection (WebSocket or long polling) to the Panopticon server will be automatically reconnected if it is disconnected.
Default Value	**false**
Required	No
Object/Name	`dataLoading.transport`
Description	Controls the which transport should be used when viewing log from the server. Valid values are "**websocket**" and "**long-polling**". If configured to "**websocket**", but the WebSocket connection fails, then the web client will automatically fall back to "**long-polling**".
Default Value	**"websocket"**
Required	No
Object/Name	`maxClipboardLength`
Description	Maximum length of text that will be attempted to be put into the system clipboard (copy). If too much text is attempted, then the browser might become unresponsive.
Default Value	**500000**
Required	No
Object/Name	`hideAuthenticationButton`
Description	Boolean. Hides the login and logout buttons.
Default Value	**false**
Required	No
Object/Name	`localization.useBrowserLocale`
Description	Boolean. If set to **true**, then the browser `navigator.language`, `navigator.userLanguage` on IE11, controls the localization of the UI. Not all languages are supported.
Default Value	**true**
Required	No
Object/Name	`localization.defaultLocale`
Description	Locale used if the browser locale is not supported, or if `useBrowserLocale` is set to **false**.
Default Value	**"en-US"**

Required	No
Object/Name	`localization.fallbackLocale`
Description	Locale used if a resource string is missing from the locale in use. Should be specified if `localization.defaultLocale` is specified.
Default Value	value of `localization.defaultLocale`
Required	No
Object/Name	`localizationOverride`
Description	Nested object with resource strings per language. Used to customize resource strings.
Default Value	
Required	No
Object/Name	`logLevel`
Description	Controls which types of logs Panopticon will write to the browser dev console. Valid values are: "**trace**", "**debug**", "**info**", "**warn**", "**error**" and "**silent**".
Default Value	**"info"**
Required	No

NOTE **If there are no config files available on the server, default ones will be created and saved. After that, you can alter them in any way you would like and keep the configuration even if the server is restarted.**

LICENSING

NOTE **In the Panopticon documentation, HyperWorks Units (HWU) and Hosted HyperWorks Units (HHWU) are now named Altair Units and Hosted Altair Units, respectively.**

In the Panopticon product, these license types are still named HyperWorks Units and Hosted HyperWorks Units.

For more information on Altair Units, visit https://www.altair.com/altair-units/.

Licensing within Panopticon Streams supports three license types:

- a volume-based XML file (named **PanopticonLicense.xml**), which is used to store all license information for a specific customer, must be copied to the designated `AppData` folder (i.e., **c:\streamsserverdata**)

NOTE **Starting with 21.2, the newly issued volume-based license file is named `PanopticonLicense.xml`. For customers with the `DatawatchLicense.xml` file, it can still be used but it is strongly recommended to rename it to `PanopticonLicense.xml`.**

- Altair Units license which is available in the Altair License Server you are connected to (local or over the network)
- Hosted Altair Units license

The license file type you will use is delivered separately from the installation packages.

Using Altair Units License in Panopticon Streams

Before using the Altair Units license type in Panopticon Streams, it is required to configure certain properties in the `Streams.properties` file located in the `AppData` folder or `c:\streamsserverdata`:

Property	Service authentication level
Attribute	`authentication.required`
Description	The property that will make the authentication required. It will force the user to login in order to use any of the services provided by the server. Must be set to **true**.
Default Value	**true**
Property	Licensing
Attribute	`license.hwu.operating.system`
Description	The operating system where Panopticon Streams Server is installed. Possible values are: WIN_X86, WIN_X64, MAC, LINUX_X64, or LINUX_ARM64. **NOTE:** If the Java bitness (e.g., 32-bit) is different from the operating system (e.g., 64-bit), it is recommended to add the Java bitness in this property (e.g., WIN_X86).
Default Value	
Property	Licensing
Attribute	`license.hwu.uri`
Description	The path where the License Server is running e.g., **6200@191.255.255.0** where the syntax is `PORTNUMBER@HOST`. If multiple servers are specified, use the '**;**' semicolon separator sign for Windows and the '**:**' colon separator sign for Linux. **NOTE:** If value is not set in the `Streams.properties`, the environment variable **ALTAIR_LICENSE_PATH** serves as the backup path and will be used.
Example	For Windows: `license.hwu.uri=6200@192.168.5.51;6200@192.168.5.52` For Linux:

	`license.hwu.uri=6200@192.168.5.51:6200@192.168.5.52`
Default Value	
Property	Licensing
Attribute	`license.hwu.version`
Description	Value must match the license version found in the Altair Units license file.
Default Value	**19.0**
Property	Licensing
Attribute	`license.mode`
Description	The license mode. Possible values are: FILE or HWU. Must be set to **HWU**.
Default Value	**FILE**

For example:

```
authentication.required=true
license.hwu.operating.system=WIN_X64
license.hwu.uri=6200@192.168.5.51;6200@192.168.5.52
license.hwu.version=19.0
license.mode=HWU
```

NOTE

- **Panopticon Streams Server supports different user roles which check out different numbers of Altair Units.**

Role	Altair Units License Draw
Designer	21
Administrator	2

- **Logging in to both Panopticon Real Time and Panopticon Streams Server with the same username levels the unit draw. A total of 21 units are drawn even if the user logs in to both servers.**
- **Running applications are leveled towards the user who started the application.**

  For example, a user can run 10 Streams applications while also being logged in as an Administrator and the total unit draw will only be 21. For the 11th application the total draw will be 22. After this, units will increase by 2 for each additional application. For the 12th application, the total draw will be 24.

Using the Hosted Altair Units License in Panopticon Streams Server

Using the Hosted Altair Units licensing will support simplifying the license management by removing all manual aspects of emailing license files, extending evaluation periods, among others.

In addition, Hosted Altair Units licensing will help small to medium deployment customers who do not want to host on-premise license server.

Before using the Hosted Altair Units license type in Panopticon Streams Server, it is required to configure certain properties in the `Streams.properties` file located in the `AppData` folder or `c:\streamsserverdata`:

Property	Licensing
Attribute	`license.hwu.hosted`
Description	Boolean stating if you wish to use Hosted or Local Altair Units licensing. Set to **true** if you wish to use hosted licensing.
Default Value	**false**
Property	Licensing
Attribute	`license.hwu.hosted.authorization.username`
Description	Username to the Altair One account.
Default Value	
Property	Licensing
Attribute	`license.hwu.hosted.authorization.password`
Description	Password to the Altair One account.
Default Value	
Property	Licensing
Attribute	`license.hwu.hosted.authorization.token`
Description	An authorization token generated through the Altair One admin portal. Used to authorize a machine to the Hosted Altair Units system.
Default Value	

NOTE

- **To use the Hosted Altair Units licensing, set the following properties:**

```
license.hwu.hosted=true
license.mode=HWU
license.hwu.operating.system= WIN_X64
authentication.required=true
license.hwu.uri=6200@localhost
license.hwu.version=20.0
```

- **Add the Panopticon application to your Altair One account.**

To authorize the machine against the Hosted Altair Units system, you have two options.

Option 1

If you wish to generate the authorization token through Altair One:

1. Log on to Altair One (https://admin.altairone.com) then navigate to **User Profile -> View My Authorized Machines -> Generate Auth Code** (up in the right corner).

2. Paste the generated code into the `license.hwu.hosted.authorization.token` property in the `Streams.properties` file.
3. Start the server.

Option 2

To eliminate token generation on your own:

1. Enter your Altair One credentials into the `license.hwu.hosted.authorization.username` and `license.hwu.hosted.authorization.password` properties in the `Streams.properties` file.
2. Start the server.

NOTE

- **If a token is entered, this will be tried first. If the token was invalid or not present, and credentials are present, the credentials will be used to authorize the machine towards the Hosted Altair Units system.**
- **A working Internet connection is required to use Hosted Altair Units licensing.**
- **If you don't have an Altair One account, you can sign up for a free trial and that will allow you to test the product for 14 days.**

MIGRATION TO STREAMS SERVER 2021.2 FROM AN OLDER VERSION

These instructions assume that you:

- have an existing 2020.1 or older server installed and want to migrate the content to a new installation of the 2021.0 server.
- want to keep running the old server while you make sure that the migration was successful, and that the new server is running as it should.

All of the server content is stored in its application data `<appdata>` folder, the path of which is set in the `PanopticonAppData` context environment property. For example, in Tomcat this would be in `<tomcat_home>/conf/catalina/localhost/streams.xml` or similar.

NOTE **Two Panopticon web applications should never share the same `<appdata>` folder, ensure that the new server is pointed at its own initially empty folder.**

Some of the content can simply be copied from the old server to the new one, while some is now stored in a new format and needs to be converted. The applications and data sources themselves can be migrated any number of times, essentially resetting the applications on the new server.

Summary of steps:

1. Copy all content.
2. Delete old content.

3. One-time conversion
4. Applications, data sources, and data files
5. Do not make changes on both servers.
6. Post-migration cleanup

1. Copy All Content

Start by copying all files from `<old_appdata>` to `<new_appdata>`. You can selectively copy some files again later to keep the old and new server in sync (e.g., copy over scheduled tasks after they are modified on the old server). This completes the **migration** of the following:

- License file - The server will not start without a valid `<appdata>/PanopticonLicense.xml`. In 2020 you also have the option to use Altair units licensing instead of the XML file.
- Properties file - The set of properties in `<appdata>/Streams.properties` that the server understands changes between releases. The first time you start it, it will add new one and remove old properties.
- Scheduled tasks - All scheduled tasks are in SCH files in `<appdata>/Schedule/`.

2. Delete Old Content

On the new server, delete the `<new_appdata>/Tokens/` folder. This holds authentication tokens for logged in users, and they are server specific.

3. One-time Conversion

NOTE **Converting applications and data sources is covered in the next section.**

On the old server, parameters were stored in `<old_appdata>/DefaultParameters.xml`. They were global and applied to all content (applications and data sources). In 2021.0 you can now organize content in folders, and you can also define parameters that only apply to content in a particular folder. The new server stores them all in `<new_appdata>/Parameters.json`.

If `<appdata>/Parameters.json` doesn't exist when the new server starts, it will create it, and if it finds `<appdata>/DefaultParameters.xml` it will import these into the new file. To repeat the conversion, e.g., if you want to re-import changed parameters from the old server, delete `Parameters.json` and restart the server.

4. Applications, Data Sources, and Data files

Applications and their change history, and data sources, are stored in a very different format in a repository inside the `<appdata>/.streams-repository/ folder`. This is preparation for better versioning, content synchronization in a cluster and other things.

Before version 2020.2, all applications were stored as individual APP files in `<appdata>/CEP/Applications`. Every time an application was updated, a backup was placed in `<appdata>/CEP/Archive`. Data sources were stored as DSM files in `<appdata>/CEP/Datasources`.

If the new server starts and the `<appdata>/.streams-repository/` folder doesn't exist, it will create one, and then look in the `<appdata>/CEP/` folder. Any applications and data source files it finds in there, it will import into the newly created repository. After the import, the `<appdata>/CEP/` folder is no longer used.

Optionally, you can also import all application backups from `<appdata>/CEP/Archive/`. If you do, they will be recorded as application edits in the new repository. While the web UI currently doesn't expose the change history, it may very well do so in the future.

> **NOTE** To opt out, set **`repository.import.archived.applications`** to false in Streams.properties.

You can repeat this migration as many times as you like: stop the new server, delete the entire `<new_appdata>/.streams-repository/` folder, then start the new server. This provides a convenient way to keep the new server in sync with changes on the old server, assuming the old server is still in use. Please note that this process will lose all changes made on the new server only, as they are stored in the repository.

Data sources that use data files (e.g., CSV, JSON, XML) with relative paths expect the path to be relative to the `<appdata>/Data/` folder. You can simply copy the entire `<old_appdata>/Data/` folder to `<new_appdata>`.

5. Do Not Make Changes on Both Servers

After the initial migration you can keep the new server up to date when content changes on the old server by repeating any of the steps above. It is much harder to move content the other way, from the new server to the old one. Therefore, avoid making changes (that you want to keep) on the new server until you've completely migrated and retired the old server.

6. Post-migration Cleanup

When you are satisfied that new server is running as it should, that all content has been migrated, switched users over to the new server, and are no longer using the old server, you can remove files from `<new_appdata>` that are no longer needed.

- `<appdata>/DefaultParameters.xml` - These are now in the JSON file.
- `<appdata>/CEP/Applications/` - Applications are now stored in the repository.
- `<appdata>/CEP/Archive/` - If you migrated the change history, this is also in the repository now. Otherwise you can keep it if you want to go back to an earlier application version.
- `<appdata>/CEP/Datasources/` - Data sources are now also in the repository.

UPGRADE

A previously installed Panopticon Streams can be upgraded through the following process:

1. Stop Tomcat.
2. Delete the existing `webapps\streams.war` file.
3. Delete the deployed application: `webapps\streams`.
4. Delete the cache from the working folder (for example):

`work\Catalina\localhost\streams`

5. Deploy the new `streams.war` file by copying it to the Tomcat `webapps` folder.
6. Restart Tomcat.

[3] AUTHENTICATION

INTRODUCTION

Panopticon Streams provides multiple approaches on authentication. It can easily be configured to use different authentication mechanisms depending on the environment and the setup. The server only supports authentication and authorization and does not have any support for user management or administration of users.

There are mainly two properties that manage the authentication on the server. These properties are listed and described in the table below. Please note that more properties might need to be configured depending on the authentication mechanism you are using.

Property	Description	Default Value
authentication.role	The required role or group that the user needs to be identified as a Panopticon user. The property can be left blank if no role or group is required.	
authentication.required	This property will make the authentication required. It will force the user to login in order to use any of the services provided by the server.	**true**
authentication.type	The type of authentication that should be used when authenticating the user. The property allows the following values: **BASIC, FILTER, HEADER, OAUTH2, SAML, WINDOWS**.	**BASIC**

Mapping Users to Roles

Depending on the authentication or user management mechanism used, the role that a user should have is specified and then mapped to a group set in `Streams.properties`.

Property	Description	Default Value
access.administrator.groups	The role that is mapped to the administrator group. Allowed to perform the following: • connect to or disconnect from the CEP Engine. • create, rename, remove folders and subfolders, upload applications or data sources, and manage users or groups that should be granted or denied access. • import and export application bundles. • rename, view topic or data source usage, move, copy, download, remove, and publish/republish applications to folders to which the user has permission. • rename, view application usage, move, copy, download, and remove data sources. • Administer the server which includes:	**admin**

	○ refresh, start, and stop data producers. ○ view engine metrics and retrieved messages. ○ add, modify, refresh, and delete parameters. ○ define file logging level or view, pause, resume logging, copy, and clear all logs ○ view Kafka properties. ○ reload configuration. ○ create, modify, and delete clear topic data tasks.	
access.default.roles	The default roles applied to all users of the server. For example, If `access.default.roles`=DESIGNER,ADMINISTRATOR and a user with a VIEWER role logs on to the server, then the user will simultaneously have a VIEWER, DESIGNER, and ADMINISTRATOR roles. However, if no default roles are wanted, then leave the property blank. **NOTE:** The roles that can be assigned in this property can only be ADMINISTRATOR, VIEWER, ANONYMOUS, and/or DESIGNER. This property is case sensitive.	**VIEWER**
access.designer.groups	The role that is mapped to the designer group. Allowed to perform the following: • create, rename, remove folders and subfolders, upload applications or data sources, and manage users or groups that should be granted or denied access. • create, rename, view topic or data source usage, move, copy, download, remove, and publish/republish applications to folders to which the user has permission • create, rename, view application usage, move, copy, download, and remove data sources. • import and export application bundles.	**designer**
access.viewer.groups	The role that is assigned to the viewer group. Allowed to view the engine status.	**viewer**

NOTE **Group sets can be added for a role, by default separated by a comma.**

Token

A web token is used when the user has successfully logged into Panopticon Streams when using one of the following authentication types: BASIC, SAML, or WINDOWS. The token is used to identify the user and represent the user's ongoing session. This is done to prevent user credentials being sent between the user and server more than necessary.

The token is returned from Panopticon Streams in the form of a cookie when the user has been authenticated. The cookie will be stored in the browser as a HttpOnly cookie.

The token can be configured differently to suit your needs and requirement. The token can be configured to be valid at a certain amount of time, if it can refresh itself and/or if it should be persistent or if it should only last for a user session (While the browser is still open). All this can be configured in the `Streams.properties`. The table below lists all available token properties.

Property	Description	Default Value
authentication.token.persistence	This property is used to determine if the token should persist if the browser is closed or if it should only last while the browser is open. There are two possible values: **PERSISTENT** and **SESSION**. PERSISTENT will persist the token in the browser even if the browser has been closed and reopened. SESSION will remove the token from the browser if it is shutdown. IMPORTANT: After modifying the property value to **SESSION**, ensure to clear the `AppData/Token` folder before starting the server.	PERSISTENT
authentication.token.refreshable	This property determines if the token can refresh itself. The web client can identify if the token is about to expire and then request a new token with the existing token. A token is refreshable if the property is set to **true**. The token will expire and invalidate the user session if the property is set to **false**.	true
authentication.token.secret	The secret is used to sign the token. The secret will be auto-generated when the server starts for the first time. **NOTE:** This value should be kept a secret.	Auto-generated
authentication.token.validity.seconds	The number of seconds that the token should be valid.	604800

TOMCAT REALM

Panopticon Streams can be configured to use the Tomcat Realm when performing authentication. The Tomcat Realm is configured in the `server.xml` file in the Tomcat `conf` folder. The Tomcat Realm itself can be configured to authenticate towards a variety of different types of authentication source, such as Tomcat user base and LDAP. The sub chapters in this chapter will give examples on how to configure the Tomcat Realm.

Panopticon Streams needs to be configured to use the BASIC type in order to do the authentication towards the Tomcat Realm. To enable Tomcat Realm authentication, set this property in the `Streams.properties` file:

`authentication.type`=**BASIC**

NOTE **It is a common approach to wrap your Tomcat Realm with the LockOutRealm. This is used to prevent brute-force attacks.**

```
<Realm  className="org.apache.catalina.realm.LockOutRealm">
    <!—Insert your own Tomcat Realm here →
  </Realm>
```

Tomcat User Base

The Tomcat User Base Realm is using a JNDI resource to store user information. By default, the JNDI resource is configured in an XML file. The default file is `tomcat-users.xml` in the Apache Tomcat `conf` folder.

We strongly recommend using this authentication approach for your test or local environment. It is easy to setup and configure. However, it is not designed to be used for large-scale production or when you have a large number of users.

The following Realm has to be added in the `server.xml` file in the Apache Tomcat `conf` folder:

```
<Realm className="org.apache.catalina.realm.UserDatabaseRealm"
resourceName="UserDatabase"/>
```

NOTE **The Tomcat User Database Realm is used as the default. No configurations are required in the `server.xml` file to be able to use the Tomcat Database Realm.**

The users and roles are managed in the `tomcat-users.xml` file in the Apache Tomcat `conf` folder. In this file, you can add users and roles as well as assign roles to users.

Add the following role and user to your `tomcat-users.xml` file:

```
<role rolename="admin"/>
<user username="John" password="john" roles="admin"/>
```

By adding these two lines you have achieved the following:

- Created a new role named **admin**
- Created a new user with username **John** and password **john**
- Assigned the newly created user the role **admin**

NOTE **Authentication towards a Tomcat Realm (i.e., Tomcat users, LDAP, AD) in Tomcat 8.5.28 is not supported. This has been supported in all the previous and the succeeding versions.**

Tomcat Memory Configuration for Windows

NOTE **It is recommended to increase the Java heap size of Tomcat to avoid the initiation of garbage collection when memory usage hits the set threshold.**

The steps may vary depending on how Tomcat was deployed.

Steps:

1. Stop Tomcat.
2. Create a file named `setenv.bat`.
3. Place the file in the Tomcat `bin` folder.
4. Set the minimum and maximum heap size with the JVM `-Xms` and `-Xmx` parameters. A minimum of 1 GB is recommended. For example:

```
set JAVA_OPTS=%JAVA_OPTS% -Dfile.encoding=UTF-8 -server -Xms512m -Xmx2g
```

NOTE **Setting the maximum value should be dependent on your system. Ensure that the heap size is not larger than the available free RAM on your system. It is recommended to use 80% of the available RAM not taken by the operating system or other processes of your JVM.**

5. Save the file.
6. Restart Tomcat to apply the increase in the heap.

Tomcat Memory Configuration for Linux

NOTE **It is recommended to increase the Java heap size of Tomcat to avoid the initiation of garbage collection when memory usage hits the set threshold.**

The steps may vary depending on how Tomcat was deployed.

Steps:

1. Stop Tomcat.
2. Create a file named `setenv.sh`.
3. Place the file in the Tomcat `bin` folder.
4. Set the minimum and maximum heap size with the JVM `-Xms` and `-Xmx` parameters. A minimum of 1 GB is recommended. For example:

   ```
   JAVA_OPTS="$JAVA_OPTS -Dfile.encoding=UTF-8 -server -Xms512m -Xmx2g"
   ```

   NOTE **Setting the maximum value should be dependent on your system. Ensure that the heap size is not larger than the available free RAM on your system. It is recommended to use 80% of the available RAM not taken by the operating system or other processes of your JVM.**

5. Save the file.
6. Restart Tomcat to apply the increase in the heap.

LDAP

Panopticon Streams can be configured to authenticate towards a Lightweight Directory Access Protocol (LDAP) or source. By configuring the Apache Tomcat Realm, the server can authenticate users and extract their roles by querying the LDAP source.

The realm's connection to the directory is defined by the `connectionURL` attribute. Each user that can be authenticated must be represented in the directory with an individual entry that corresponds to an element in the initial `DirContext from` the `connectionURL`. This user entry must have an attribute containing the username that is presented for authentication.

You can add a dedicated user with `connectionName` and `connectionPassword` in a Realm to define a user with a Read access to the user database and roles. If for example the admin `cn` name is set as **admin** and the admin `password` is set as **admin**, then you need to add these properties as shown in the example below.

The `userPattern` attribute may be used to specify the DN, with "`{0}`" marking where the username should be substituted.

The role is usually an LDAP group entry with one attribute containing the name of the role and another one whose values are distinguished names or usernames of the users in that role. The following attributes configure a directory search to find the names of roles associated with the authenticated user:

- **roleBase**: The base entry for the role search. If not specified, the search base is the top-level directory context
- **roleSearch:** The LDAP search filter for selecting role entries
- **roleName:** The attribute in a role entry containing the name of that role
- **roleNested:** Includes nested roles if set to **true**. This means every newly found `roleName` and distinguished `Name` will be recursively tried for a new role search. The default behavior is **false**.

The following is an example on how the Realm can be configured when using LDAP. Please note that the values should be replaced with details from your own LDAP source.

```
<Realm className="org.apache.catalina.realm.JNDIRealm"
    connectionURL="ldap://localhost:389"
    connectionName="cn=admin,dc=test,dc=com"
    connectionPassword="admin"
    userPattern="uid={0},ou=users,dc=test,dc=com"
    roleBase="ou=groups,dc=test,dc=com"
    roleName="cn"
    roleSearch="(uniqueMember={0})"
    rolenested="true"

/>
```

Using this configuration, the realm determines the user's distinguished name by substituting the username into the `userPattern`, authenticates by binding to the directory with this DN and the password received from the user, and searches the directory to find the user's roles.

NOTE **If you opt not to have a dedicated user, remove `connectionName` and `connectionPassword`, and then have each user extract information about itself. You do this by adding `userSearchAsUser` and `roleSearchAsUser` in a Realm, and setting both values to `true`. The recommended usage, however, is to have a dedicated user. This allows you to always have the rights to query a LDAP, unlike using `userSearchAsUser` and `roleSearchAsUser` where there is no guarantee that each user is authorized to extract these details.**

Active Directory

Panopticon Streams can be configured to authenticate towards an Active Directory server. Panopticon Streams is using LDAP to interact and communicate with the Active Directory server. Therefore, the configuration is very similar to the LDAP configuration in the previous section.

The following is an example on how the Realm can be configured when using Active Directory. Please note that the values should be replaced with details from your own LDAP source.

```
<Realm className="org.apache.catalina.realm.JNDIRealm"
   connectionURL="ldap://ad.dwch.com:3268"
   alternateURL="ldap://ad.dwch.com:389"
   authentication="simple"
   referrals="follow"
   connectionName=admin@DWCH.com
   connectionPassword="admin"
   userBase="cn=Users,dc=DWCH,dc=com"
   userSearch="(sAMAccountName={0})"
   userSubtree="true"
   roleBase="cn=Users,dc=DWCH,dc=com"
   roleName="cn"
   roleSearch="(member={0})"
   roleSubtree="true"
   roleNested="true"
/>
```

NOTE **Similar with LDAP, you can opt not to have a dedicated user by removing `connectionName` and `connectionPassword` and instead let each user extract information about itself by adding `userSearchAsUser` and `roleSearchAsUser` in a Realm. Set both values to `true`. As mentioned in the LDAP section, the recommended usage is to have a dedicated user since there is no guarantee that each user is authorized to extract these details.**

WINDOWS AUTHENTICATION

Panopticon Streams supports Windows authentication. Panopticon Streams will authenticate a user towards the local machine and verify its credentials with the existing and configured users on the Windows machine. The Windows authentication operates similarly to the Basic authentication function. Both the username and the password are sent to Panopticon Streams which they are then verified.

To enable Windows authentication, set this property in the `Streams.properties` file:

`authentication.type`=**WINDOWS**

NOTE **Single Sign On is currently not supported with the Windows authentication. In addition, Windows authentication only supports authentication towards the local machine. This means that the machine where Panopticon Streams is deployed on also has to manage all of the users.**

SAML

Panopticon Streams supports Security Assertion Markup Language, SAML2. Upon a login request, Panopticon Streams will redirect the user to an Identity provider (IdP). The IdP will authenticate the user and redirect the user back to Panopticon Streams. The response message will be controlled and validated. Username and roles will be extracted from the response message and used within Panopticon Streams.

Panopticon Streams will redirect the user back to the IdP upon a logout request. The IdP logout service should then invalidate the SAML token.

Property	Description
authentication.saml.assertion.roles	User attribute for roles configured in the IdP.
authentication.saml.assertion.username	User attribute for username configured in the IdP.
authentication.saml.assertionconsumerservice.url	The URL to the Panopticon assertion consumer service. URL: [Protocol]://[Host]:[Port]/[Context]/server/rest/auth/login
authentication.saml.certificate.name	The name of the certificate used to validate signature and/or sign outgoing SAML messages
authentication.saml.certificate.password	The password of the certificate used to validate signature and/or sign outgoing SAML messages..
authentication.saml.identityprovider.logout.url	The URL to the IdP logout service.
authentication.saml.identityprovider.url	The URL to the IdP login service.
authentication.saml.keystore.file	The location of the Keystore file that contains the certificate.
authentication.saml.keystore.password	The password to the Keystore file.
authentication.saml.serviceprovider.id	The ID of the service provider configured in the IdP.
authentication.saml.identityprovider.certificate.file	Takes a file path to a certificate file that contains the IdP's public key.
authentication.saml.identityprovider.signature.validation.required	Specifies whether to require a valid IdP signature to be present on the SAML response. Default value is **false**.
authentication.saml.provider	The IdP provider. Possible values are **OPENSAML**, **OPENAM**. Default value is **OPENSAML**.
authentication.saml.keystore.type	The key store type. Possible values are **JKS, JCEKS**, **PKCS12**. Default value is **JKS**.
authentication.saml.openam.meta.alias	The meta alias for the IdP if you are using OpenAM.

OAUTH 2.0

This section discusses how to configure Panopticon Streams to use the OAuth 2.0 for authorization. Upon a logon request, Panopticon Streams will redirect the user to the Login page provided by the OAuth 2.0.

Note that OAuth 2.0 does not normally provide support on how to authenticate the user, Panopticon Streams will only know if the user is authorized or not. To authenticate the user, Panopticon Streams can be configured to use a REST service to extract the user identity with an access token retrieved from the OAuth 2.0 provider. In addition to the standard OAuth 2.0 configurations, the server includes properties (i.e., `authentication.oauth2.*`) that are specifically used to extract the user details.

`authentication.type`=**OAUTH2**

Property	Description
authentication.oauth2.client.id	The ID of the OAuth 2.0 client.
authentication.oauth2.client.secret	The secret used by the OAuth 2.0 client.
authentication.oauth2.identity.attribute.username	The attribute that will be extracted from the identity response and used as the username.
authentication.oauth2.identity.url	The URL to the REST service that provides details about the authenticated user.
authentication.oauth2.login.callback.url	The callback URL. The URL should be the same as one of the specified callback URLs used by the client. The URL should refer to Panopticon Streams
authentication.oauth2.login.response.type	The response type. The only response type that is currently supported is **CODE**. The value can also be left blank.
authentication.oauth2.login.scope	The requested scope. The field can be left blank.
authentication.oauth2.login.state	The requested state. The field can be left blank.
authentication.oauth2.login.url	The URL to the OAuth 2.0 login resource.
authentication.oauth2.logout.url	The URL to the OAuth 2.0 logout resource. This field can be left blank.
authentication.oauth2.token.method	The method on how the token should be retrieved. Supported values are **QUERY**, **BODY,** and **HEADER.**
authentication.oauth2.token.url	The URL to the OAuth 2.0 token resource.

Example

```
authentication.oauth2.client.id=ClientId
authentication.oauth2.client.secret=ClientSecret
authentication.oauth2.identity.attribute.username=name
authentication.oauth2.identity.url=https://oauth2/me
authentication.oauth2.login.callback.url=http://localhost:8080/panopticon/ser
ver/rest/auth/login
authentication.oauth2.login.response.type=CODE
authentication.oauth2.login.scope=
authentication.oauth2.login.state=
authentication.oauth2.login.url=https://oauth2/authorize
authentication.oauth2.logout.url=
authentication.oauth2.token.method=QUERY
authentication.oauth2.token.url=https://oauth2/access_token
authentication.type=OAUTH2
```

FILTER

Custom authentication filters can be applied to the server and the application when the default authentication settings are not sufficient. This type of authentication is referred to as **Filter authentication**. When Panopticon Streams is configured to use filter authentication, it means that the incoming requests have already been authenticated and authorized before reaching the server. Follow the steps below to configure filter authentication:

1. Open the `Streams.properties` file in the `AppData` folder (**c:\streamsserverdata**).
2. Enable `authentication.type=FILTER` in `Streams.properties`.
3. Apply the following URL pattern to your own filter: `/*`
4. Save the changes and restart the Tomcat.

Creating a Custom Filter

The custom filter will be a basic authentication filter which will authenticate the user with hardcoded values. The Principal forwarded by the filter will be used to authenticate the user.

The filter will require the following dependencies:

- Javax Servlet
- Tomcat embed core

Steps:

1. Create a HTTP request wrapper.

   The class will contain the following:

   - the original incoming HTTP request
   - the Principal which contains both the credentials and the roles for the authenticated user.

   The HTTP wrapper will be forwarded to Panopticon Streams instead of the original incoming HTTP request.

```
import org.apache.catalina.realm.GenericPrincipal;
import org.apache.catalina.users.MemoryUser;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletRequestWrapper;
import java.security.Principal;

public class FilterRequestWrapper extends HttpServletRequestWrapper {

   private final GenericPrincipal principal;

   public FilterRequestWrapper(final HttpServletRequest request, final
GenericPrincipal principal) {
        super(request);
        this.principal = principal;
    }

    @Override
    public Principal getUserPrincipal() {
        return principal;
    }

    @Override
    public boolean isUserInRole(final String role) {
        if (principal != null) {
            return principal.hasRole(role);
        }
        return super.isUserInRole(role);
    }
}
```

2. Create a custom filter. The filter will create a new Principal which includes both the credentials and the groups/roles for the user.

   In this example, the class `GenericPrincipal` contains username, password, and groups. Panopticon Streams is only able to extract the groups from `GenericPrincipal` class or the `MemoryUser` class. Both the Principal and the original HTTP request will be wrapped in an instance of `FilterRequestWrapper`. The wrapper will then be forwarded towards Panopticon Streams.

```
import org.apache.catalina.realm.GenericPrincipal;
import org.apache.catalina.users.MemoryUser;
import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.security.Principal;
import java.util.Arrays;
import java.util.List;

public class ExampleFilter implements Filter{

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {}

    @Override
    public void doFilter(final ServletRequest servletRequest, final
ServletResponse servletResponse, FilterChain filterChain) throws
IOException, ServletException {
        if (!(servletRequest instanceof HttpServletRequest ||
!(servletRequest instanceof HttpServletResponse))) {
            return;
        }

        final HttpServletRequest request = (HttpServletRequest)
servletRequest;
        final HttpServletResponse response = (HttpServletResponse)
servletResponse;
        final String username = "username";
        final String password = "password";
        final List<String> groups = Arrays.asList("Group1", "Group2");
        final GenericPrincipal principal = new GenericPrincipal(username,
password, groups);
        filterChain.doFilter(new FilterRequestWrapper(request, principal),
response);
    }

    @Override
    public void destroy() {}
}
```

3. When these classes have been created, you can compile them and package them in a jar file.
4. Copy the jar file to the `WEB-INF/lib` folder in the `panopticon` war file (or the extracted folder).
5. Enable the filter by adding the following code to the `web.xml` file in `panopticon WEB-INF` folder:

```
    <filter>
        <filter-name>ExampleFilter</filter-name>
```

```
        <filter-class>com.datawatch.server.filter.ExampleFilter</filter-
class>
    </filter>
    <filter-mapping>
        <filter-name>ExampleFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
```

HEADER

It is possible to use a web-facing Panopticon Streams behind a proxy server that will handle the authentication of users. The proxy server forwards the name of the user and roles to Panopticon Streams as HTTP headers for every request.

For requests where headers are blank or missing, they are treated like anonymous requests while requests where the user HTTP header are valid are treated like authenticated requests with that specific username.

Requests from the proxy server are fully trusted and checks are no longer performed at Panopticon Streams with regard to the validity of the username. The authorization on workbooks and administration will work as usual.

To activate the Header authentication, add or update the following properties in the `Streams.properties` file:

```
authentication.type=HEADER
authentication.header.role.delimiter=,
authentication.header.roles={roles header, ie. X-Roles}
authentication.header.username=={userid header, ie. X-User}
```

[4] PCLI: COMMAND UTILITIES FOR PANOPTICON

Panopticon Streams is supplied with a command line utility PCLI.jar.

EXPORT DATA SOURCES

THE PCLI provides functionality to export data sources from one or all workbooks in a directory. The exported data sources can be uploaded and used directly by Panopticon Streams.

Parameters

Parameter	Description	Required
-w, --workbook	The name of the workbook.	Yes (or -wd)
-od, --output-directory	The output directory where the data source will be exported to.	No
-wd, --workbook-directory	The directory of the workbooks folder.	Yes (or -w)
-dd, --data-directory	The directory of the data folder.	Yes
-l, --license-file	The path of the license file.	Yes

Example 1: Export data sources from a workbook

```
java -jar pcli.jar exportdatasource
-w "C:/vizserverdata/Workbooks/VizGuide.exw"
-l "C:/vizserverdata/PanopticonLicense.xml"
-dd "C:/vizserverdata/Data"
-od "C:/streamsserverdata/CEP/Datasources"
```

Example 2: Export data sources from all workbooks example

```
java -jar pcli.jar exportdatasource
-wd "C:/vizserverdata/Workbooks"
-l "C:/vizserverdata/PanopticonLicense.xml"
-dd "C:/vizserverdata/Data"
-od "C:/streamsserverdata/CEP/Datasources"
```

Where:

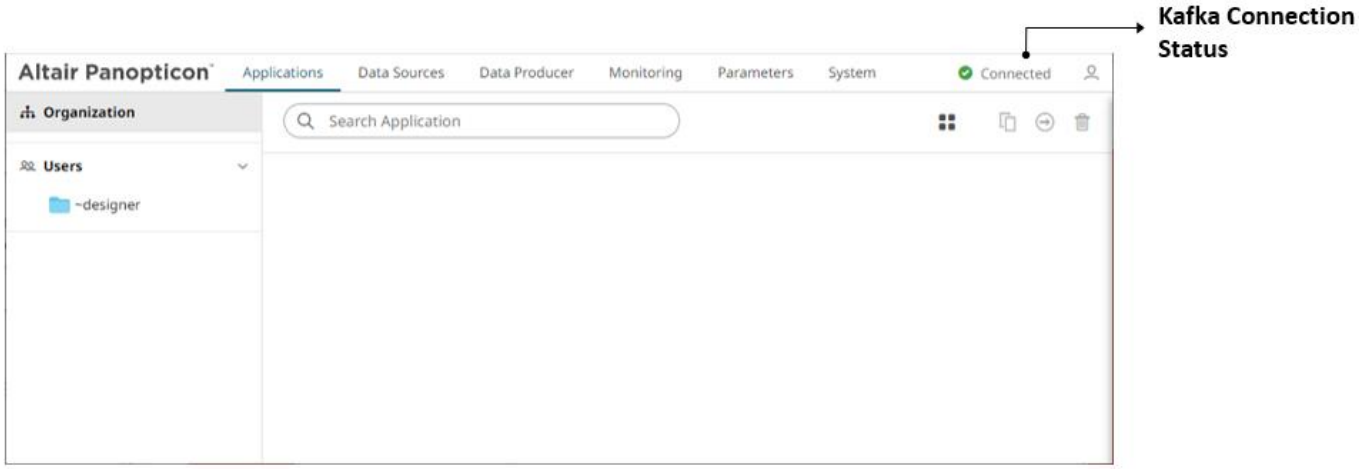
- `C:\vizserverdata` is the `AppData` folder of Panopticon Real Time
- `C:\streamsserverdata` is the `AppData` folder of the Streams server

[5] USING ALTAIR PANOPTICON STREAMS

CONNECTING TO OR DISCONNECTING FROM THE CEP ENGINE

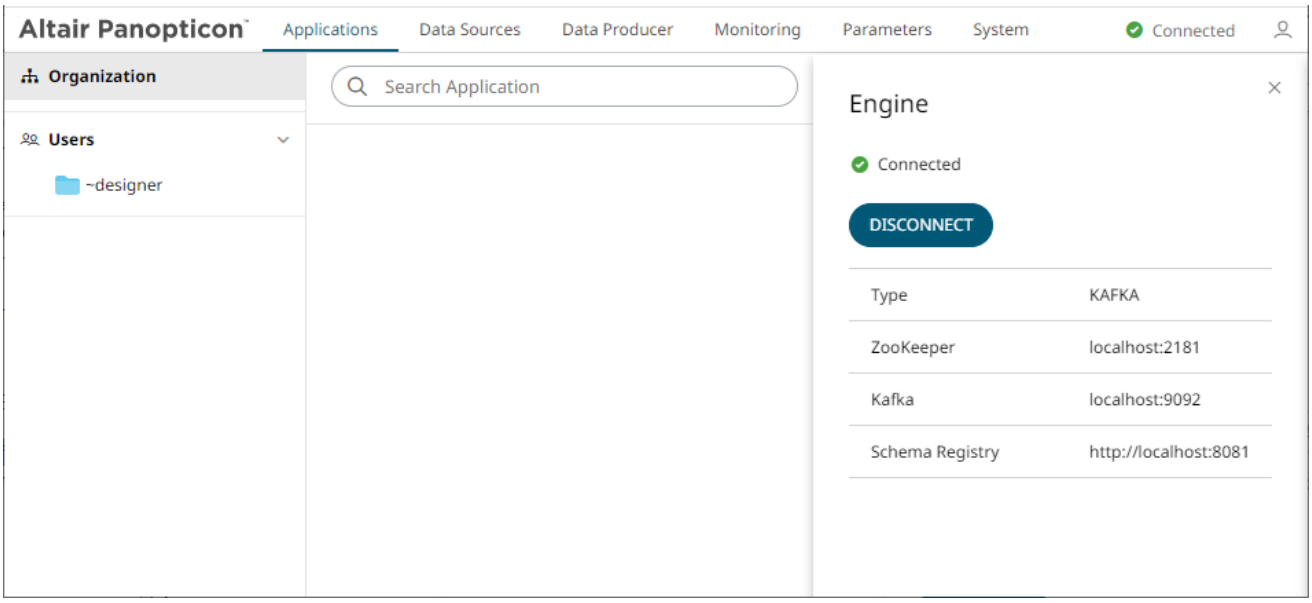
NOTE **Panopticon Streams Server will be connected to the CEP engine after start up if any of the following settings is true:**

- **The default setting of the localhost for the Kafka broker is available.**
- **The Kafka settings in the Streams.properties file are correct.**

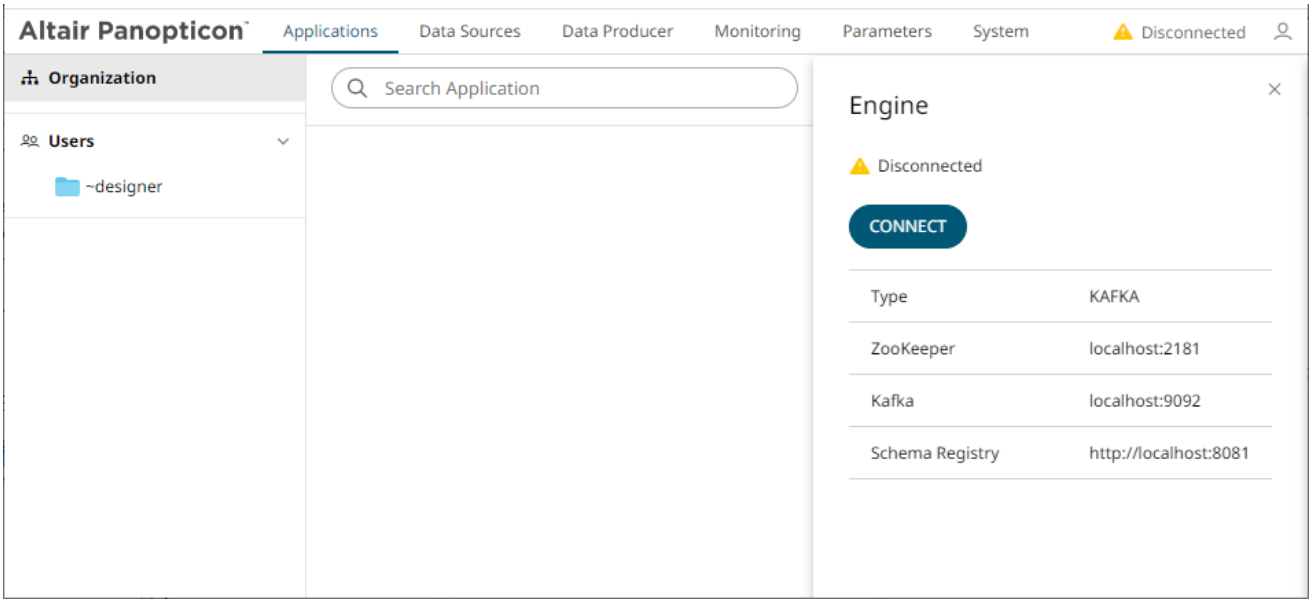


Click **Kafka Connection Status** to expand and display the *Engine* panel and view the settings.

For Connected :



For Disconnected :



Property	Description
Status	Displays whether Panopticon Streams is connected to or disconnected from the CEP Engine (Kafka).
Type	The CEP Engine that Panopticon Streams engine will work with (**KAFKA**).
ZooKeeper	The URL to the ZooKeeper servers. Default is `localhost:2181`.
Kafka	The URL of all the Kafka servers. Default is `localhost:9092`.

Schema Registry	The URL to the Schema Registry. Default is `http://localhost:8081`.

Connecting to the CEP Engine

Starting with version 2021.0, the “local” or “internal” Kafka connectivity is deprecated. To connect to the CEP engine, use the external setup.

> **NOTE**
>
> **Before connecting to the CEP engine, ensure the following are running:**
>
> - **Confluent Kafka Enterprises services if you are using a Dockerized Kafka.**
> - **ZooKeeper, Kafka, and Schema Registry batch files if you are using a local cluster.**

Click CONNECT to connect to the external Kafka.

Disconnecting from the CEP Engine:

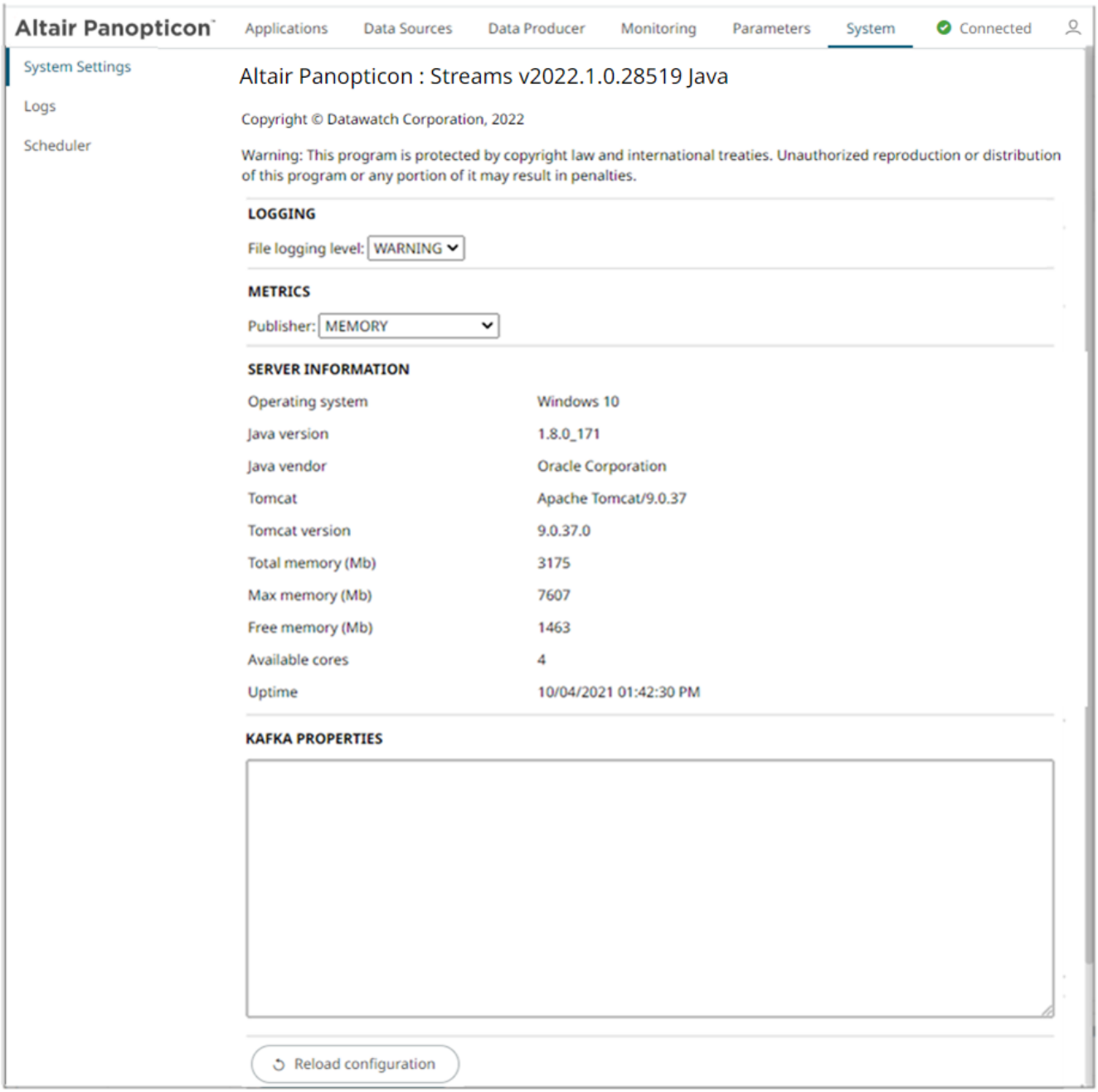
Click DISCONNECT. Consequently, the applications cannot be started and the input and output topics will not be generated.

[6] MANAGING THE STREAMS SYSTEM

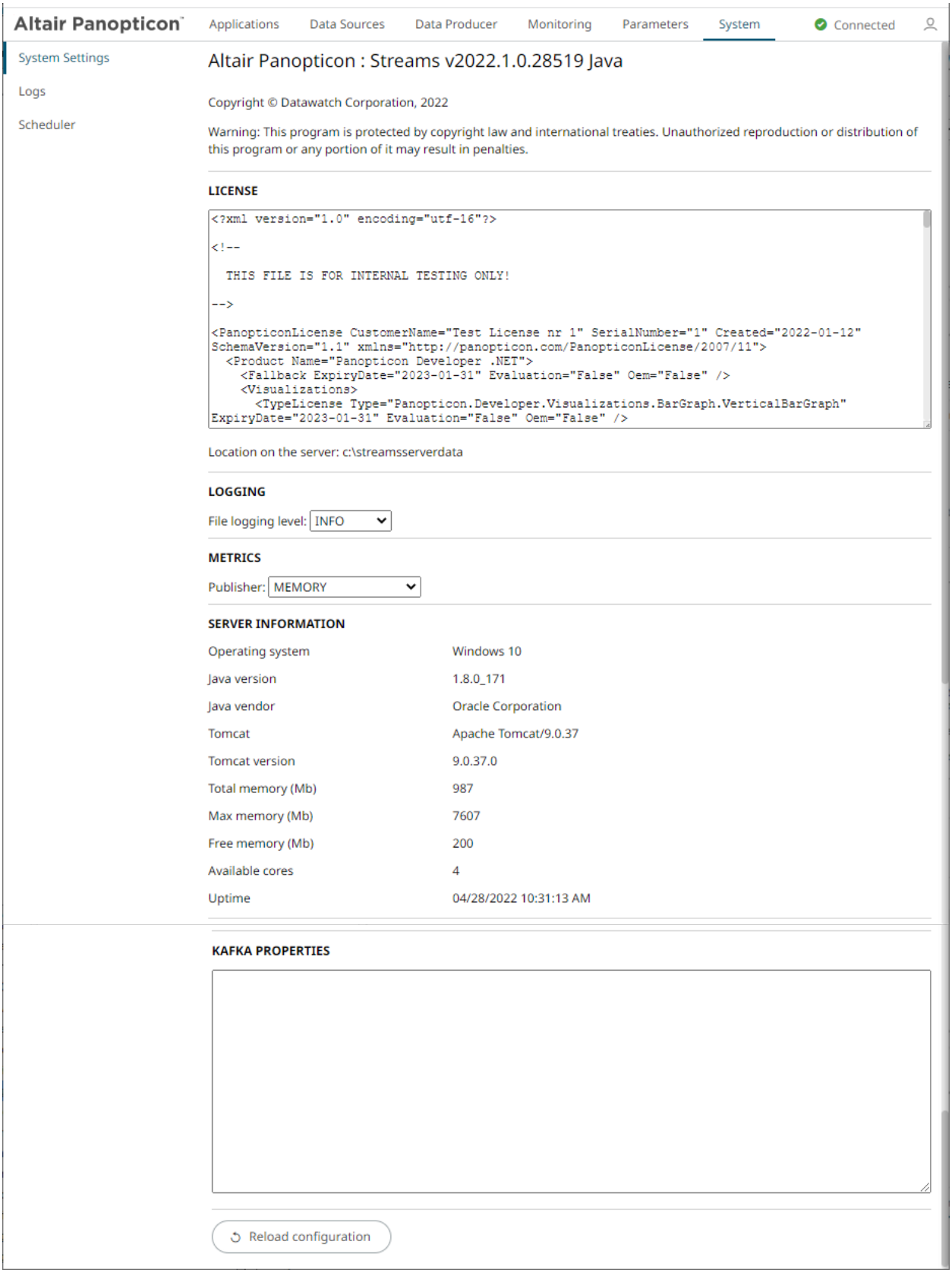
The **System** tab displays the following sections where an administrator can:

- view the active license
- monitor and set the File Logging Level
- select the Metrics Publisher
- view the server properties
- view Kafka properties
- reload configurations
- schedule tasks

If the licensing used is the Altair Units license, the page will be displayed as:



If the licensing used is the volume-based XML file (named **PanopticonLicense.xml**), the page will be displayed as:

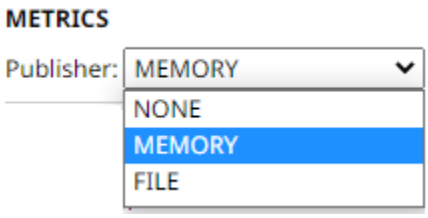

Altair Panopticon
Applications
Data Sources
Data Producer
Monitoring
Parameters
System
Connected
System Settings
Logs
Scheduler
Altair Panopticon : Streams v2022.1.0.28519 Java
Copyright © Datawatch Corporation, 2022
Warning: This program is protected by copyright law and international treaties. Unauthorized reproduction or distribution of this program or any portion of it may result in penalties.
LICENSE
<?xml version="1.0" encoding="utf-16"?>
<!--
THIS FILE IS FOR INTERNAL TESTING ONLY!
-->
<PanopticonLicense CustomerName="Test License nr 1" SerialNumber="1" Created="2022-01-12" SchemaVersion="1.1" xmlns="http://panopticon.com/PanopticonLicense/2007/11">
<Product Name="Panopticon Developer .NET">
<Fallback ExpiryDate="2023-01-31" Evaluation="False" Oem="False" />
<Visualizations>
<TypeLicense Type="Panopticon.Developer.Visualizations.BarGraph.VerticalBarGraph" ExpiryDate="2023-01-31" Evaluation="False" Oem="False" />
Location on the server: c:\streamsserverdata
LOGGING
File logging level: INFO
METRICS
Publisher: MEMORY
SERVER INFORMATION
Operating system Windows 10
Java version 1.8.0_171
Java vendor Oracle Corporation
Tomcat Apache Tomcat/9.0.37
Tomcat version 9.0.37.0
Total memory (Mb) 987
Max memory (Mb) 7607
Free memory (Mb) 200
Available cores 4
Uptime 04/28/2022 10:31:13 AM
KAFKA PROPERTIES
Reload configuration

SETTING THE SERVER METRICS PUBLISHER

The server performance metrics can be used to report, monitor, and configure the server's health and limits. The collected metrics may include the following information:

- Long polling, WebSocket, and total number of connections
- CPU loading percentage
- Maximum, size, and used Heap Bytes
- Subscription alerts, users, and total
- Number of parallel data loading and live threads
- Average data load time or refresh rate

On the *Metrics* section of the *System Settings* page, select the *Publisher* of the server performance metrics.



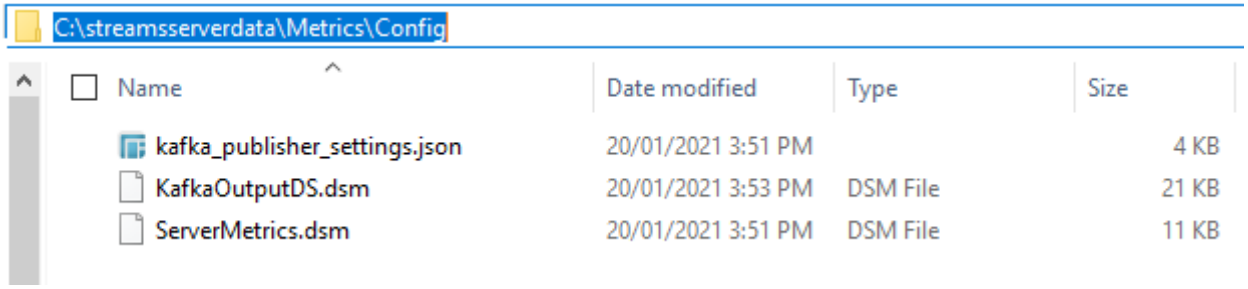
Metrics Publisher	Description
None	No metrics are published.
Memory	Metrics are published to a queue in memory.
File	Metrics are published to a file on disk located in the `AppData/Metrics/` folder (i.e., `c:\streamsserverdata\Metrics`).

To add other Kafka publishers in the drop-down list, ensure their configuration file are available in the `AppData/Metrics/Config` folder.

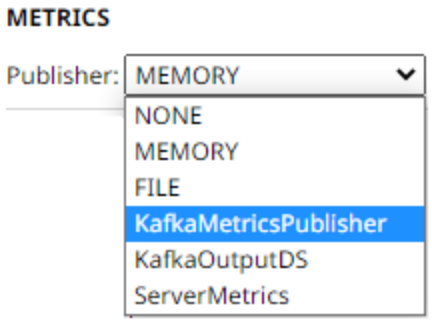
A configuration file can be generated by creating a new data source in Panopticon Streams Server and selecting any of the *Output* connectors. You can either:

- export the JSON file from the repository, or
- download the DSM file by right-clicking the **Data Source** and clicking **Download** on the context menu

For example, when the generated Kafla data sources are added in the `AppData/Metrics/Config` folder:



The ID of the new configuration files are displayed in the *Publisher* drop-down list.



Selecting any of these specific Kafka data sources means that this is only place where metrics will be published to.

VIEWING AND MANAGING KAFKA PROPERTIES

The user-defined `Kafka.properties` file contains properties for controlling the Kafka configurations in Panopticon Streams.

Below is a sample properties file:

```
# Broker endpoints where we will discover the cluster broker members.
If this
# is set here, any results from ZooKeeper are ignored.
# common producer and consumer settings
#bootstrap.servers=localhost:9092
bootstrap.servers=localhost:9093
security.protocol=SASL_PLAINTEXT
sasl.mechanism=PLAIN
sasl.jaas.config=\
  org.apache.kafka.common.security.plain.PlainLoginModule required \
    username="dwchuser" \
    password="dwchpwd";

#Global properties applied on any topic created
topic.retention.ms=50000
topic.cleanup.policy=delete

aggregate.cachingEnabled=true

#Specific operator/node applicationId.operatorId.propertyname
AggregationExample.Input.retention.ms=60000
AggregationExample.Output.retention.ms=30000
AggregationExample.Aggregation.cachingEnabled=false
```

When Panopticon Streams server is started, it checks the `AppData` folder for the `kafka.properties` file and loads the properties in the *Kafka Properties* box.

KAFKA PROPERTIES

```
# Broker endpoints where we will discover the cluster broker members. If this
# is set here, any results from ZooKeeper are ignored.
# common producer and consumer settings
#bootstrap.servers=localhost:9092
bootstrap.servers=localhost:9093
security.protocol=SASL_PLAINTEXT
sasl.mechanism=PLAIN
sasl.jaas.config=\
  org.apache.kafka.common.security.plain.PlainLoginModule required \
    username="dwchuser" \
    password="dwchpwd";

#Global properties applied on any topic created
topic.retention.ms=50000
topic.cleanup.policy=delete
```

However, if the `kafka.properties` file is not available, the *Kafka Properties* box will display a blank *Kafka Properties* box:

KAFKA PROPERTIES

If you opt to copy the `kafka.properties` file to a different location, open the `Streams.properties` file and set the attribute `cep.kafka.properties` to the value of the file path along with the Kafka properties file name. For example:

```
cep.kafka.properties=c:\kafkafile\kafka.properties
```

NOTE

- **The values in the *Kafka Properties* box is not editable on the System tab. Changes can be made in the actual `kafka.properties` file. To reload the properties on the System tab, click Reload Configuration.**
- **The `kafka.properties` file supports any Kafka configurations available on their documentation**
- **The configurations made in the `kafka.properties` will supersede any of the Kafka-related properties in the `streams.properties` file**
- **Some of the configurations in the `kafka.properties` file can be overridden by the settings made in Panopticon Streams applications**

RELOADING CONFIGURATIONS

On the *System Settings* page under the **System** tab, click Reload configuration. This will stop and restart applications, reload data sources, and Kafka properties along with the administrators and parameters from the file system.

LOGGING/MONITORING

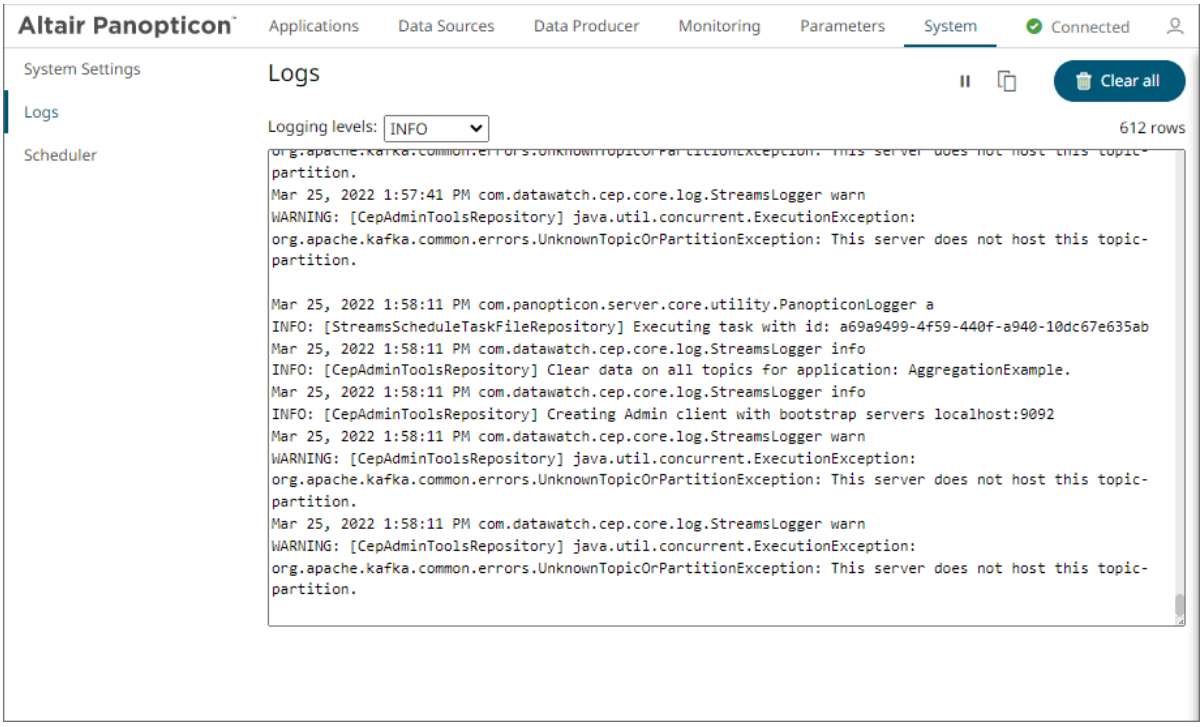
View Logs

View the latest 300 rows of a *Logging Level* on the **Logs** tab:

- FINEST (lowest level)
- FINER
- FINE
- CONFIG
- INFO (default level)
- WARNING
- SEVERE (highest level)

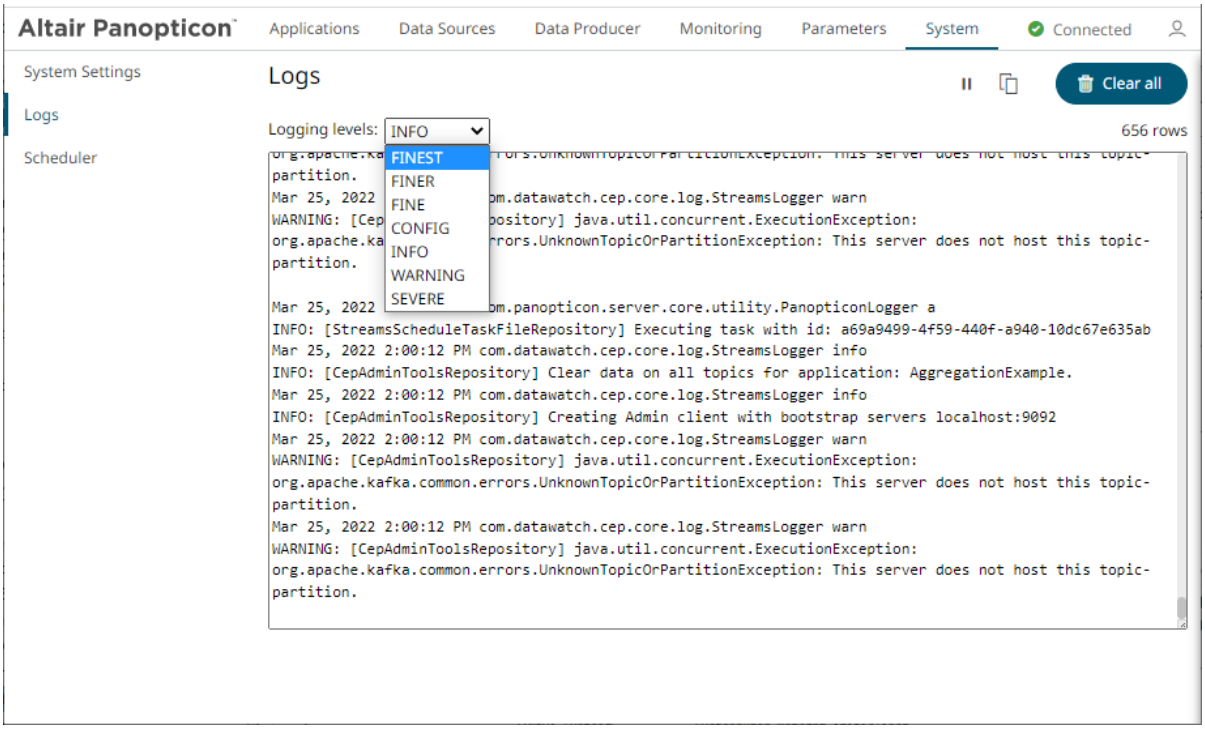
Steps:

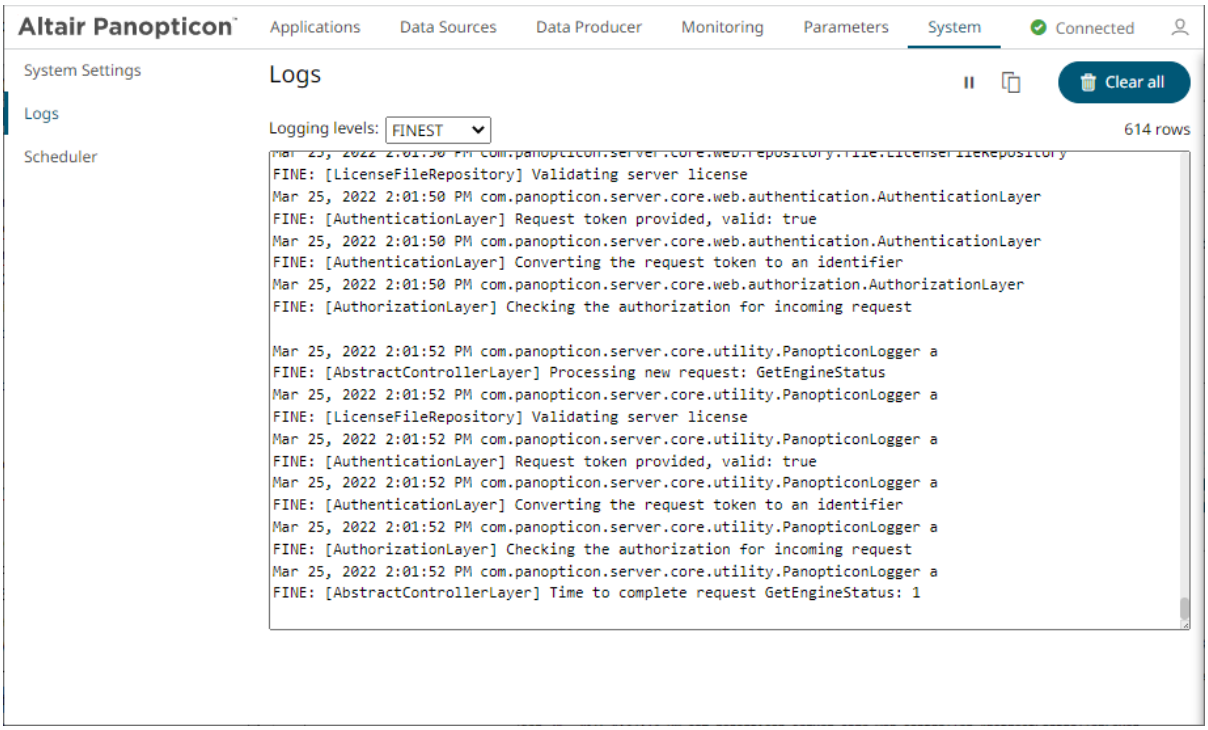
1. Under the **System** tab, click the **Logs** link. Initially, the default level (**INFO**) logs are displayed.



2. Select another *Logging Level* in the drop-down.

   For example: **FINEST**





The latest 300 rows of the selected log level or higher are fetched.

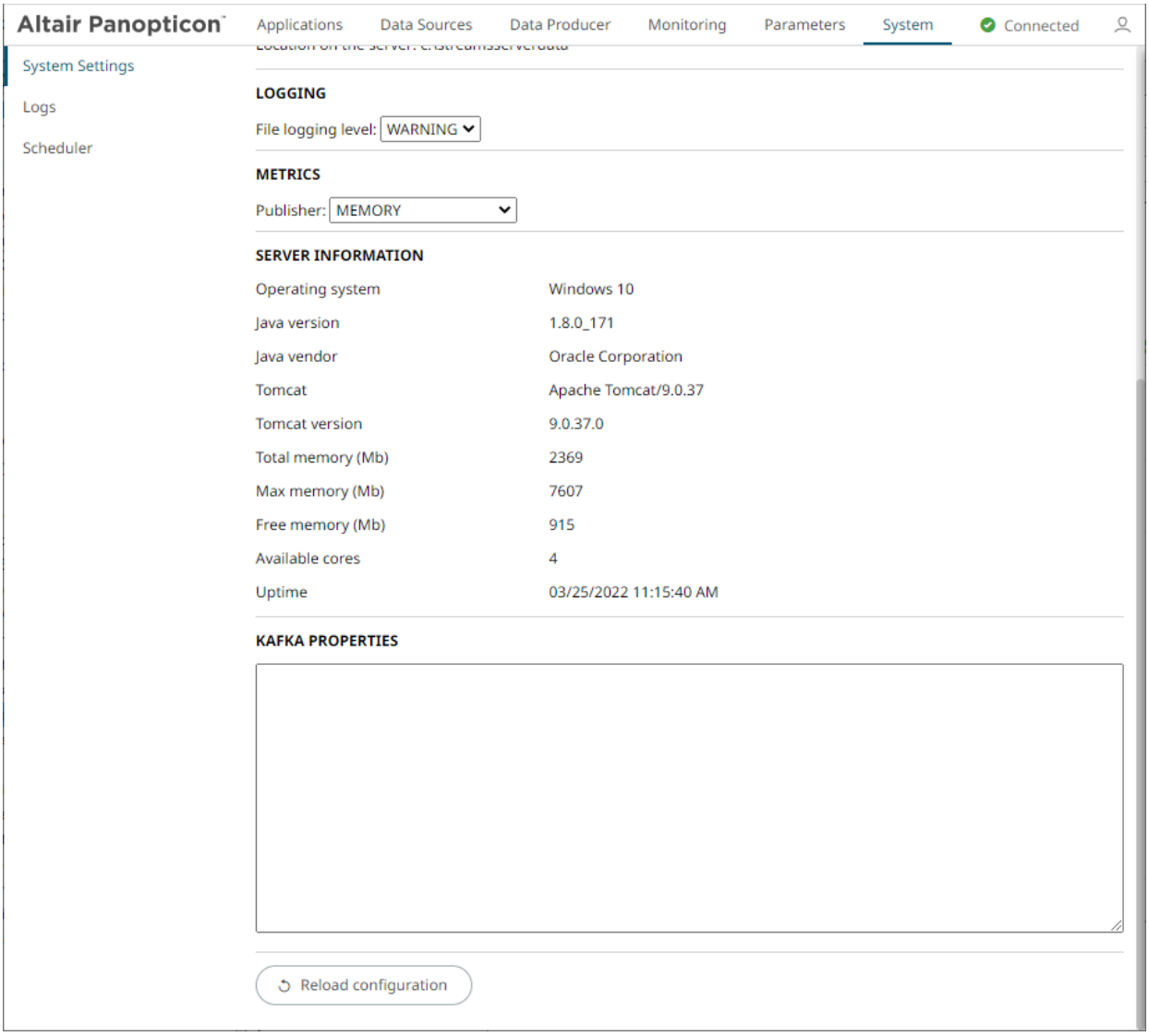
3. You can also click any of the following buttons:

- to pause the logging, it changes to

- to resume the logging
- to copy log to clipboard
- Clear all to clear the logs

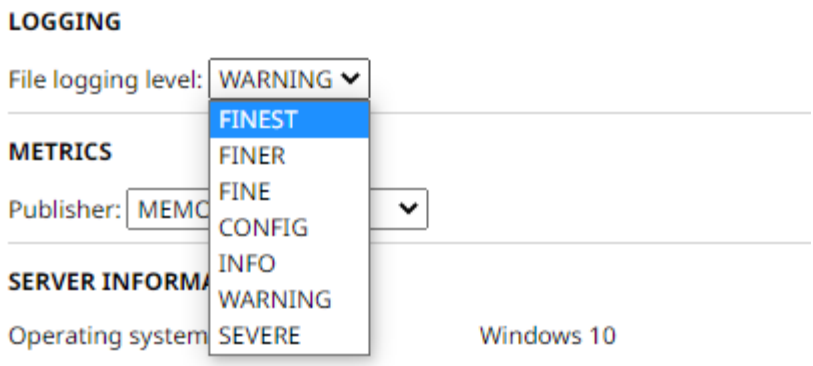
Set File Logging Level

On the *System Settings* page under the **System** tab, the level that is logged to file can be set.



Steps:

1. The current set level (e.g., **WARNING**) is displayed. To change, click the drop-down list and select another log level.



The new log level is written in the **`Streams.properties`** file:

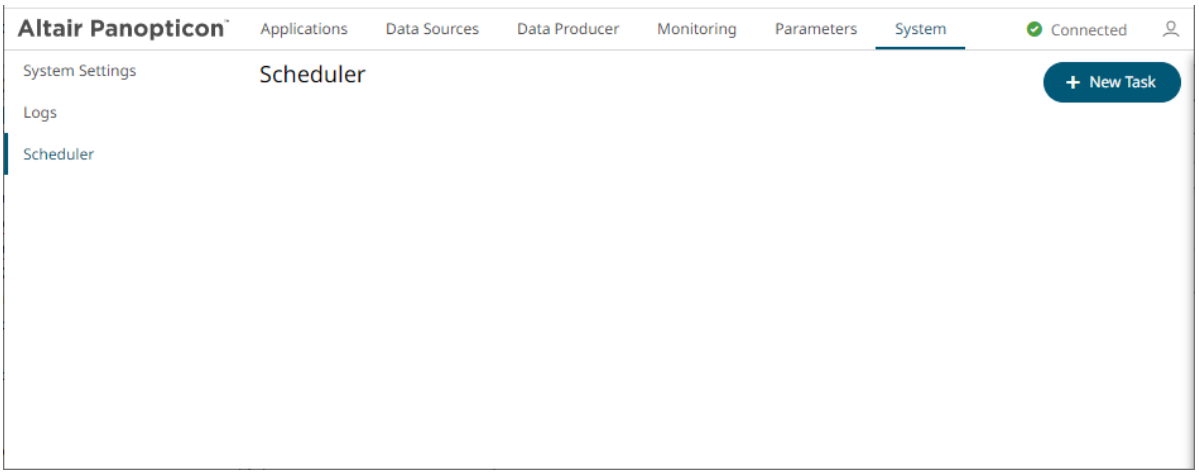
```
logger.level.file=FINEST
```

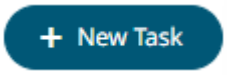
SCHEDULING TASK TO CLEAR TOPIC DATA

Panopticon Streams supports scheduling of tasks such as daily deletion of application topics.

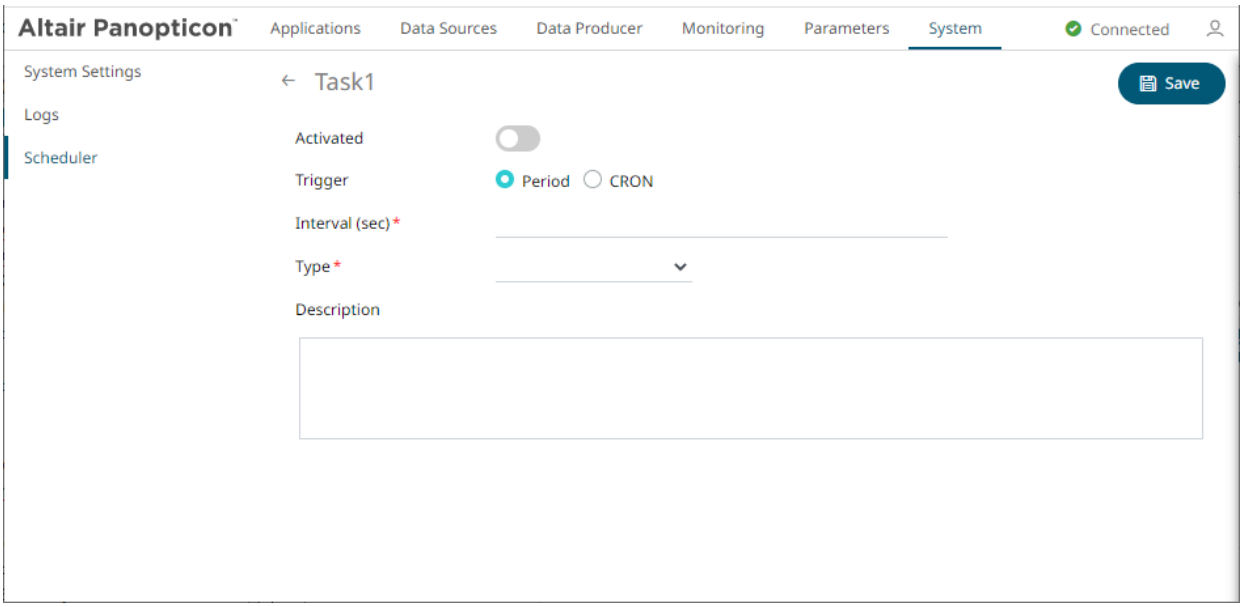
Steps:

1. Under the **System** page, click the **Scheduler** tab.

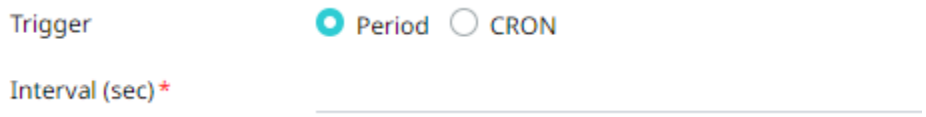


2. Cilck **New Task** 

. The *New Task* pane displays.

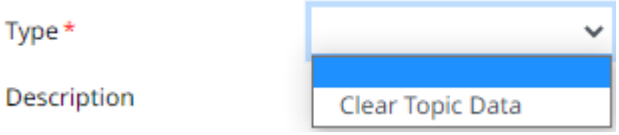


3. Enter the *Name* of the task and click ✓. Ensure the name is unique.
4. Tap the *Activated* slider to turn it on.
5. Select the *Trigger.* You can either select:
   - **Period** then enter the *Interval* (in seconds), or



   - **CRON** then enter the CRON Expression

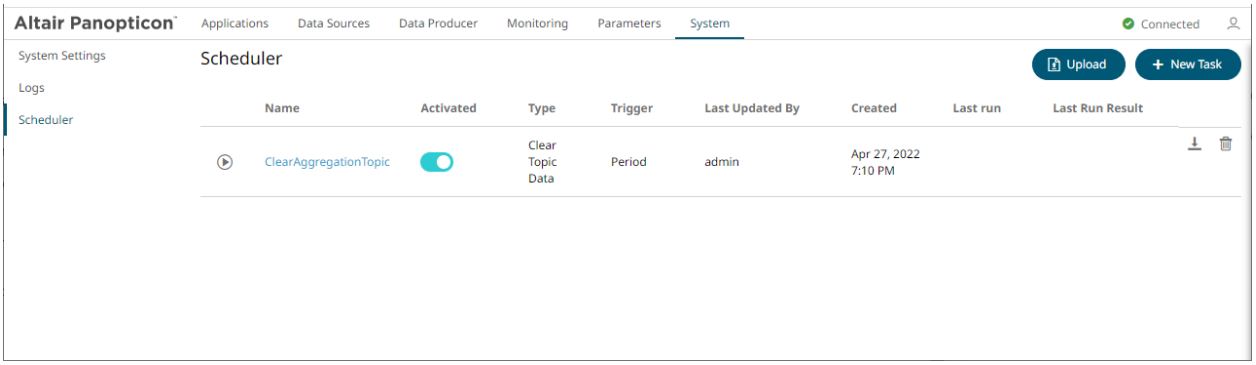
6. Select the task *Type*: **Clear Topic Data**.



7. Enter the *Description* of the task.
8. Select the *Application* in the drop-down list. These are the applications available on the **Applications** tab.
9. Click

.

   Click ← to go back to the *Tasks* pane. The new task is added in the list.



A task displays the following columns: *Name*, *Activated, Type, Trigger, Last Updated By, Created*, *Last Run,* and *Last Run Result.*

Modify the sorting of the list by clicking the ↓ or ↑ button of any of these columns. The icon beside the column that was used for the sorting will indicate if it was in an ascending or descending order.

Tasks can also be:

- manually started

  Instead of waiting for the set Period interval or CRON Expression, you can manually execute the task by clicking ▶. A confirmation message displays. Click Yes.

  The *Last Run* and *Last Run Result* (**Success** or **Failed**) are displayed. For failed results, you can hover on the tooltip to view the error.

- modified

- deleted

  Click 🗑 of a task. A confirmation message displays. Click Yes.

Modify a Scheduled Task

Steps:

1. On the *Scheduler* tab under the on the **System** page, click the link of a task to modify.

   The properties of the task are displayed.

2. Apply the desired changes.

3. Click Save.

[7] AUTHORIZATION

NOTE **Starting with version 2020.0, mapping of administrators through `Administrators.txt` and `AdministratorGroups.txt` is no longer supported. The property `access.administrator.groups` should be used instead.**

If the customer's authentication method relied to the use of the `Administrators.txt` or `AdministratorGroups.txt` file, they can still do so by additionally using the `tomcat-users.xml` to replicate the usage of these administrator text files.

For example, in the `tomcat-users.xml`, they can assign groups from the administrator text files to specific users like this:

```
<user username="admin" password="admin" roles="role1,otherRole"/>
<user username="admin2" password="admin2" roles="role2"/>
```

Then in the `Streams.properties` file, use the `access.administrator.groups` property to map the admins (i.e., admin and admin2) to the administrator groups by adding their roles:

```
access.administrator.groups=role1,role2
```

SECURE ACCESS

Panopticon applications and data sources published to the folders or subfolders in Panopticon Streams Server can be secured by granting allowed or denied permissions.

Creating Folders

A user with an Administrator or Designer role can create folders.

NOTE **Users that log on with a Designer role will have their own personal folder created and displayed on the Applications and Data Sources tabs (e.g., ~designer).**

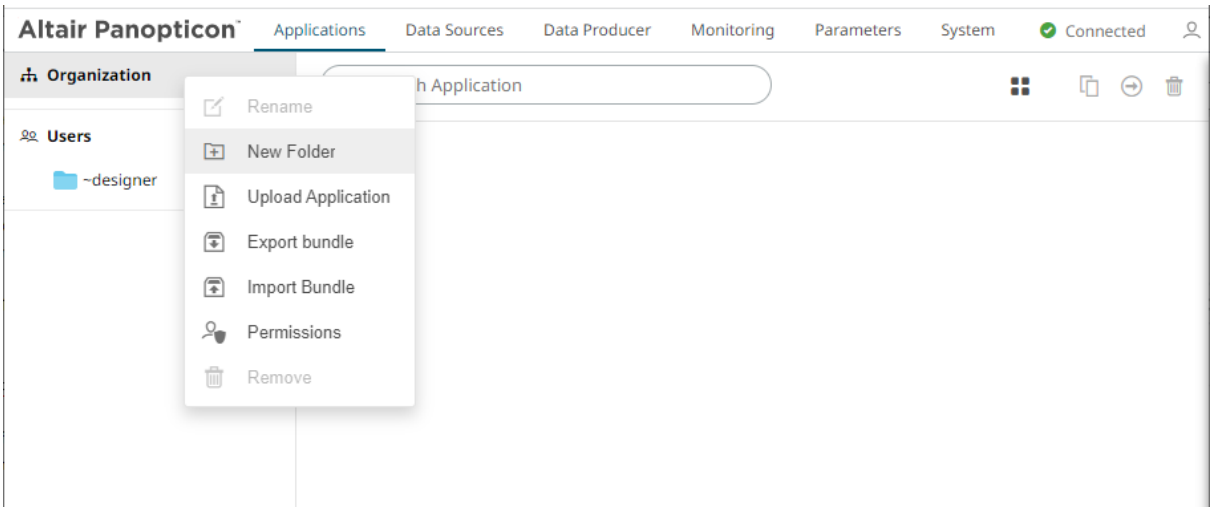
The personal folders:

- **are displayed and can be accessed for users with an Administrator or Designer role.**
- **are where Designers can create applications and data sources. For more information, refer to the Creating a New Application or Creating a Data Source sections.**

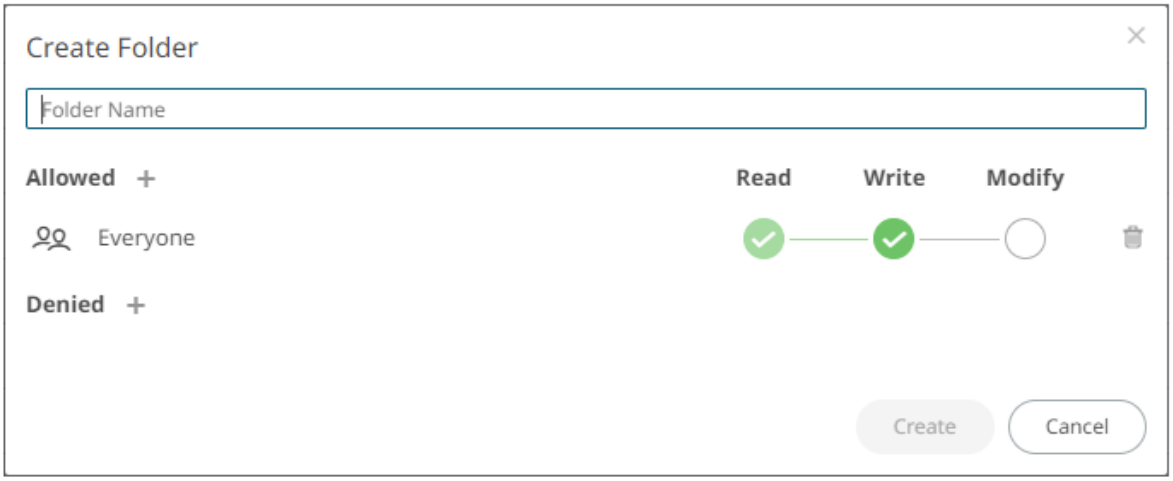
Creating Folders on the Applications Tab

Steps:

1. On the **Applications** tab, right-click on the topmost folder (**Organization**) or the *Applications* pane and select **New Folder**.



The *Create Folder* dialog displays.



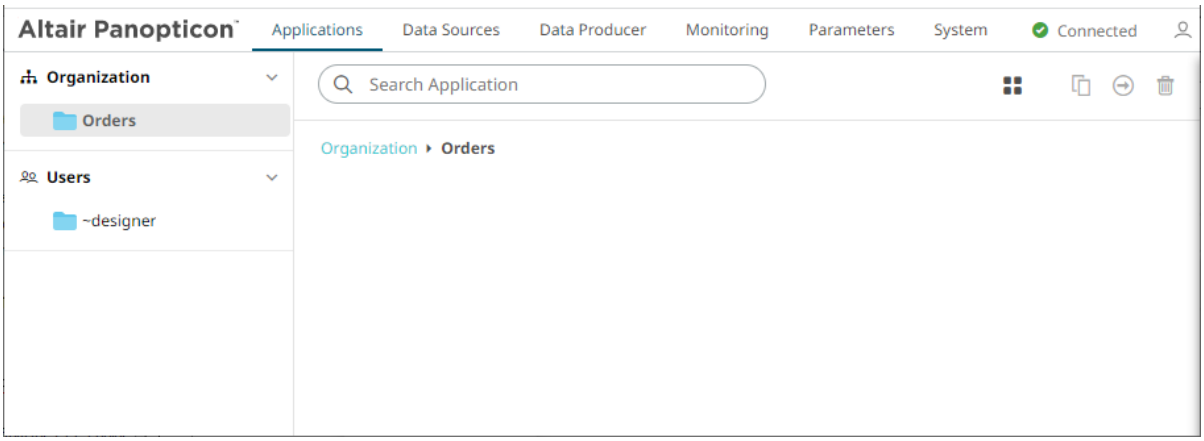
NOTE

- **Everyone is available in the *Allowed* section by default.**
- **Removing the Everyone group will mean that the folder and its subfolders will not be available for public access.**

2. Enter a *Folder Name*.
3. Proceed to defining the Authorization to *Allowed* or *Denied* groups and users.
4. Click

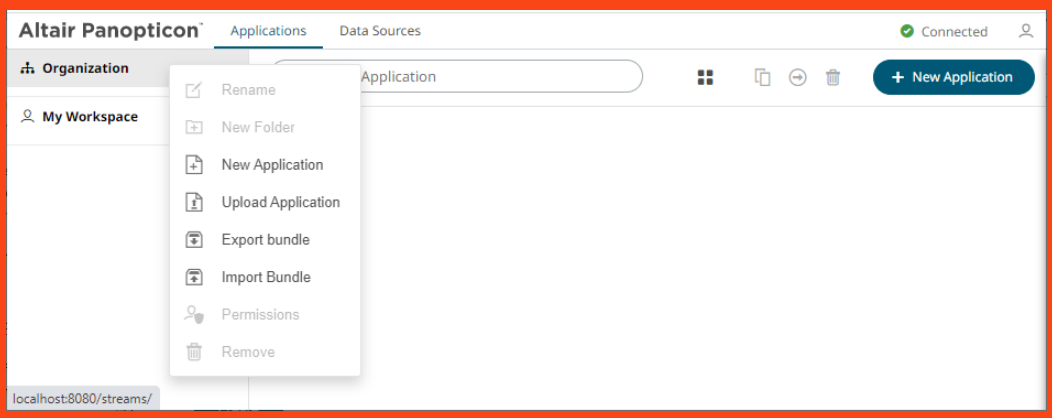
.

   The new folder is displayed on the expanded *Folder* hierarchy list and on the *Root Folder* list.



NOTE

- **A user with a Designer role is not allowed to create a folder on the root folder.**

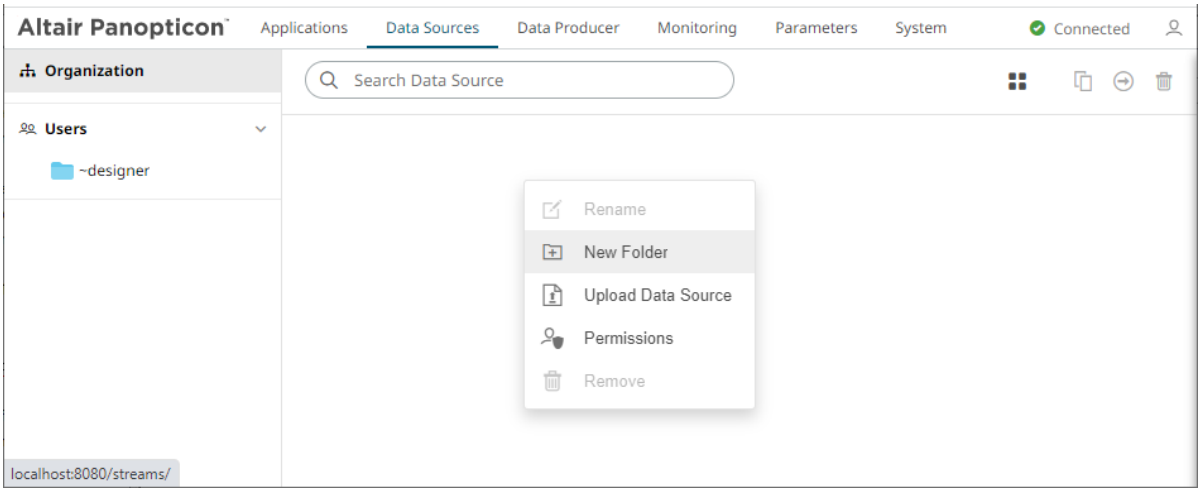


- **Folders and subfolders can be deleted as long as they do not contain applications.**
- **The folders and subfolders on the Applications tab will also be available on the Data Sources tab.**

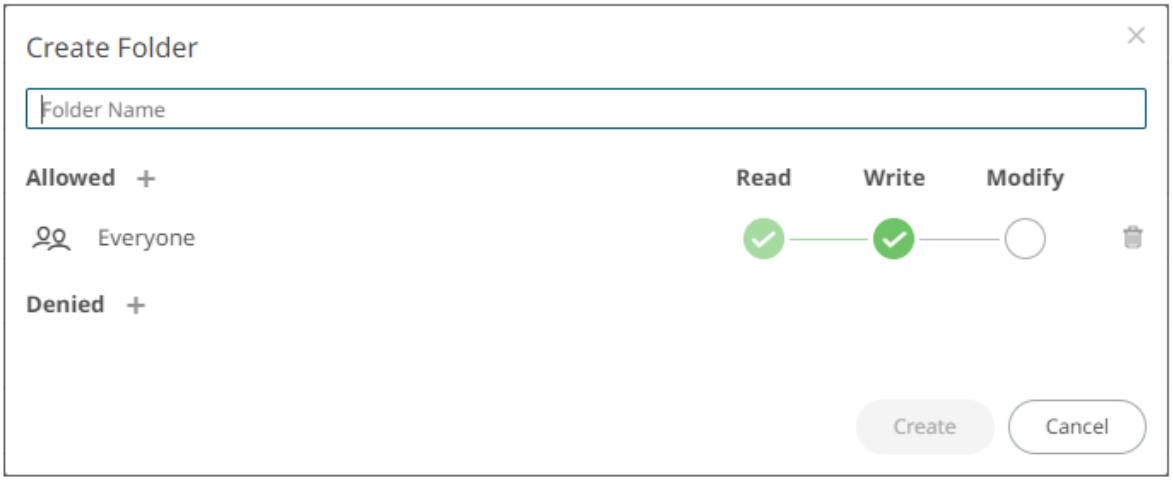
Creating Folders on the Data Sources Tab

Steps:

1. On the **Data Sources** tab, right-click on the topmost folder or the *Data Sources* pane and select **New Folder**.



The *Create Folder* dialog displays.



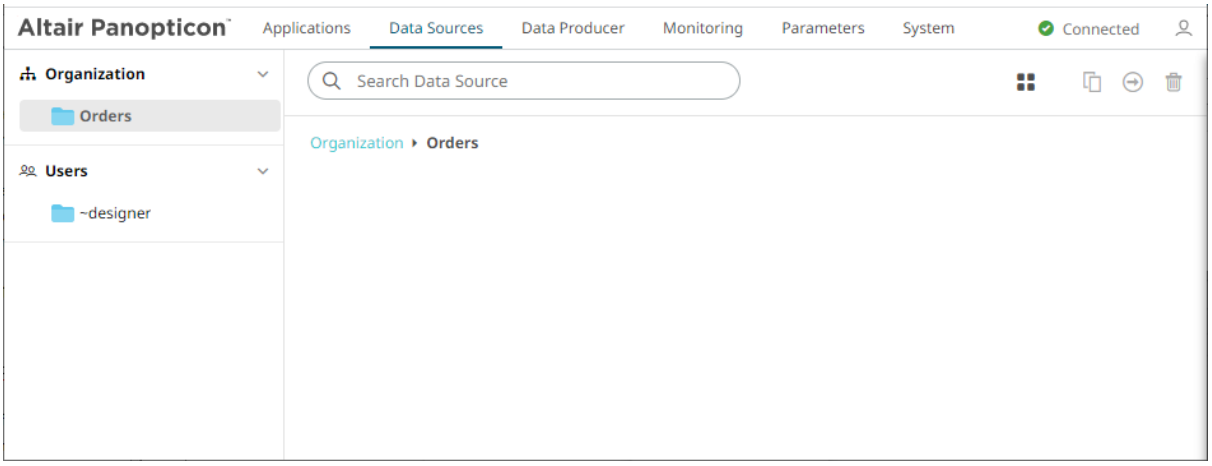
NOTE

- **Everyone is available in the *Allowed* section by default.**
- **Removing the Everyone group will mean that the folder and its subfolders will not be available for public access.**

4. Enter a *Folder Name*.
5. Proceed to defining the Authorization to *Allowed* or *Denied* groups and users.
6. Click

.

   The new folder is displayed on the expanded *Folder* hierarchy list and on the *Root Folder* list.



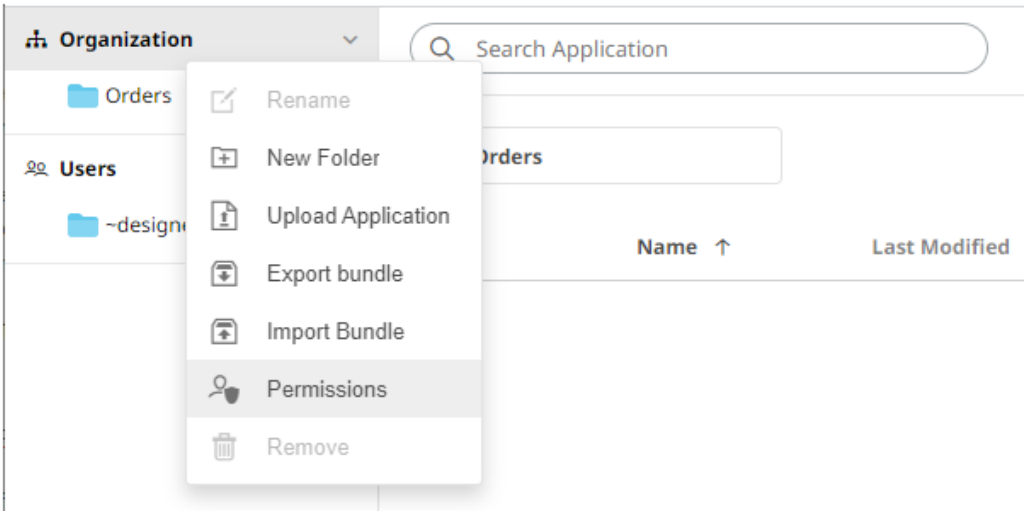
NOTE **Folders and subfolders can be deleted as long as they do not contain data sources.**

Adding Groups and Users with Allowed Authorization

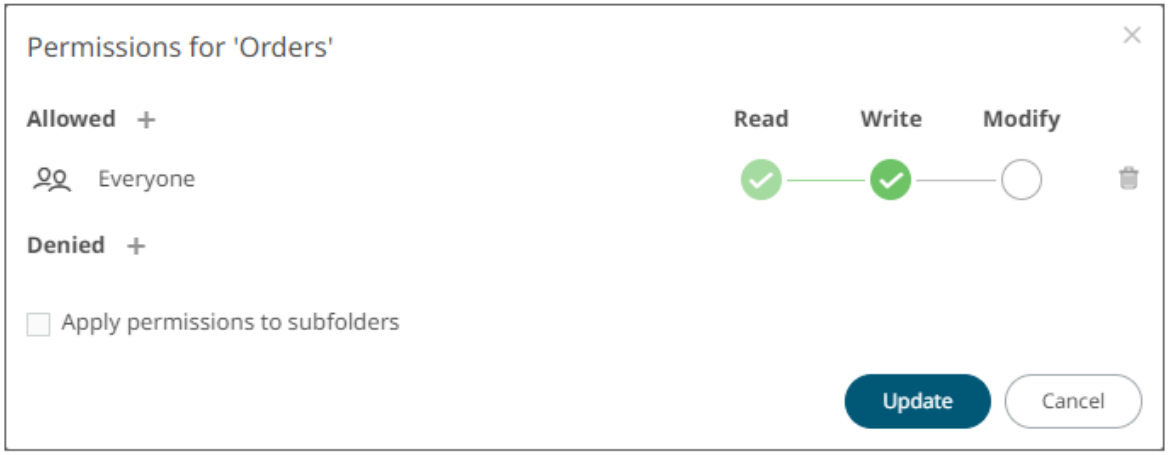
A user with an Administrator or Designer role can grant permission for users or groups access to application or data source folder or subfolder.

Steps:

1. Right-click on a folder (except the root folder) and select **Permissions** on the context menu.

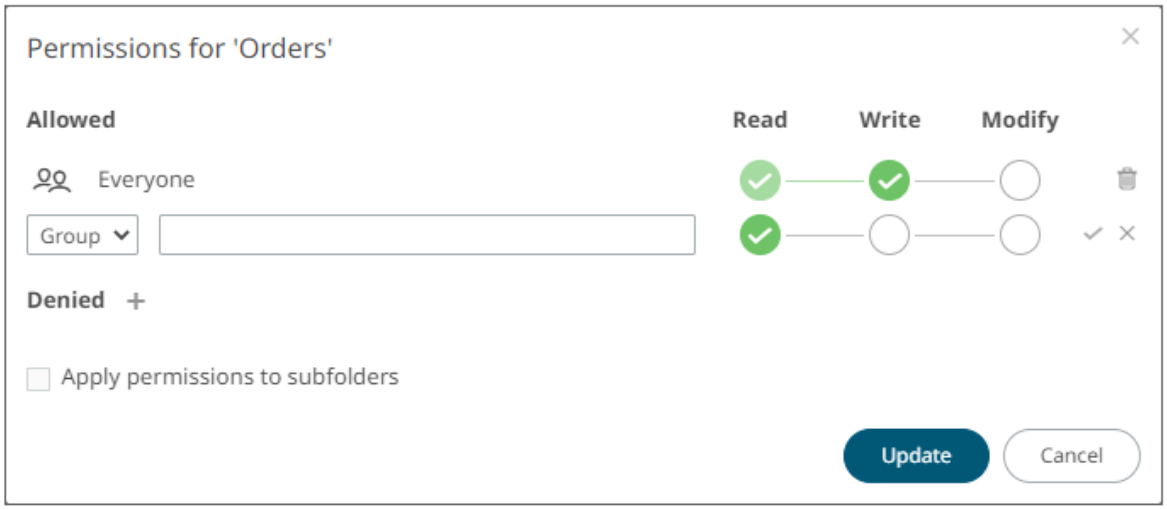


The *Permissions* dialog displays.



2. Under the *Allowed* section, click the **Add** icon.

   A new *User/Group Allowed* section is displayed.



3. Select **User** or **Group** to be given permission in the drop-down list.

4. Enter the user or group *Name*.
5. Select the permission level that will be granted to the user or group:
   - READ

     Permission to read the folder.
   - READ + WRITE

Permission to write to the folder and read.

- MODIFY + WRITE + READ

  Permission to read, modify, and write to the folder as well as create subfolders.

6. Click ✓. The user or group is added under the *Allowed* list.

7. You can either:
   - check the **Apply Permissions to Subfolders** box

   This means the permissions that will be used on all of the subfolders will be fetched from the root folder.

   NOTE **The Apply Permissions to Subfolders check box is only enabled when there is an existing subfolder.**

   - leave the **Apply Permissions to Subfolders** box unchecked and modify the permission properties of the subfolders

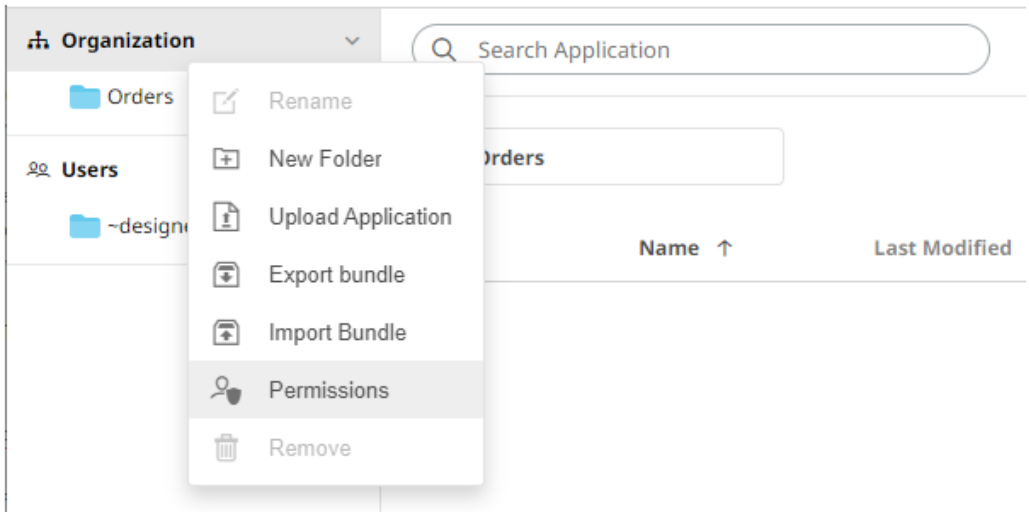
8. Click

to save the changes.

Adding Groups and Users with Denied Access

Steps:

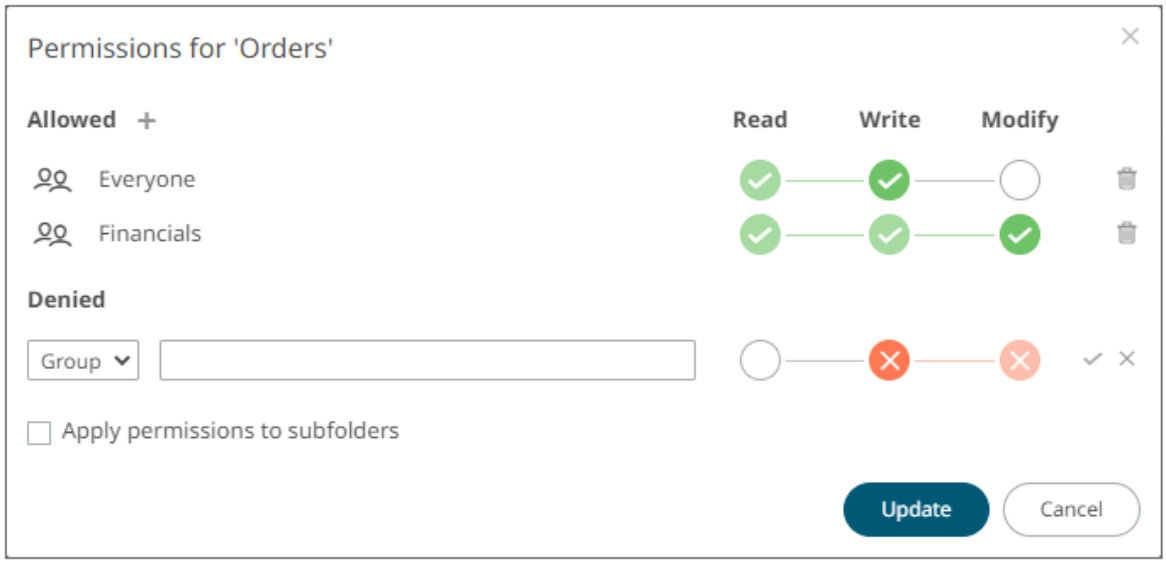
1. Right-click on a folder and select **Permissions** on the context menu.



The *Permissions* dialog displays.

2. Under the *Denied* section, click the **Add** icon.

   A new *User/Group Denied* section is displayed.



3. Select **User** or **Group** that will be given denied permission in the drop-down list.
4. Enter the user or group *Name*.
5. Select the denied permission level that will be granted to the user or group:
   - MODIFY

     Prevent user or group to modify and create subfolders.
   - WRITE + MODIFY

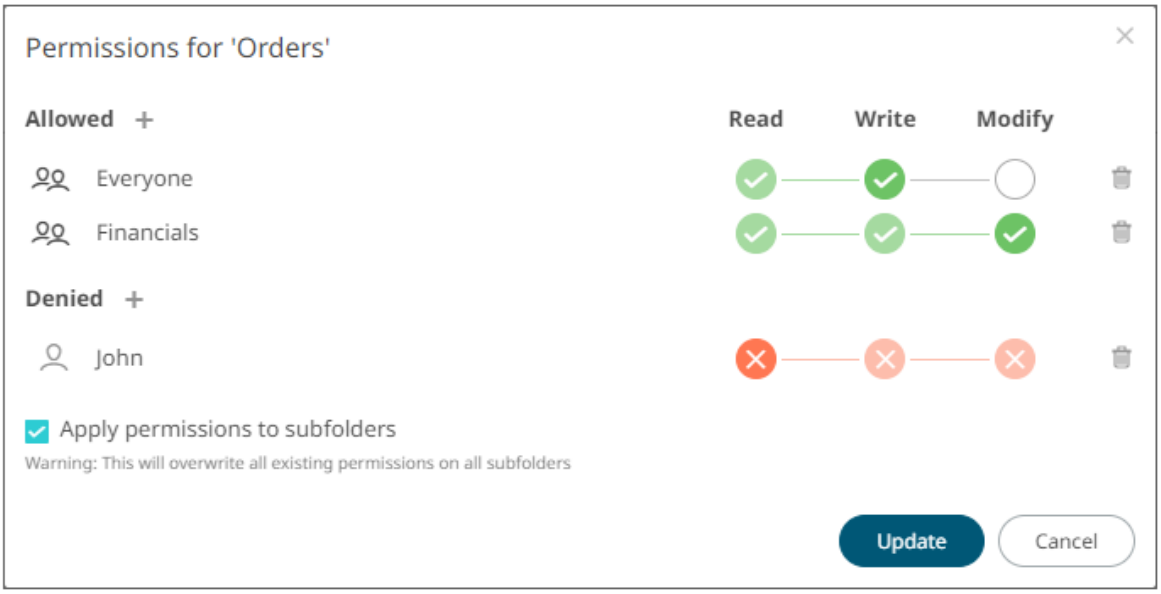
     Prevent user or group to modify and write to the folder.
   - READ + WRITE + MODIFY

     Prevent user or group to modify and create subfolders, modify and write to the folder, as well as read the folder.

6. Click ✓. The user or group is added under the *Denied* list.

Repeat until all of the users with denied access are added.

7. You can either:
   - check the **Apply Permissions to Subfolders** box, or



This means the permissions that will be used on all of the subfolders will be fetched from the root folder.

NOTE **The Apply Permissions to Subfolders check box is only enabled when there is an existing subfolder.**

- leave the **Apply Permissions to Subfolders** box unchecked and modify the permission properties of the subfolders.

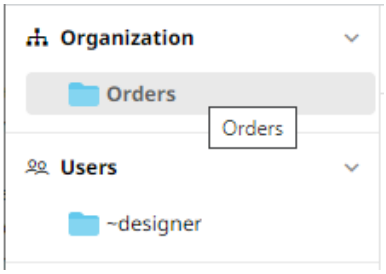
8. Click

to save the changes.

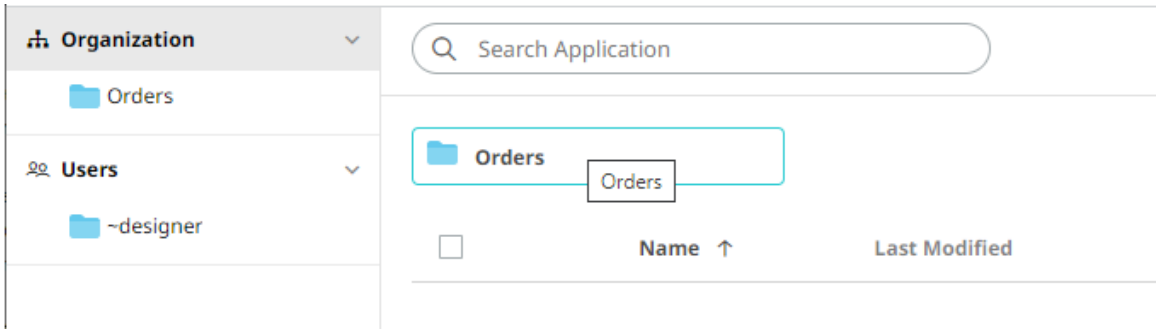
Creating Subfolders

Steps:

1. To create subfolders, you can either click a folder:
   - on the expanded *Folder* hierarchy list, or

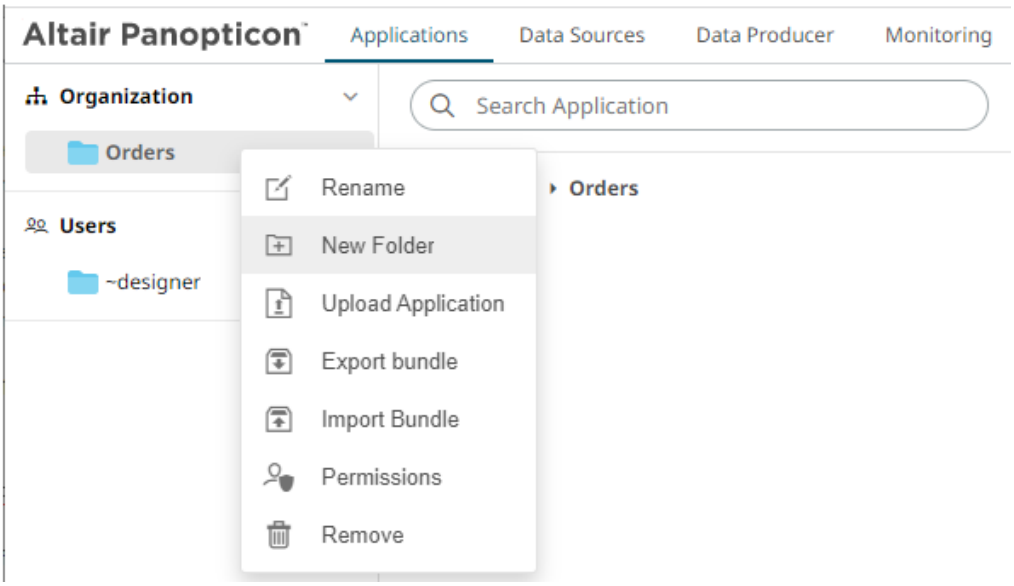


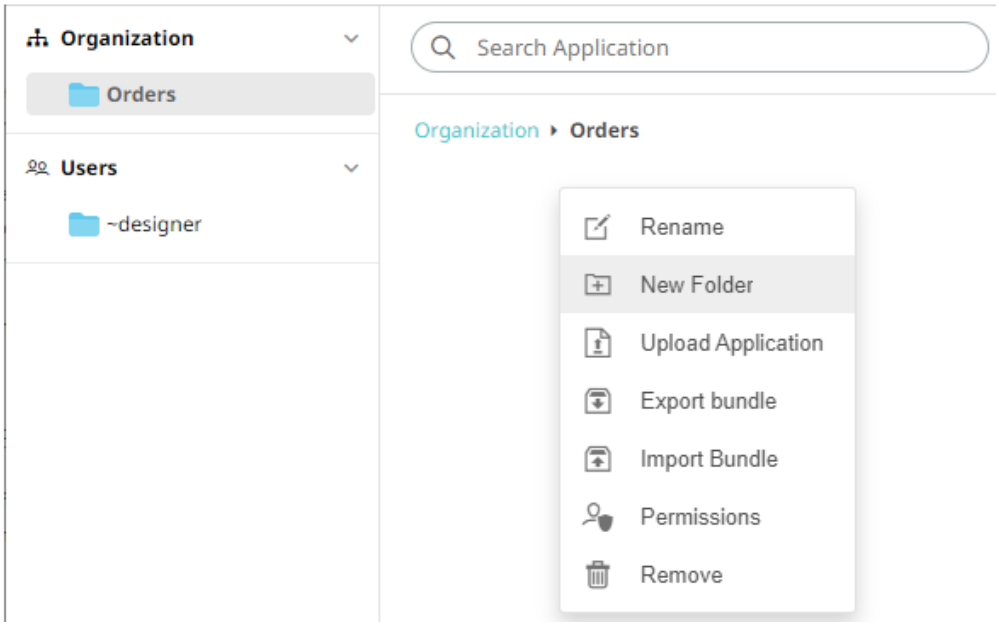
- on the Root folders list.



The *Folders* page is displayed.

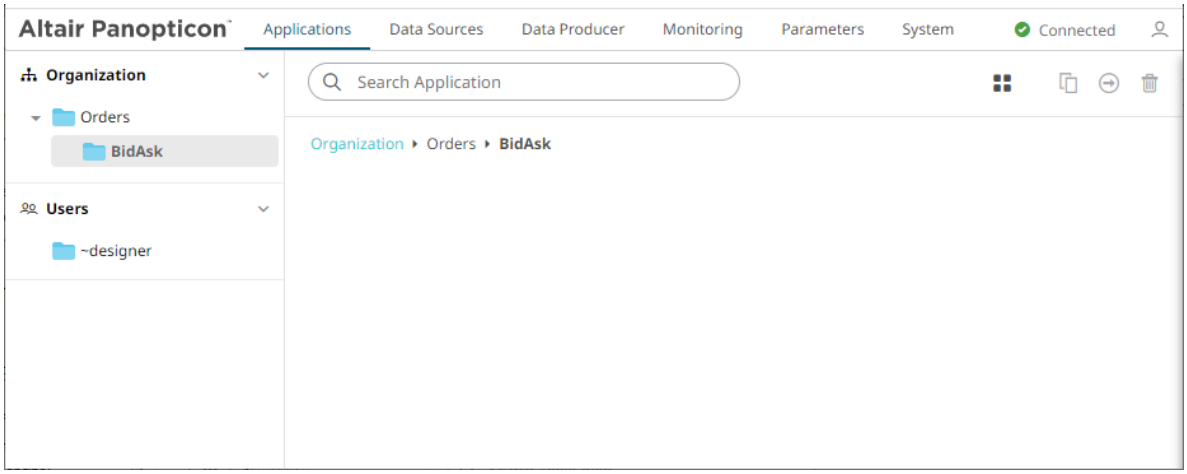
2. Right-click on the folder or on the *Applications* pane or *Data Sources* pane and select **New Folder**.



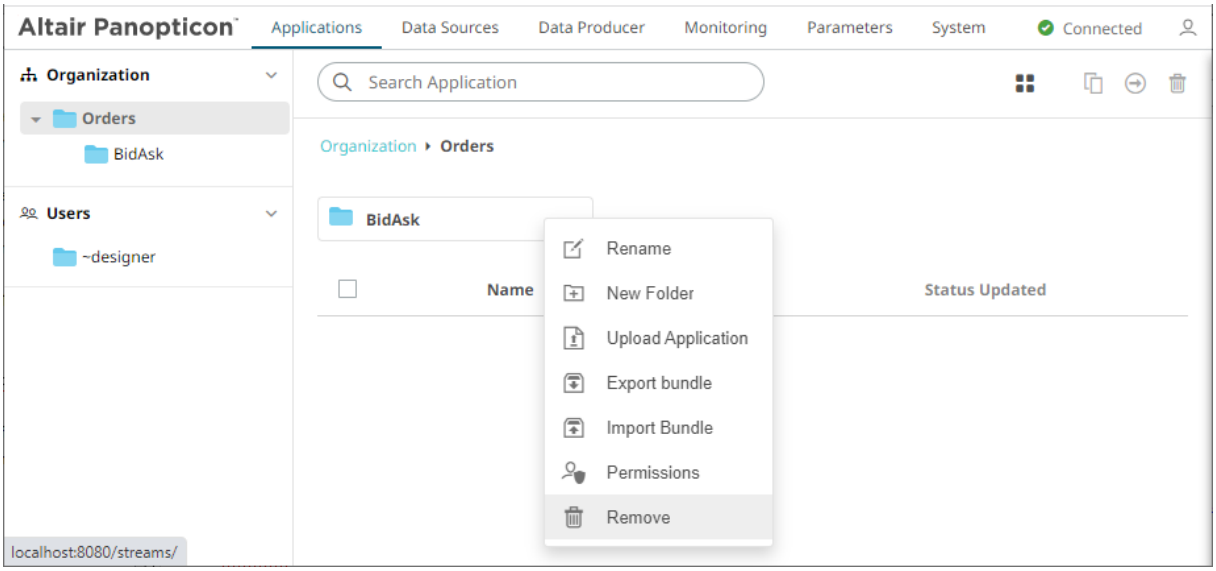


Refer to Creating Folders for the steps in creating the subfolders. Also, Adding Groups and Users with Allowed Authorization and Adding Groups and Users with Denied Access for more information on adding users and groups with allowed or denied authorization.

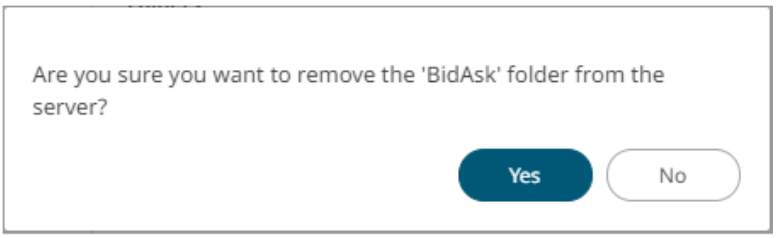
The subfolder is added.



3. You can also opt to delete a subfolder by right-clicking on the folder and selecting **Remove** on the context menu as long as it does not contain applications or data sources.



A confirmation message displays.



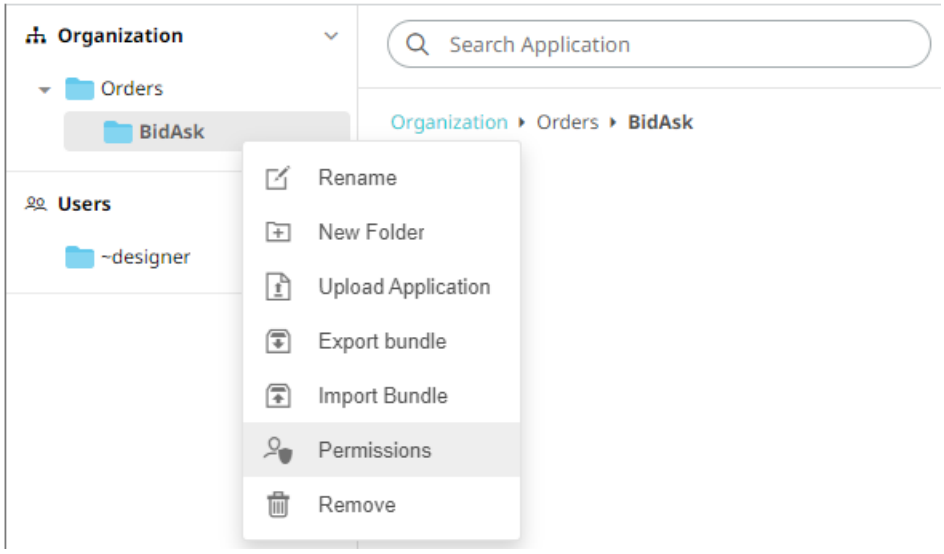
Click

.

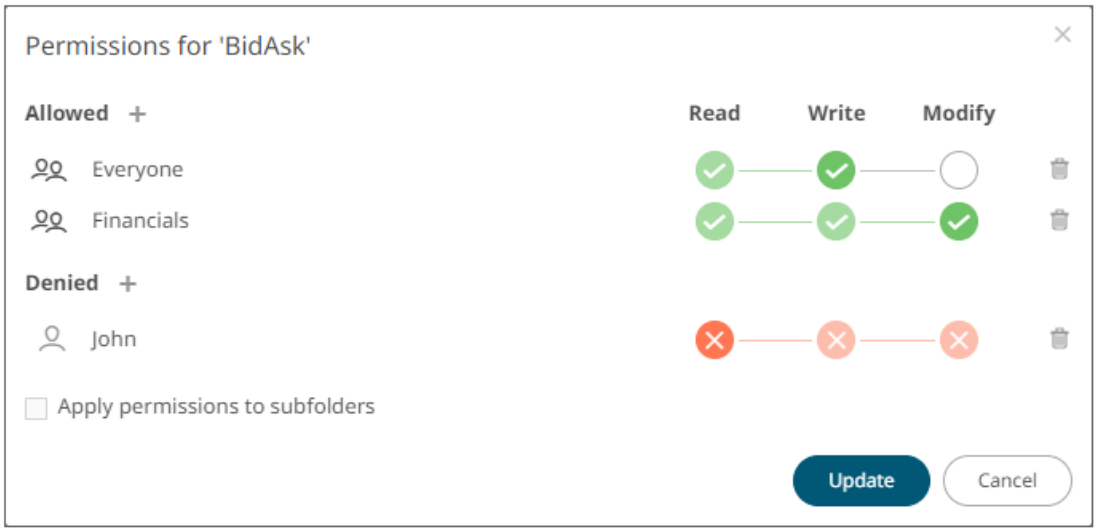
Updating Folder or Subfolder Properties

Steps:

1. To update folder properties, click a folder or a subfolder.
2. Right-click on the folder or subfolder and select **Permissions**.



The corresponding *Permissions* dialog displays.



4. Make the necessary changes such as new folder name, add or delete users and groups.
5. You can either:
   - check the **Apply Permissions to Subfolders** box

     This means the permissions that will be used on all of the subfolders will be fetched from the root folder.
   - leave the **Apply Permissions to Subfolders** box unchecked and modify the permission properties of the subfolders

NOTE	**The Apply Permissions to Subfolders check box is not enabled when defining the permissions for a subfolder.**

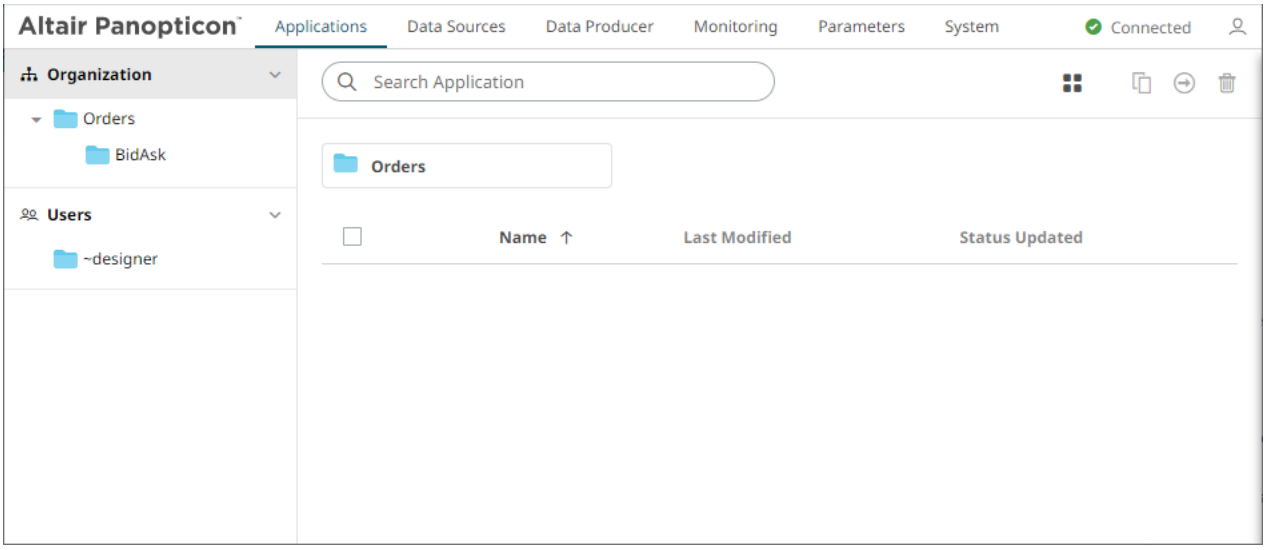
6. Click

to save the changes.

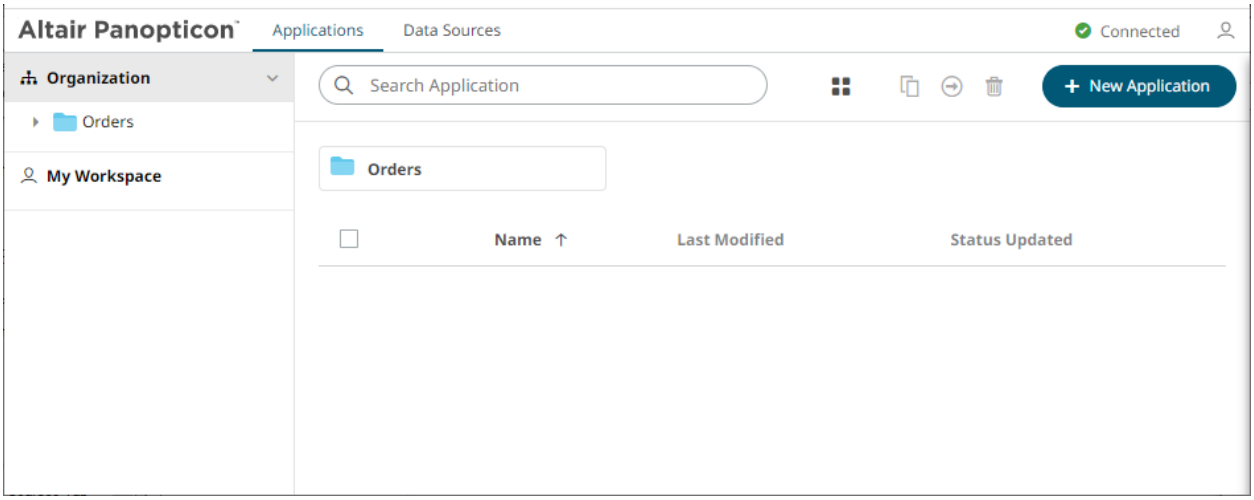
[8] MANAGING APPLICATIONS

On the **Applications** tab, users with Administrator or Designer role can:

- import and export application bundles
- upload applications
- rename applications
- view topic or data source usage
- move or copy applications to folders or subfolders to which the user has permission
- download applications
- remove applications
- publish/republish applications to folders to which the user has permission



To create a new application, a user must have a Designer role.

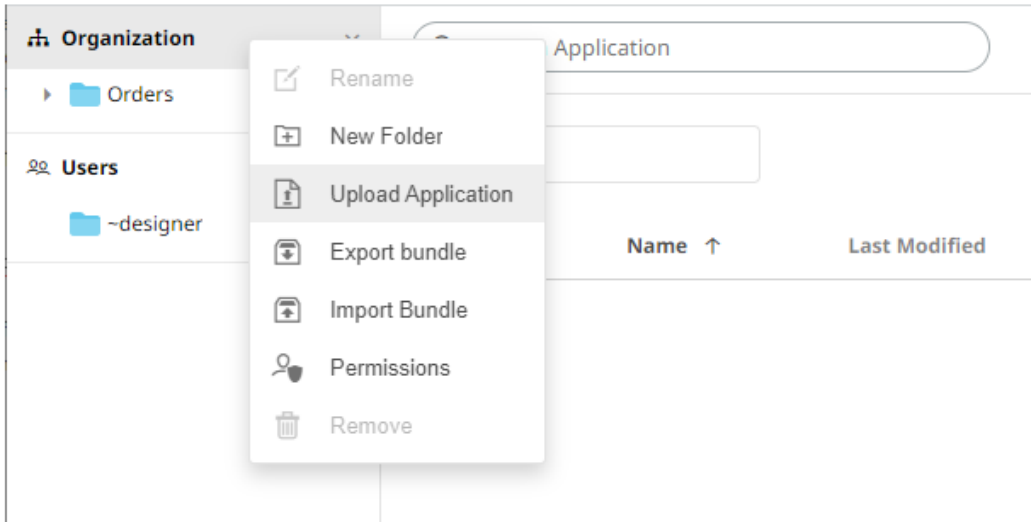


UPLOADING APPLICATIONS

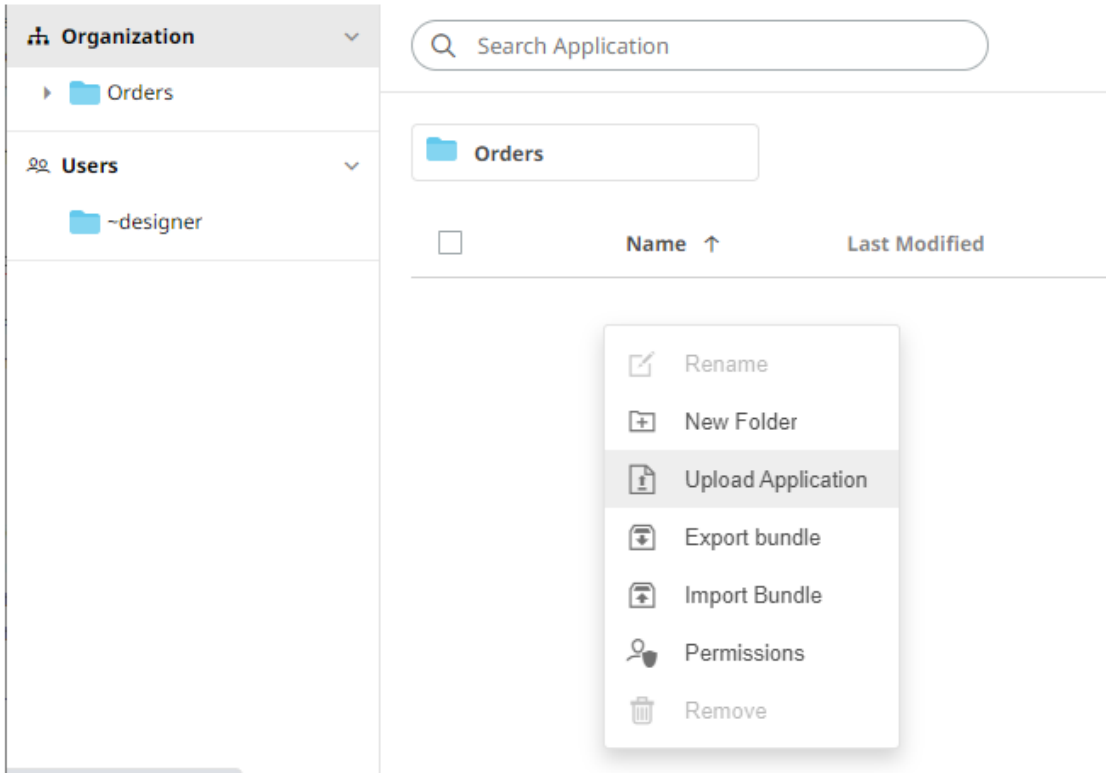
Users with Administrator or Designer role can upload applications to folder or subfolders that they have permission.

Steps:

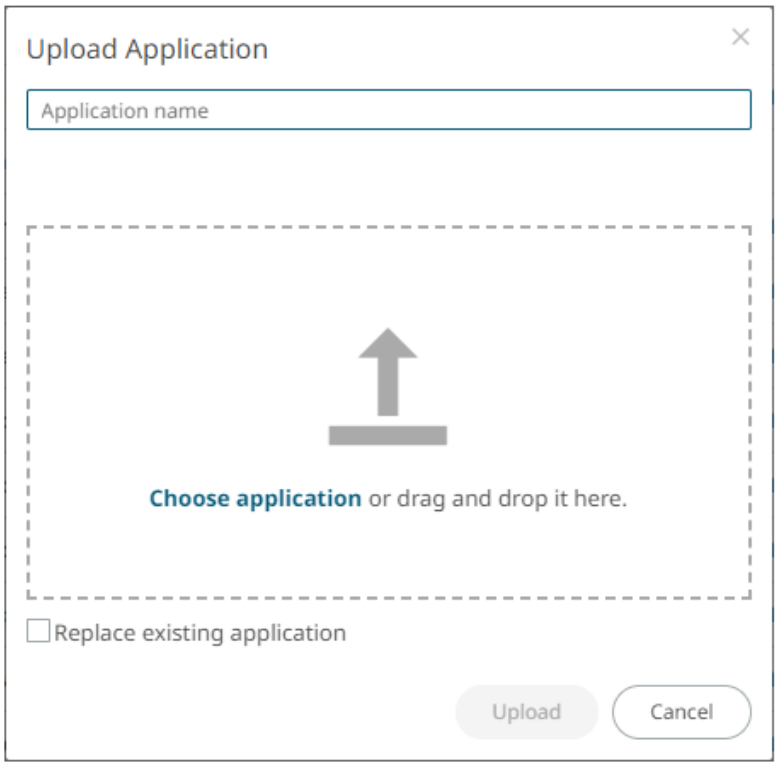
1. To upload applications, you can either right-click a folder or subfolder then select **Upload Application**:
   - on the expanded *Folder* hierarchy list



   - on the Root folders list

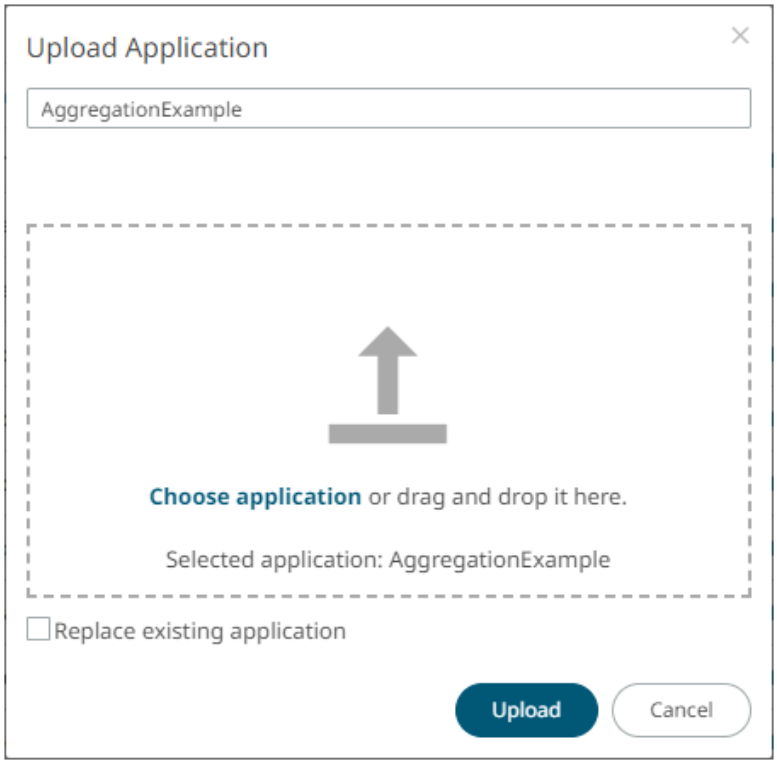


The *Upload Application* dialog displays.



2. To upload an application, you can either:
   - drag it from your desktop and drop in the dialog, or
   - click **Choose Application** and select one in the *Open* dialog that displays.

     The name of the application is displayed on the uploaded application area and in the *Name* box.



3. You can opt to rename the application.

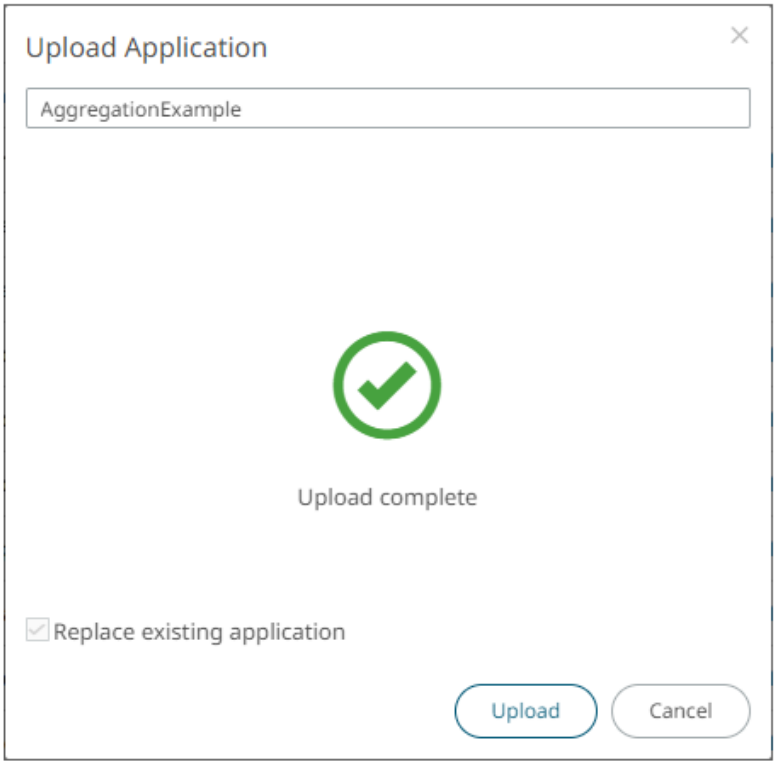
NOTE	**The application name must start with a letter (a to Z) or underscore. Also, it can only contain letters (a to Z), numbers (0 to 9), and underscores.**

4. To replace an existing application, check the **Replace existing application** box.

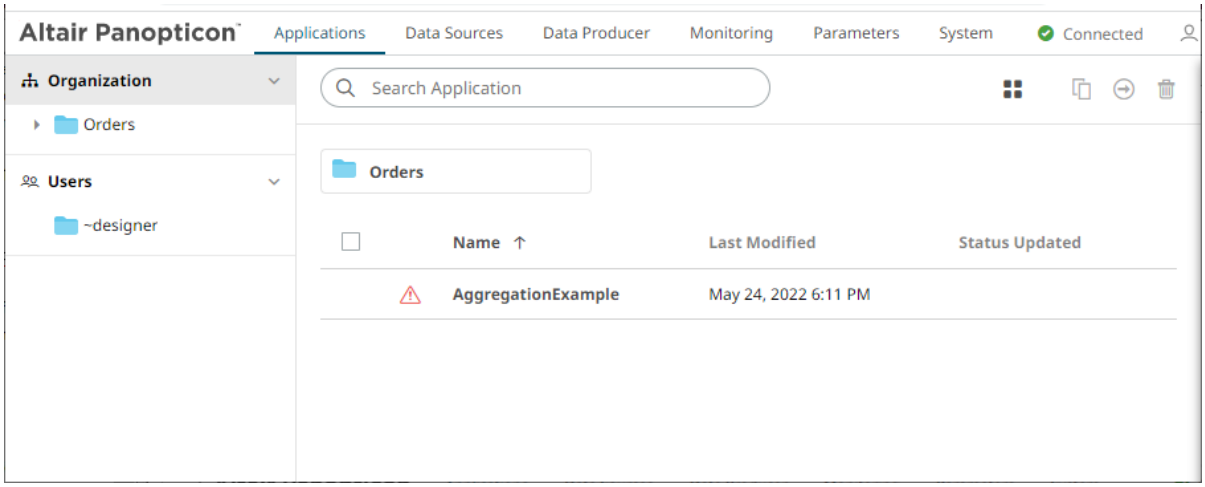
5. Click

.

   You will be notified when the application has been uploaded.



The application is added and displayed on the **Applications** tab.

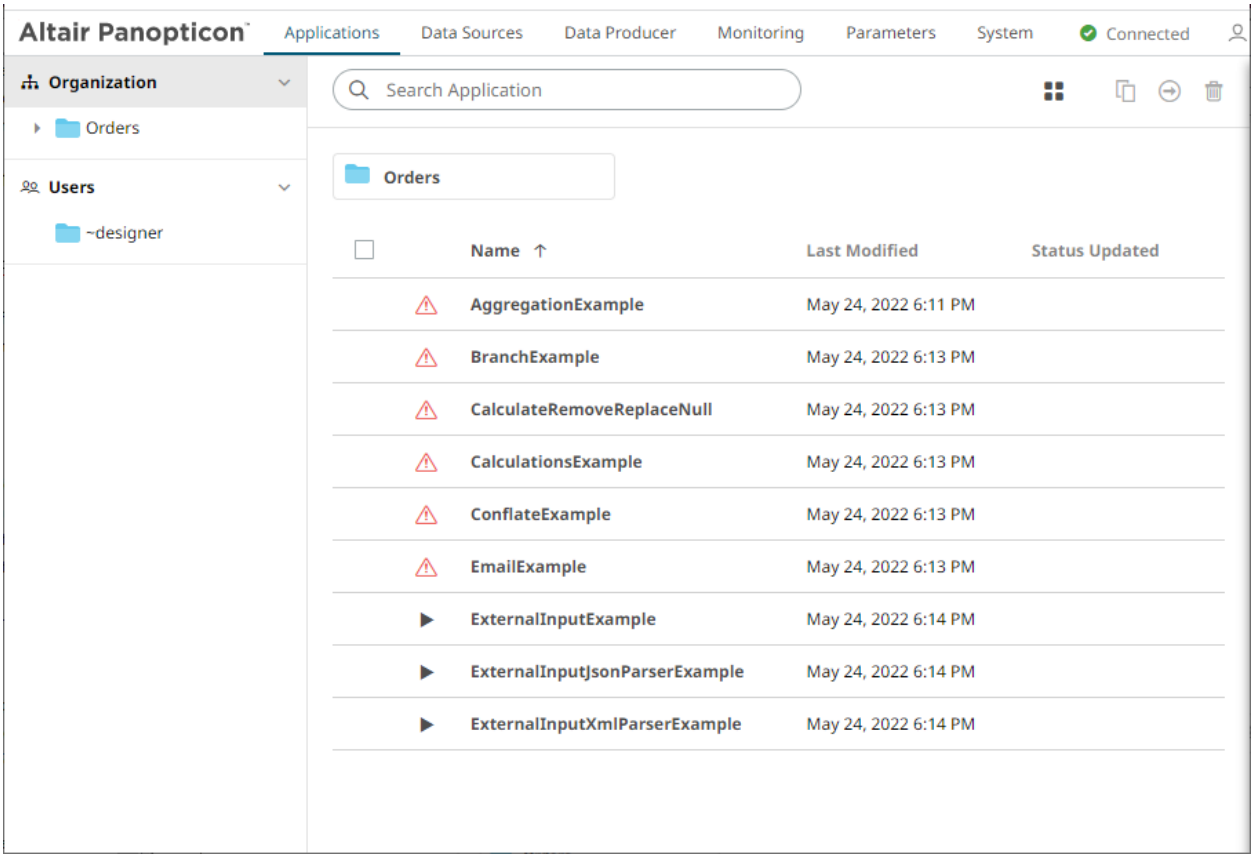


NOTE **A icon displays before the application name. This means the required data source is not available. Refer to Uploading Data Sources for more information.**

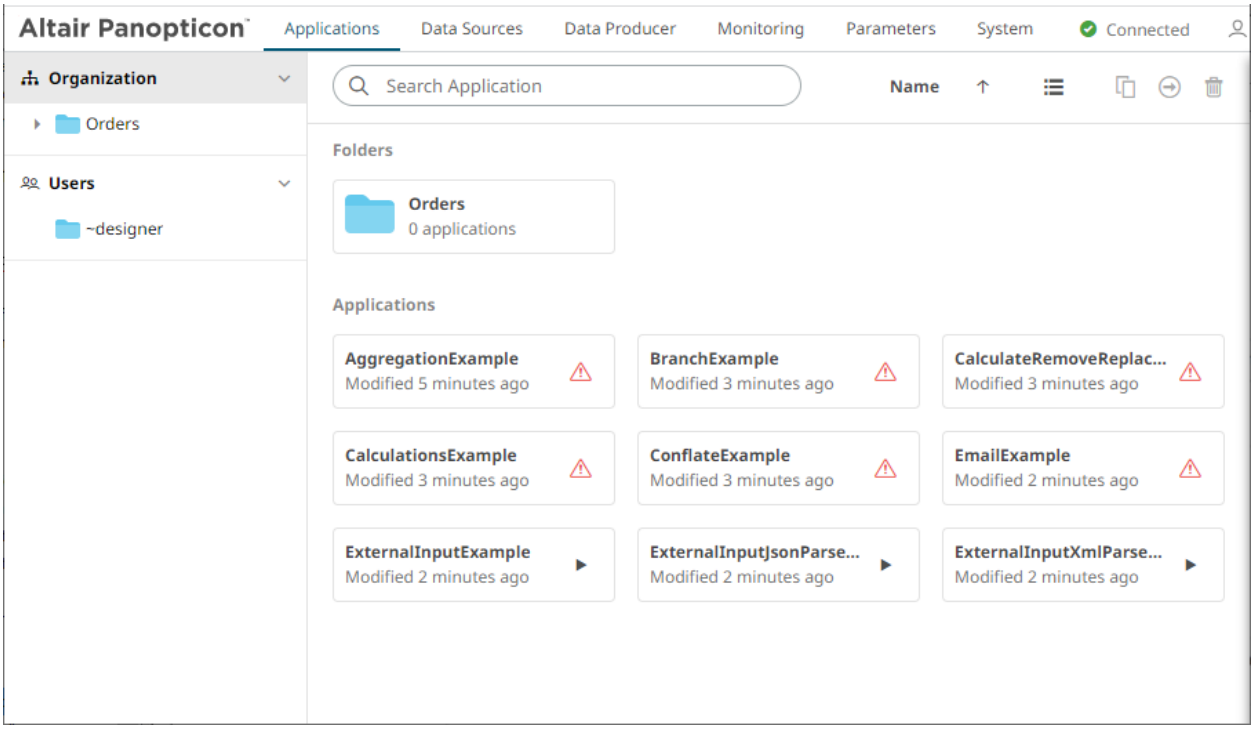
When the data source is available, the icon changes to .

Folders and Applications Display View

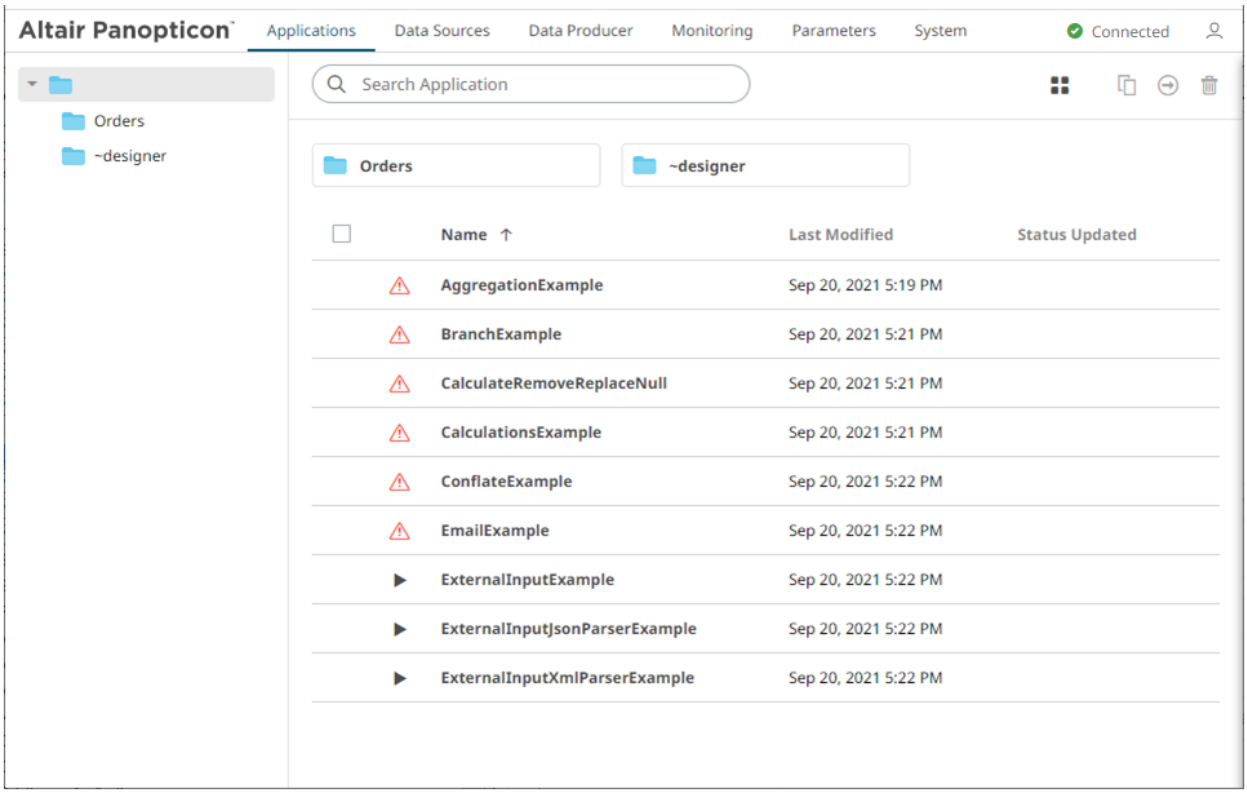
Folders and applications can be displayed either on a *List* or *Grid View*. By default, the applications are displayed in a *List View*.



Click **Grid View** . The folders and applications are displayed as thumbnails.



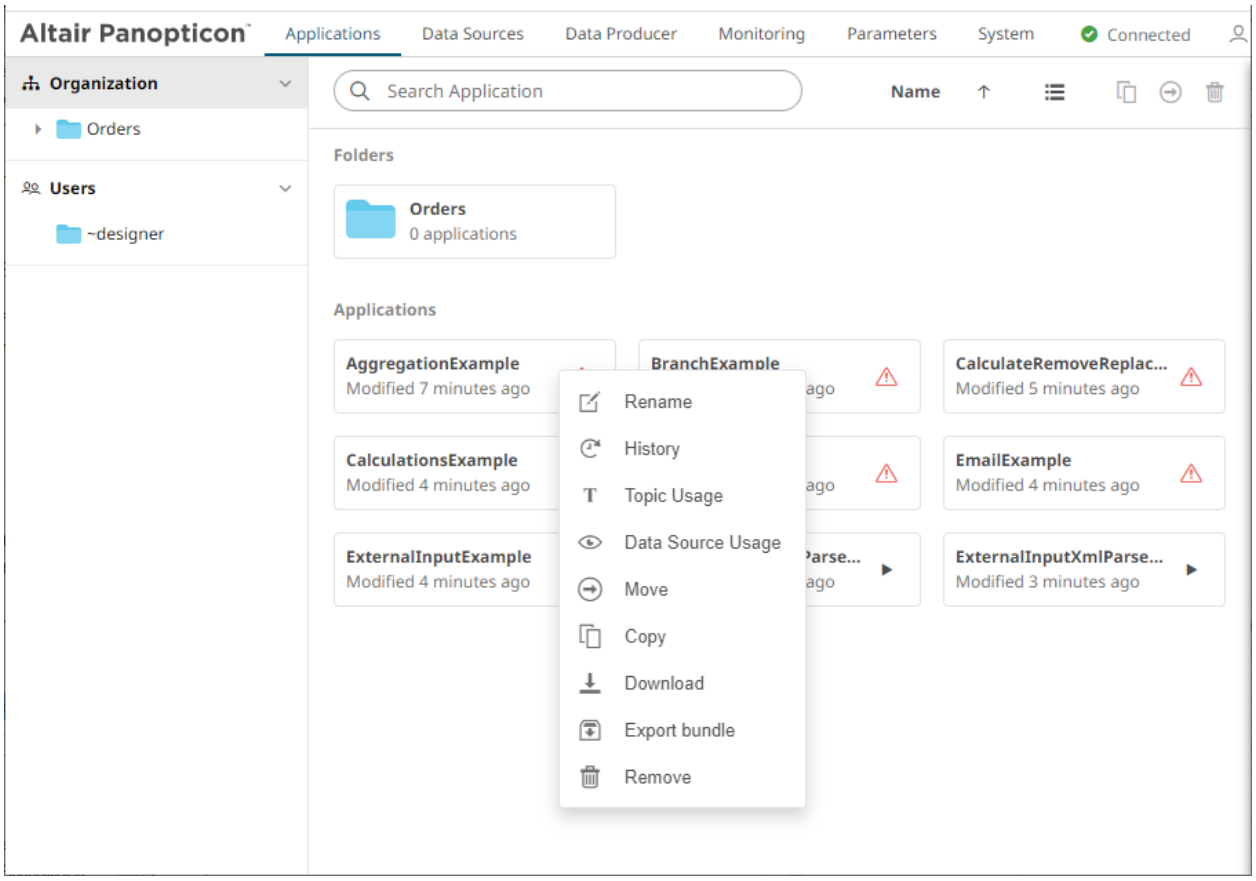
Click **List View** to return to the standard listing.



On either display view style:

- clicking on an application title or thumbnail displays the application

- right-clicking on an application displays the context menu

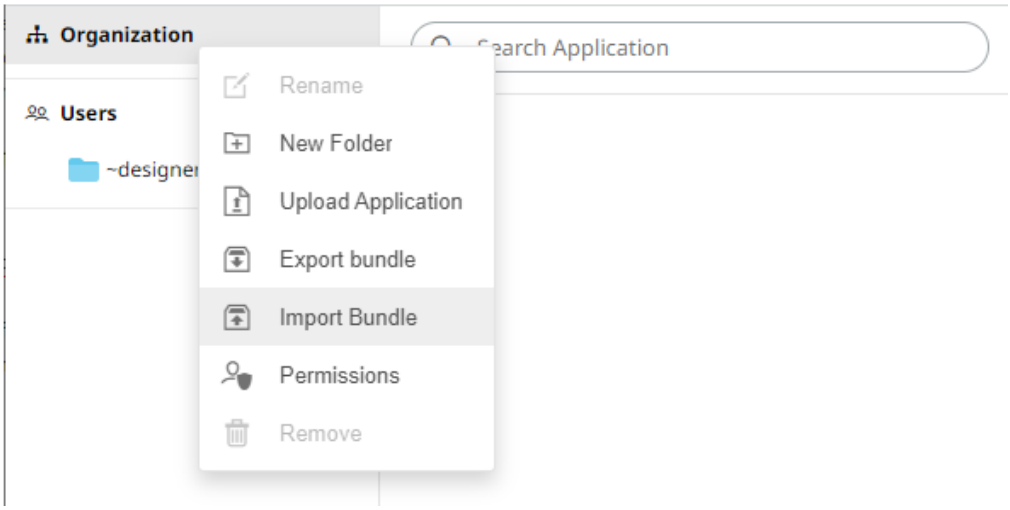


IMPORTING AN APPLICATION BUNDLE

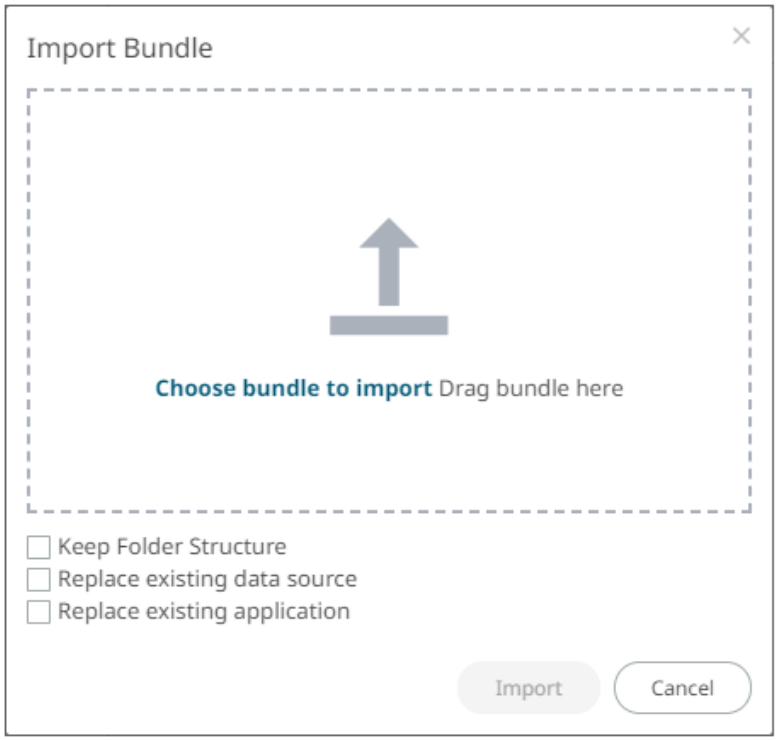
Follow the instructions below to import an application bundle to Panopticon Streams Server.

Steps:

1. On the **Applications** tab, right-click on a folder and select **Import Bundle** on the context menu.



The *Import Bundle* dialog displays.



2. To import the bundle, you can either:
   - drag it from your desktop and drop on the dialog, or
   - click **Choose Bundle to Import** and select one on the *Open* dialog that displays.

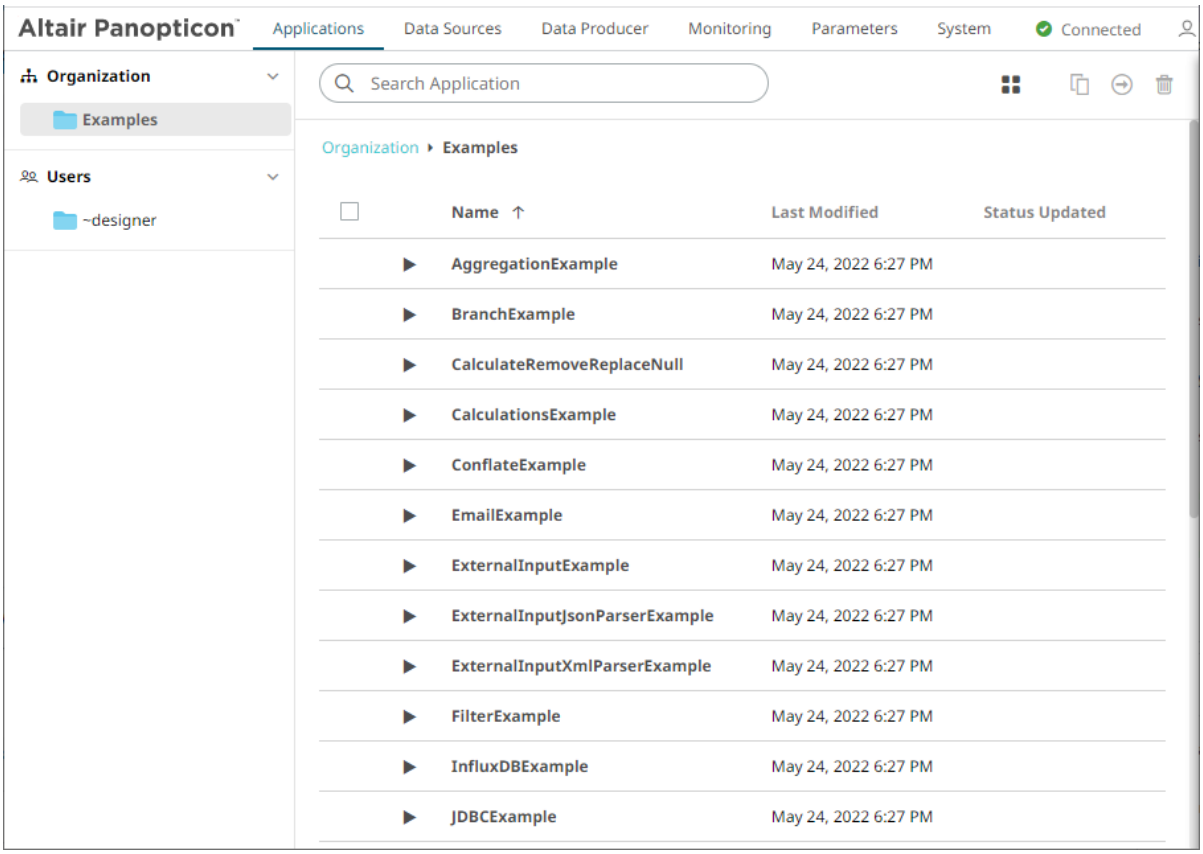
     The name of the selected bundle is displayed on the dialog box.

3. Check the **Keep Folder Structure** box.

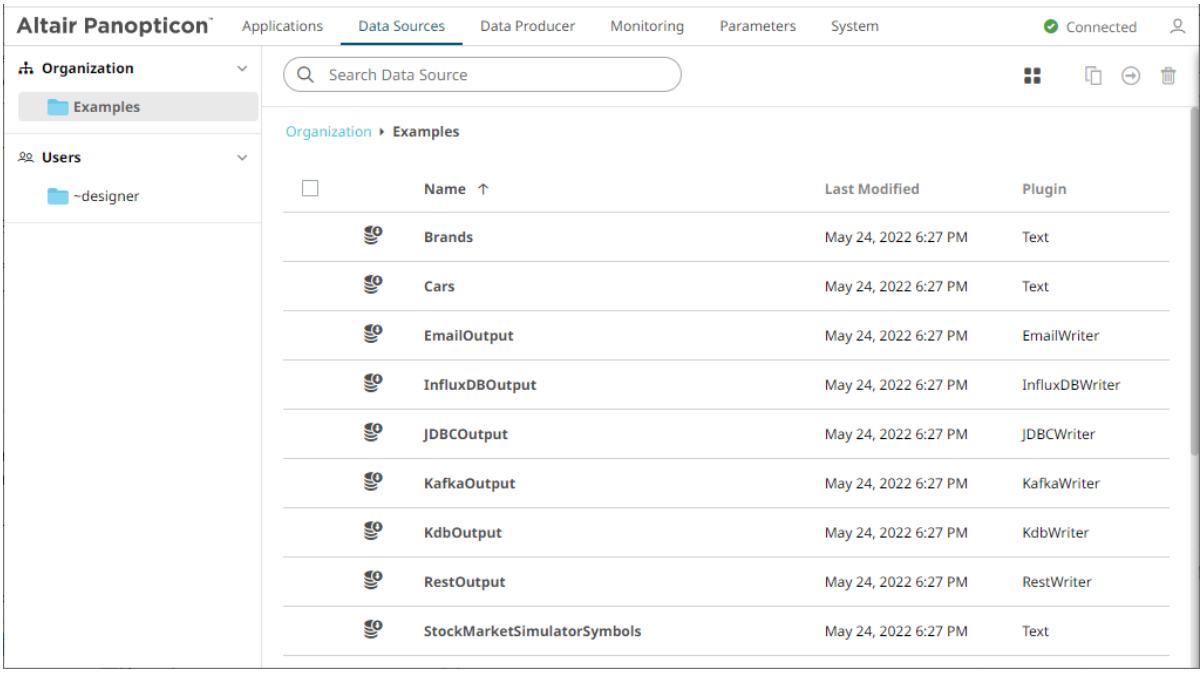
   This means the exported folder structure is maintained when uploading the bundle. If the folders do not exist on the server, they will be created.

4. To replace an existing data source, check the **Replace existing data source** box.
5. To replace an existing application, check the **Replace existing application** box.
6. Click Import .

   The applications and data sources that you can view and explore are imported.



Clicking the **Data Sources** tab displays the associated data sources of the sample applications.

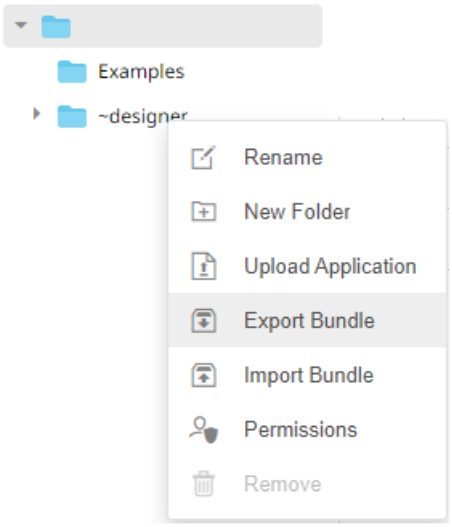


EXPORTING AN APPLICATION OR FOLDER BUNDLE

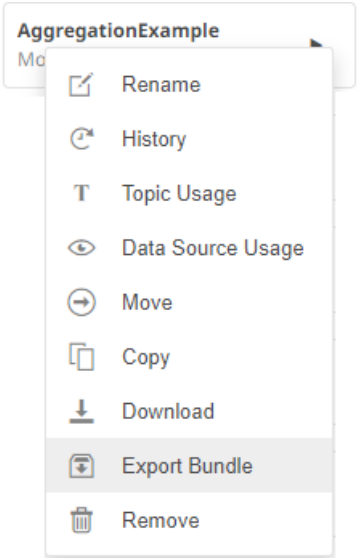
Users with an Administrator or Designer role have the ability to export application or folder bundle and the associated data files.

Steps:

1. Right-click on a application or folder and select **Export Bundle** on the context menu.



Application Folder or Subfolder Context Menu



Application Context Menu

A notification message displays.

The **Include datasources and Include data files** boxes are checked by default. This means the associated application data sources and data files will be included in the download.

2. Click

. A copy of the application or folder bundle is downloaded.

SORTING THE LIST OF APPLICATIONS

Sorting applications can be done by *Name*, *Last Modified*, or *Status Updated*.

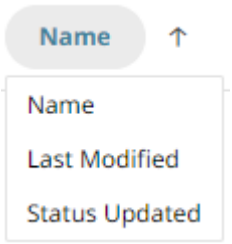
Sort By	Default Sort Order
Name	Ascending
Last Modified	Descending
Status Updated	Descending

Steps:

On the *Folders and Applications Summary* layout, either:

- click the **Sort By** option on the *Toolbar* of the *Grid View*

  By default, the sorting is by **Name**.

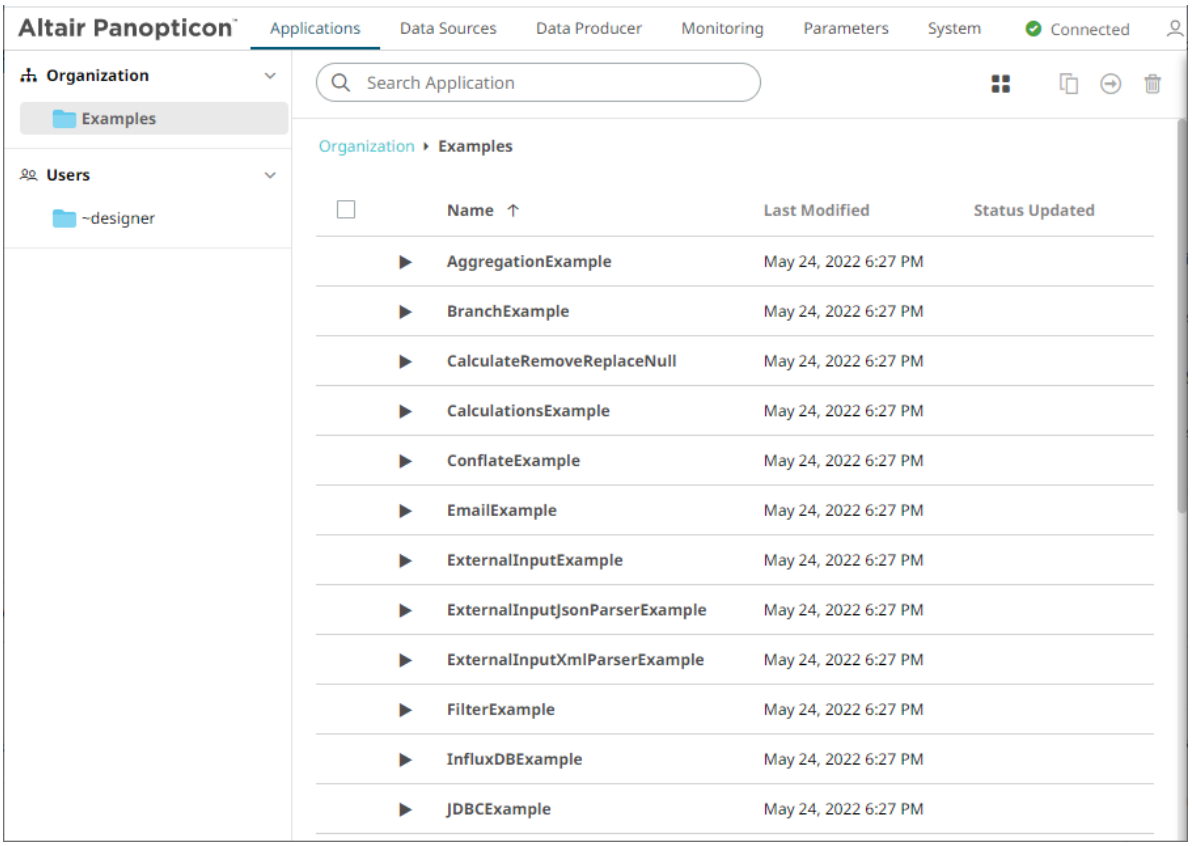
  

  - Name
  - Last Modified
  - Status Updated

  Then click the *Sort Order*:

  - Ascending
  - Descending

- click on the **Name**, **Last Modified**, or **Status Updated** column header of the *List View*

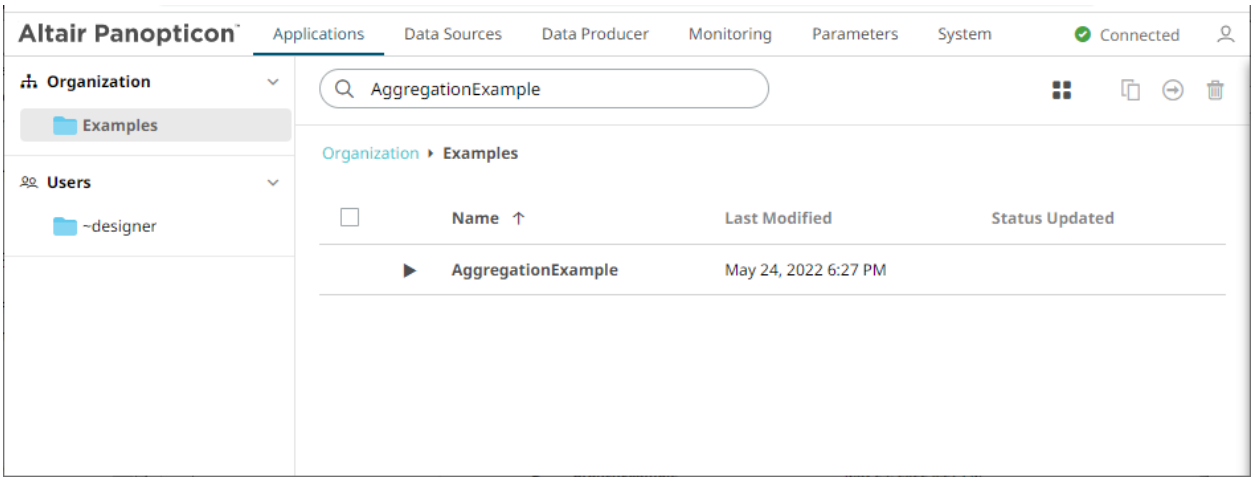


Then click the *Sort Order*:

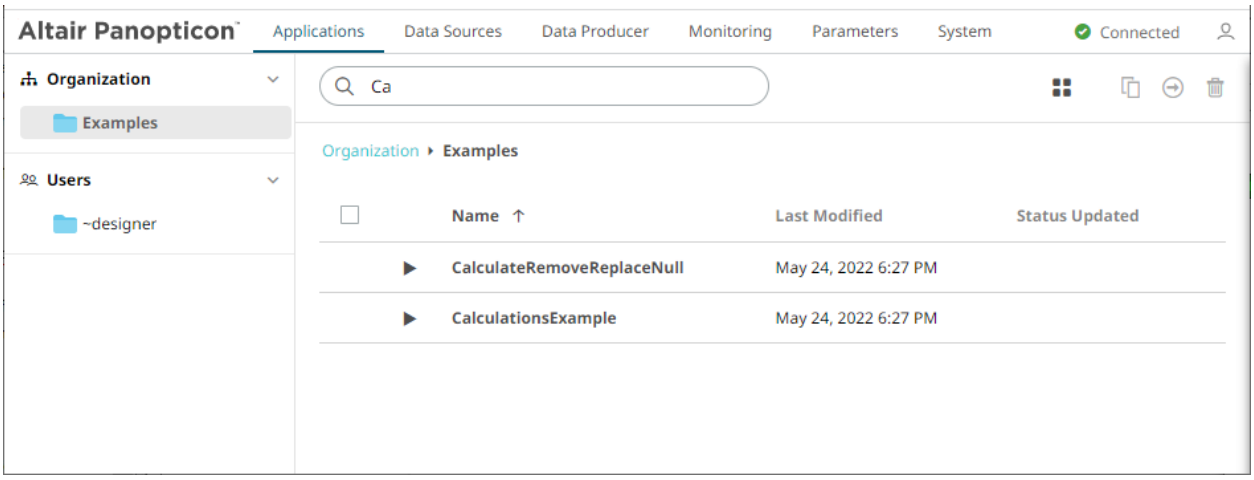
- ↑ Ascending
- ↓ Descending

SEARCHING FOR APPLICATIONS

To search for a particular application, enter it in the *Search Application* box.



You can also enter one of more characters into the *Search Application* box then click **Enter**. The list of applications that matched the entries will be displayed.

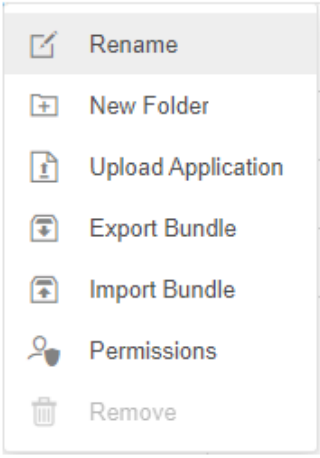


RENAMING APPLICATIONS OR FOLDERS

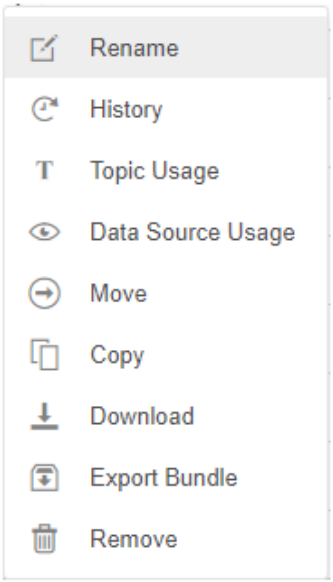
A user with an Administrator or Designer role can rename applications and folders.

Steps:

1. Right-click on an application or folder then select **Rename** on the context menu.

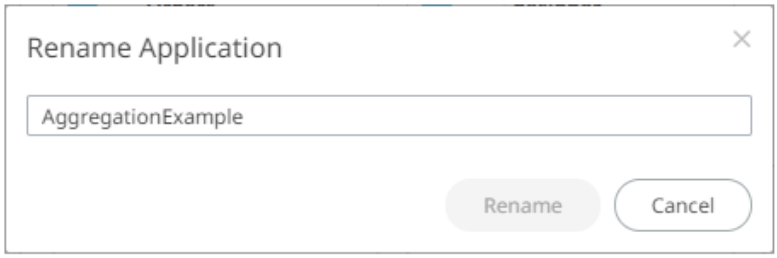


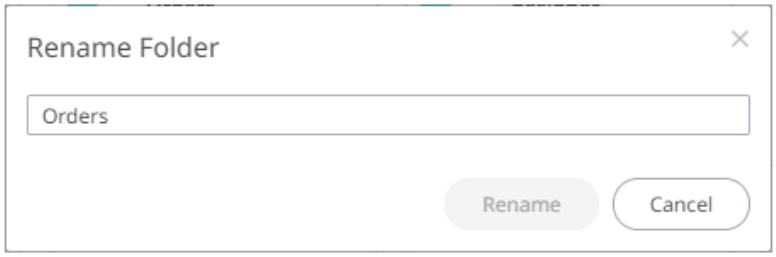
Folder or Subfolder Context Menu



Application Context Menu

The *Rename Application* or *Rename Folder* dialog displays.





2. Enter a new name then click

.

VIEWING APPLICATION HISTORY AND REPUBLISHING

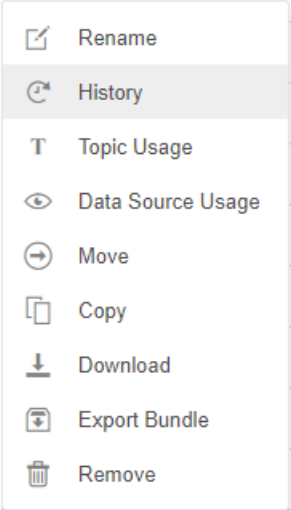
A user with either an Administrator or Designer role can also perform the following:

- View the change history of applications
- Republish an archived application to the recent version of Panopticon Streams Server

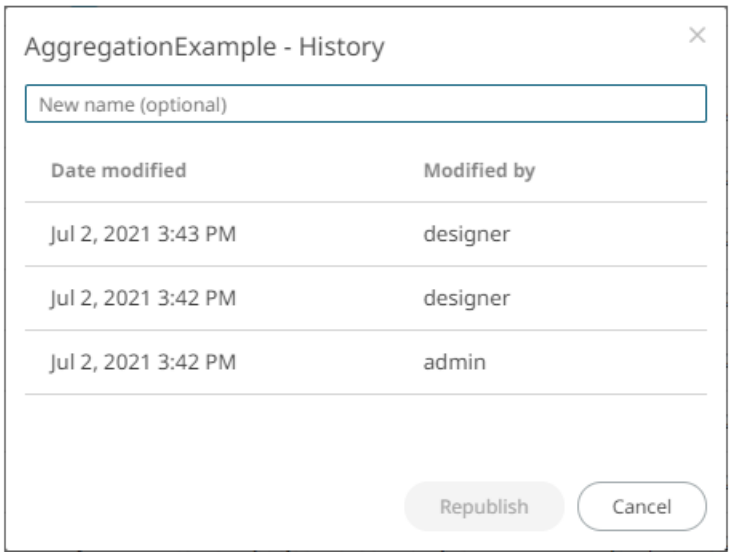
- Rename an archived application

Steps:

1. On the **Applications** tab, right-click on an application and select **History** on the context menu.



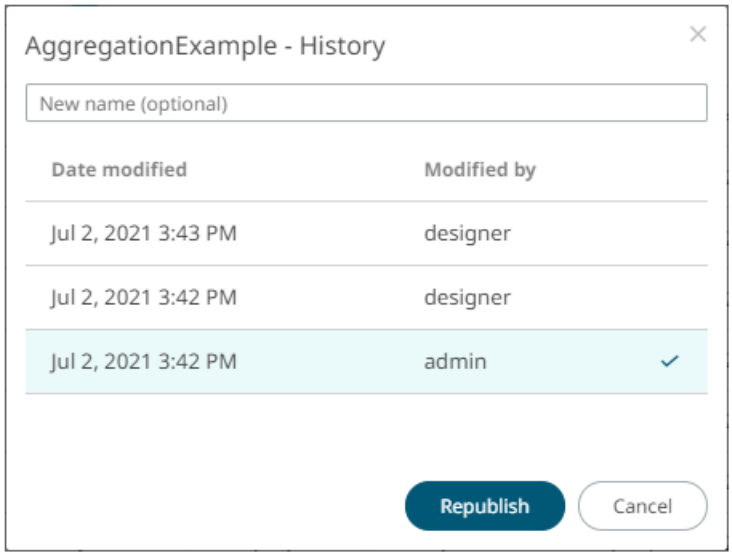
The *<Application Name> - History* dialog displays:



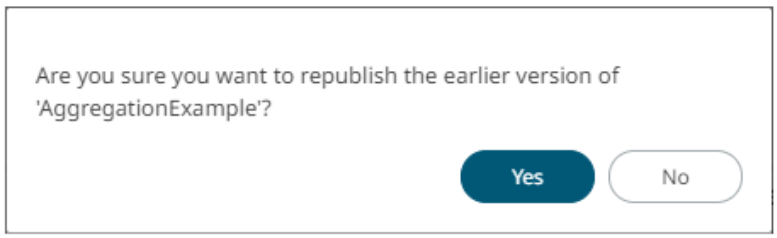
Sort the archival list either through the *Date Archived* or *Archived By* by clicking on the ↓ or ↑ button.

Also, move to the other pages of the list by clicking on a page or clicking the « or » button.

2. You may opt to rename an archived application by entering a new one in the *New Name* box.
3. Click on an archived application in the list.



Then click Republish. A notification message displays.



4. Click Yes.

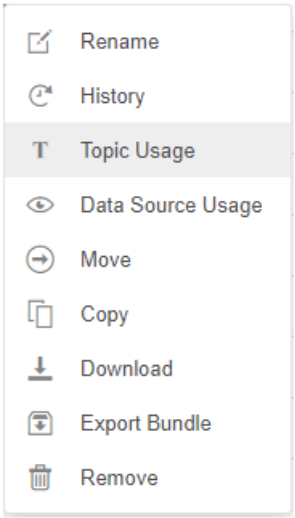
   A confirmation message displays.

VIEWING AND MANAGING APPLICATION TOPIC USAGES

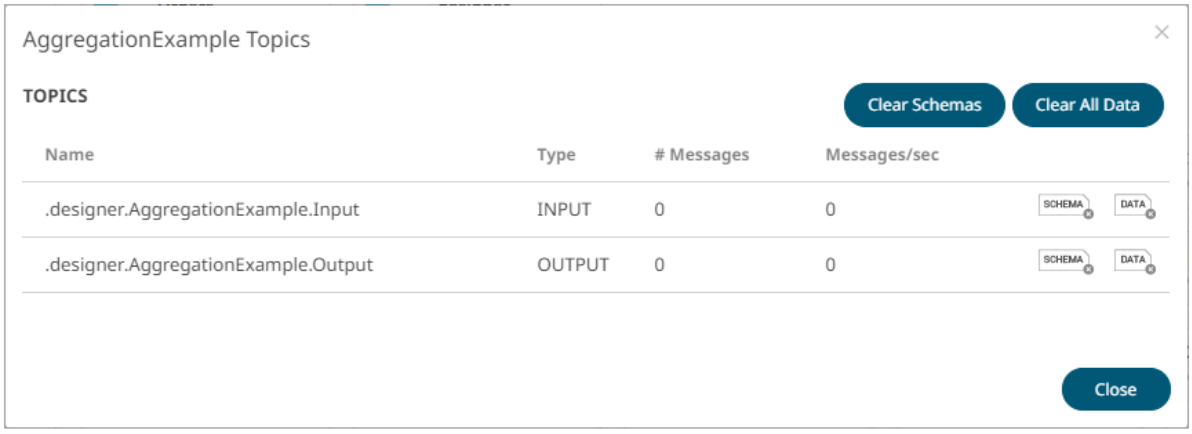
On the **Applications** tab, you can view the input and output topic usages of an application.

Steps:

1. On the **Applications** tab, right-click on an application and select **Topic Usage** on the context menu.



The *<Application Name> Topics* dialog displays.



If the application has been executed, the *#Messages* column will display the number of retrieved messages, while the *Messages/sec* column will display the number of retrieved messages per second.

If the application is not yet executed, both the *#Messages* and *Messages/sec* columns will display 0 values.

2. You can then opt to:
   - clear data
   - clear schemas
   - sort topics
3. Click **Close**.

Clearing the Data In an Application Topic

You can recycle an application by:

- stopping the application
- deleting data in the topics
- restarting the application

Follow the steps below to clear the data in an application topic.

Steps:

1. You can either:
   - Click DATA to delete the data in a topic, or
   - Click Clear All Data to delete the data in all of the topics in an application
2. Click Close .

Clearing the Schema in an Application Topic

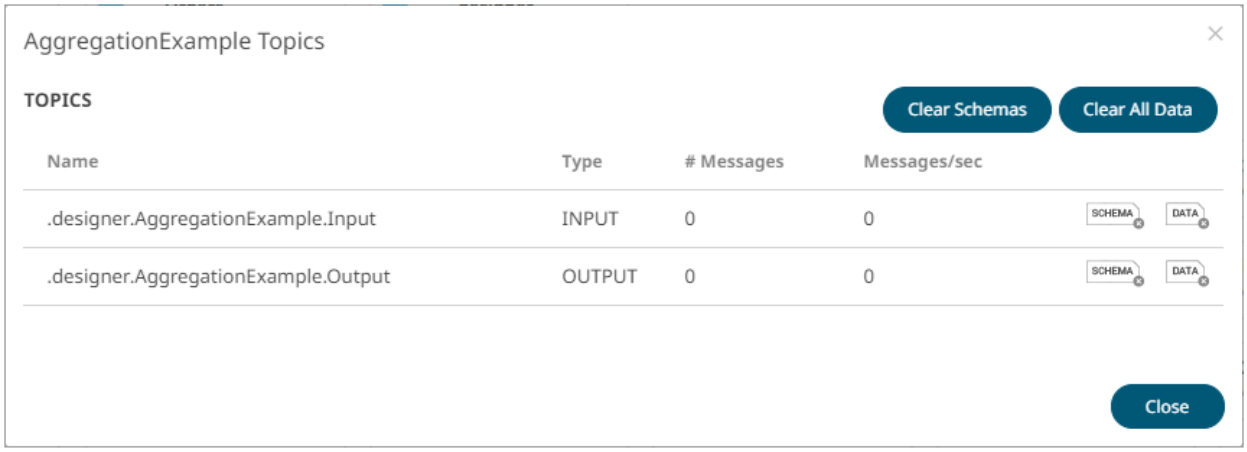
Schema registry can be cleared in any application topic.

Steps:

1. You can either:
   - click SCHEMA to delete the schema in a topic, or
   - click Clear Schemas to delete the schema in all of the topics in an application
2. Click Close .

Sorting Application Topics

By default, the list of topics is sorted by *Name* in an ascending order.



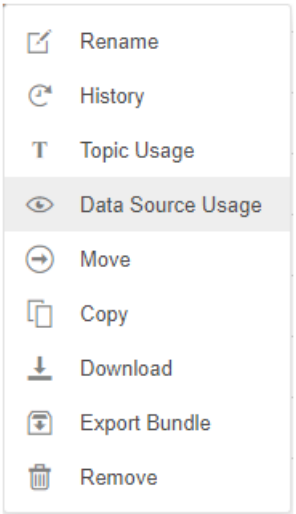
Modify the sorting of the list by clicking the ↓ or ↑ button of the *Name*, *Type*, *#Messages* or *Messages/sec* columns. The icon beside the column that was used for the sorting will indicate if it was in an ascending or descending order.

VIEWING THE APPLICATION DATA SOURCES USAGE

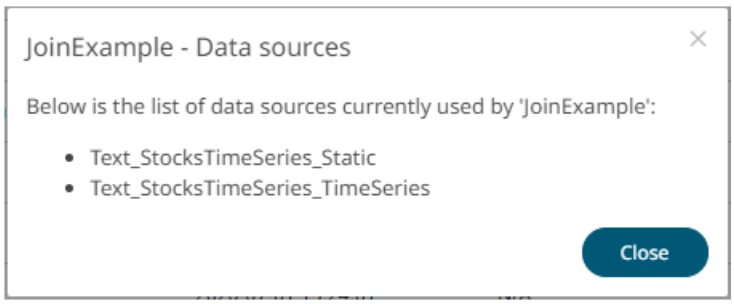
On the **Applications** tab, you can view the data sources that are currently used by an application.

Steps:

1. On the **Applications** tab, right-click on an application and select **Datasource Usage** on the context menu.



The list of data sources that is currently used by the application displays.



2. Click Close .

MOVING APPLICATIONS

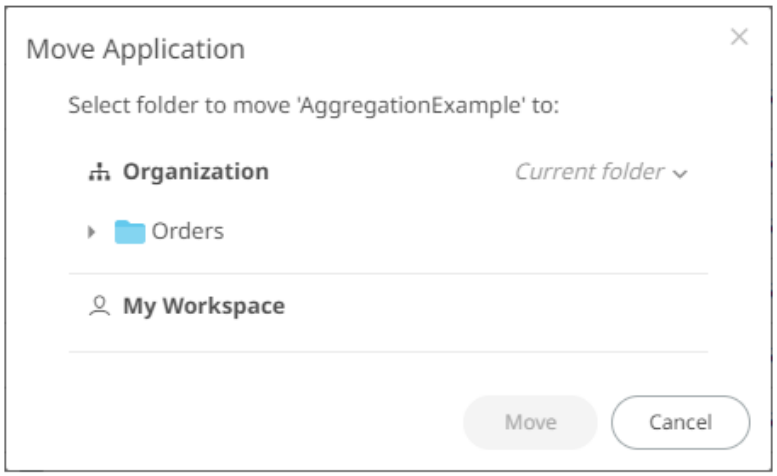
Users with Administrator or Designer role are allowed to move an application to another folder or subfolder to which they have permission.

Moving applications can be done either through the toolbar or context menu.

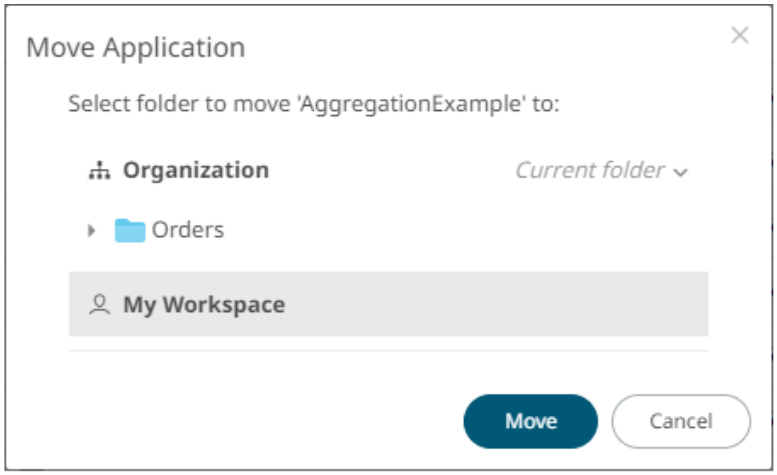
Steps:

1. Check the box of one or multiple applications either on the *Grid View* or *List View*.
2. Then select either:
   - the **Move** icon on the toolbar
   - **Move** on the content menu

   The *Move Application* dialog displays with the folder or subfolders to which the user is allowed to move the application.



3. Select the folder or subfolder.



4. Click Move.

NOTE **If applications with the same name are already in the selected folder, a notification message displays if they will be replaced.**

Click Yes to replace or No to move a copy of the same applications.

The application is moved and displayed on the selected folder.

COPYING APPLICATIONS

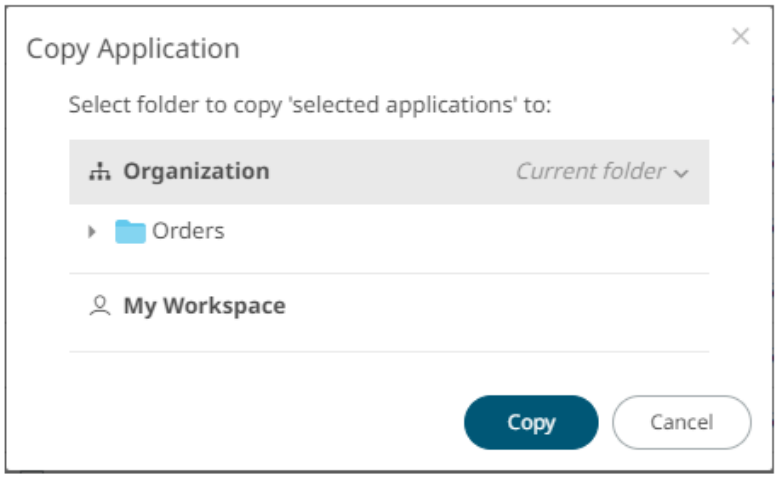
Users with Administrator or Designer role are allowed to copy an application to another folder or subfolder to which they have permission.

Copying applications can be done either through the toolbar or context menu.

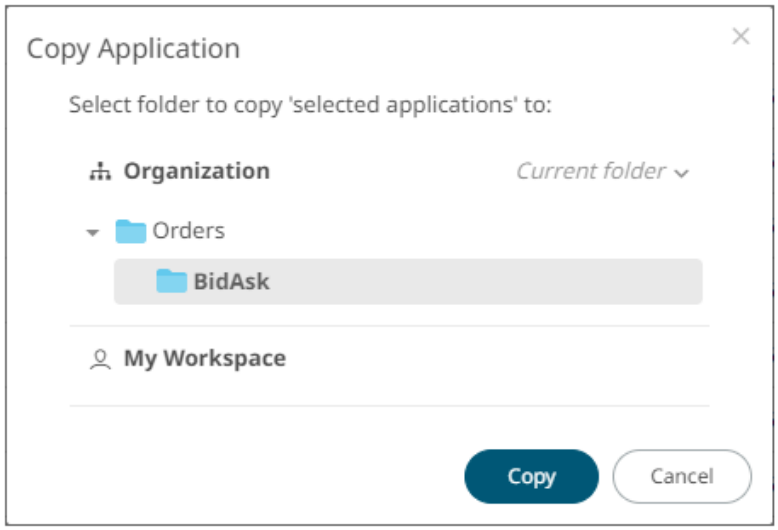
Steps:

1. Check the box of one or multiple applications either on the *Grid View* or *List View*.
2. Then select either:
   - the **Copy** icon on the toolbar
   - **Copy** on the content menu

   The *Copy Application* dialog displays with the folder or subfolders the user is allowed to copy the applications to.



3. Select the folder or subfolder.



4. Click

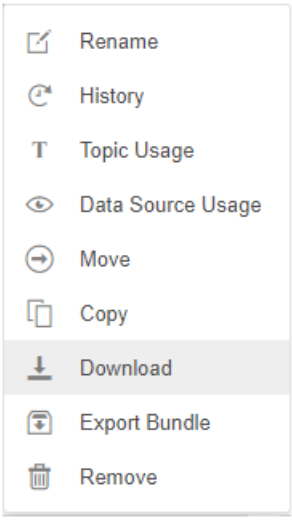
.

NOTE	If applications with the same name are already in the selected folder, a copy of the applications are added.

The applications are copied and displayed on the selected folder.

DOWNLOADING AN APPLICATION

Users with an Administrator or Designer role are allowed to download a copy of an application by right-clicking on an application and selecting **Download** on the context menu.



The application is downloaded.

DELETING AN APPLICATION

Users with Administrator or Designer role are allowed to delete an application which can be done either through the toolbar or context menu.

Deleting Applications Using the Toolbar

Steps:

1. Check the box of applications either on the *Grid View* or *List View*:

2. Click on the toolbar.

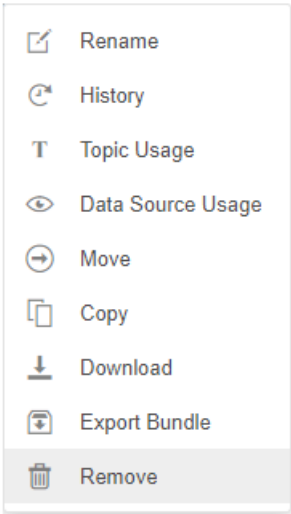
   A notification message displays.

3. Click Yes to remove.

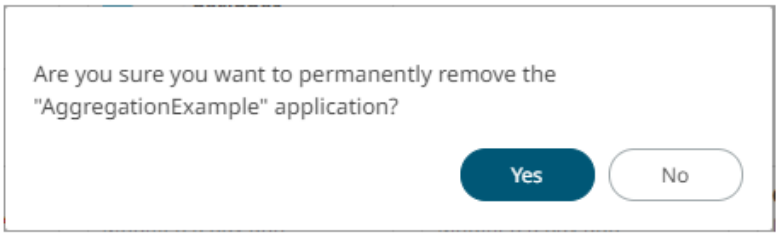
Deleting Applications Using the Context Menu

Steps:

1. Right-click on an application and select **Remove** on the context menu.



   A confirmation message displays.



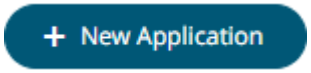
2. Click

to delete.

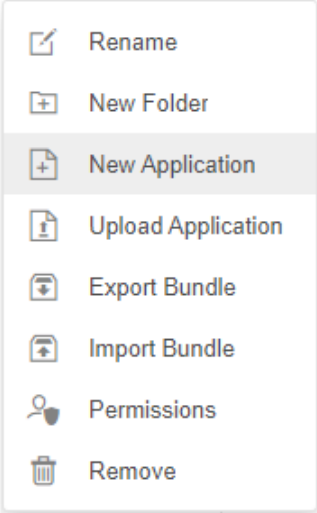
CREATING A NEW APPLICATION

A user with a Designer role can create new applications to folders or subfolders to which they have permission.

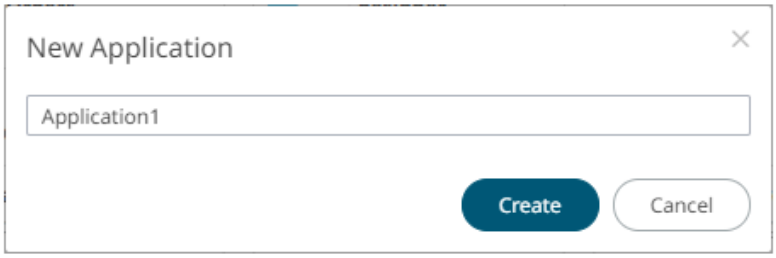
Steps:

1. On the **Applications** tab:
   - click 

on the toolbar
   - right-click on a folder or subfolder and select **New Application**.



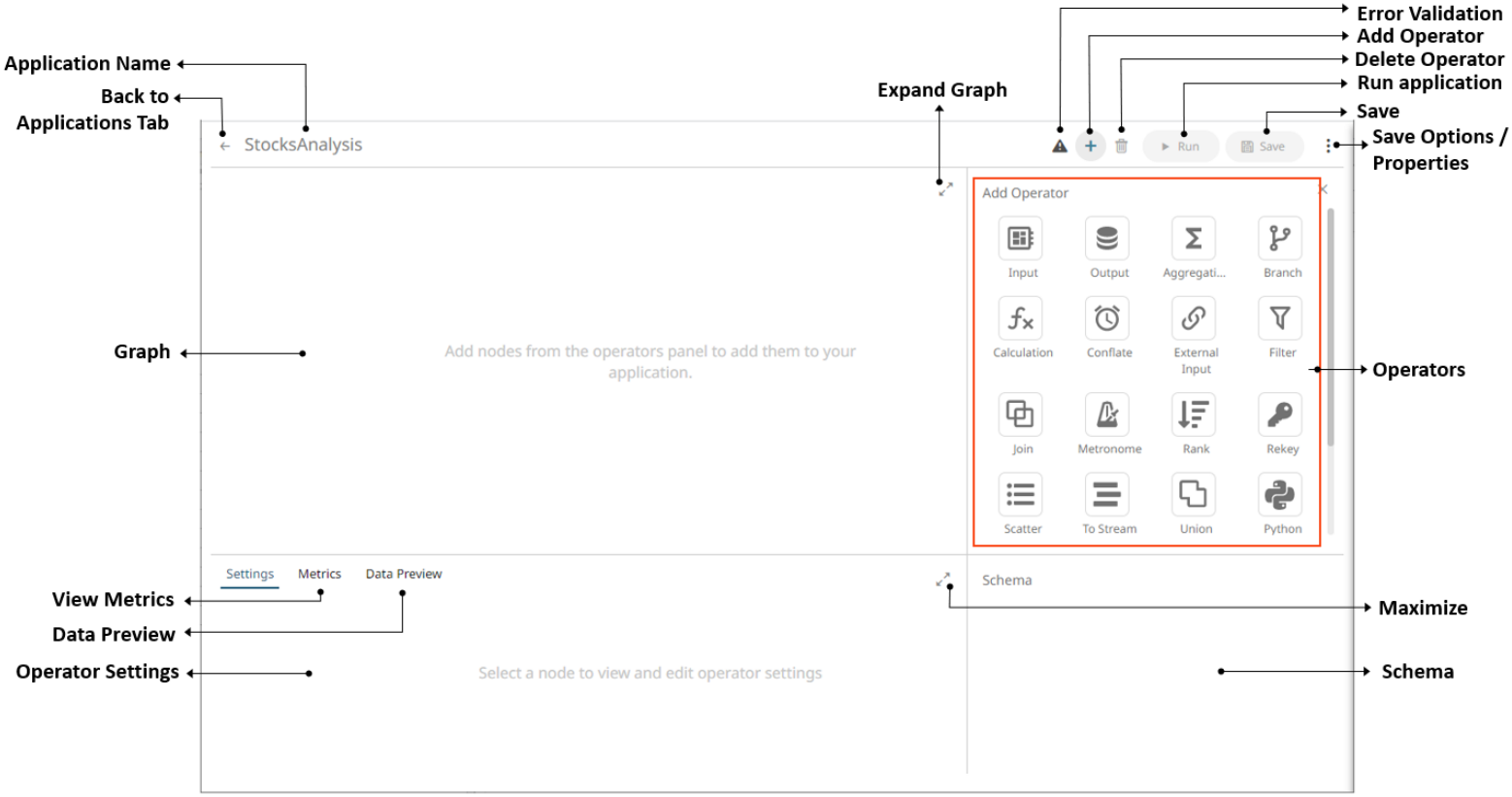
The *New Application* dialog displays.



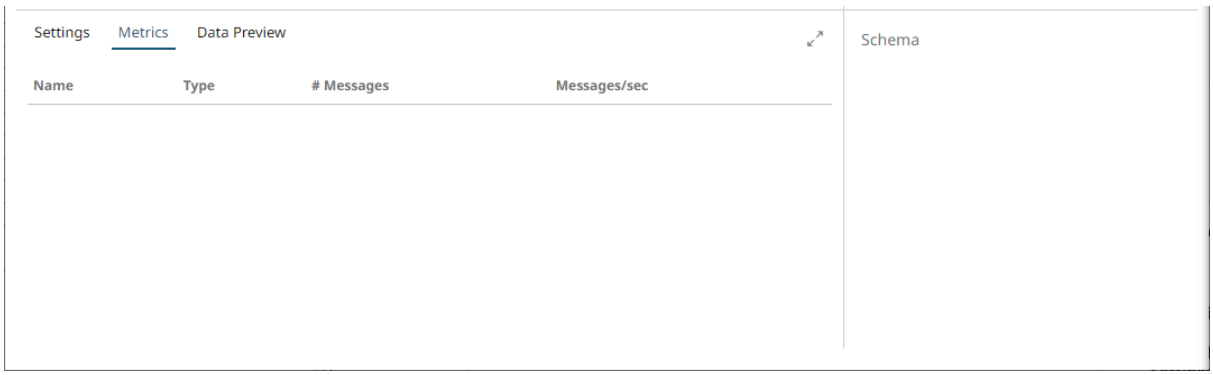
2. Enter the *Name* of the application and click

.

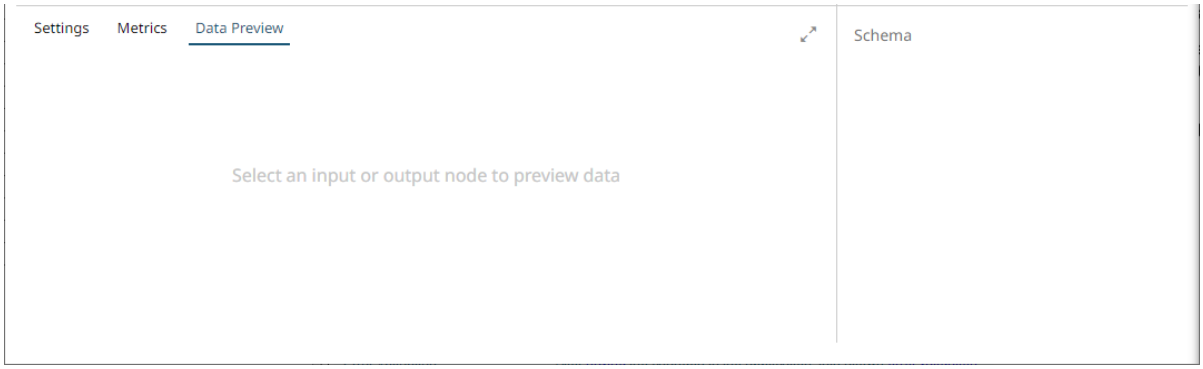
The *Application* page displays with the following sections. Initially, the *Operator Settings* pane is displayed.



Clicking the **Metrics** tab displays the *Metrics* pane:



Clicking the **Data Preview** tab displays the *Data Preview* pane. Note that you need to save and run the application to preview the data.



Section/Panel	Description
Application Name	Name of the application. Click the ← button to go back to the *Applications* listing page.
Expand Graph	Expands the *Graph* panel.
Error Validation	After saving the changes in the application, this allows error validation. If there are definition issues (red node) or if there is no traffic on the topic (yellow node), you can click ⚠ to help fix the errors. If there are no issues, ⚠ is no longer displayed in the *Application* page.
Add Operator	When enabled +, displays the *Add Operator* pane and allows operators to be added in the application. Click × to close the *Add Operator* pane.
Delete Operator	Deletes the selected operator.
Run Application	Runs or executes an application.
Save	Saves the changes made in the *Application* page.
Save Options and Properties	Allows saving of changes made in the application or saving another copy. The application properties are also displayed and allows adding new ones.
Graph	Display the nodes and edges of the application model. It allows single node selection.
Operators	Available operators that can be added in the application.
View Metrics	Displays the throughput of the selected nodes (total and current message rates) in the Metrics panel. When the application is running, the metrics data are refreshed.
Data Preview	Displays the retrieved query or table in the Data Preview panel.
Operator Settings	When an operator has been added or a node is selected in the application graph, the corresponding operator settings are displayed in the *Operator Settings* panel to allow editing.

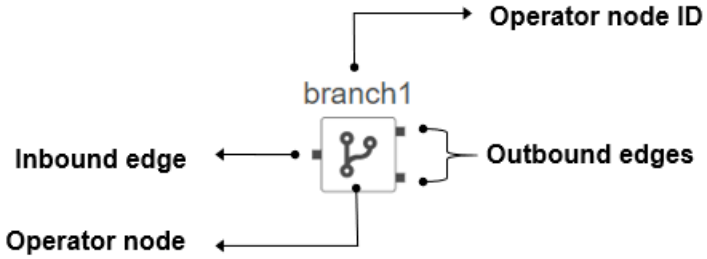
	When the application is running, the operator settings are displayed but are not editable.
Maximize	Expands the Operator Settings, Metrics, or Data Preview panel.
Schema	Displays the preview of the data.

Refer to the sections below to proceed in creating an application.

ADDING AN OPERATOR

Operators can be added in any order. The sequence or direction of the graph will be based on the inputs or outputs that will connect the nodes.

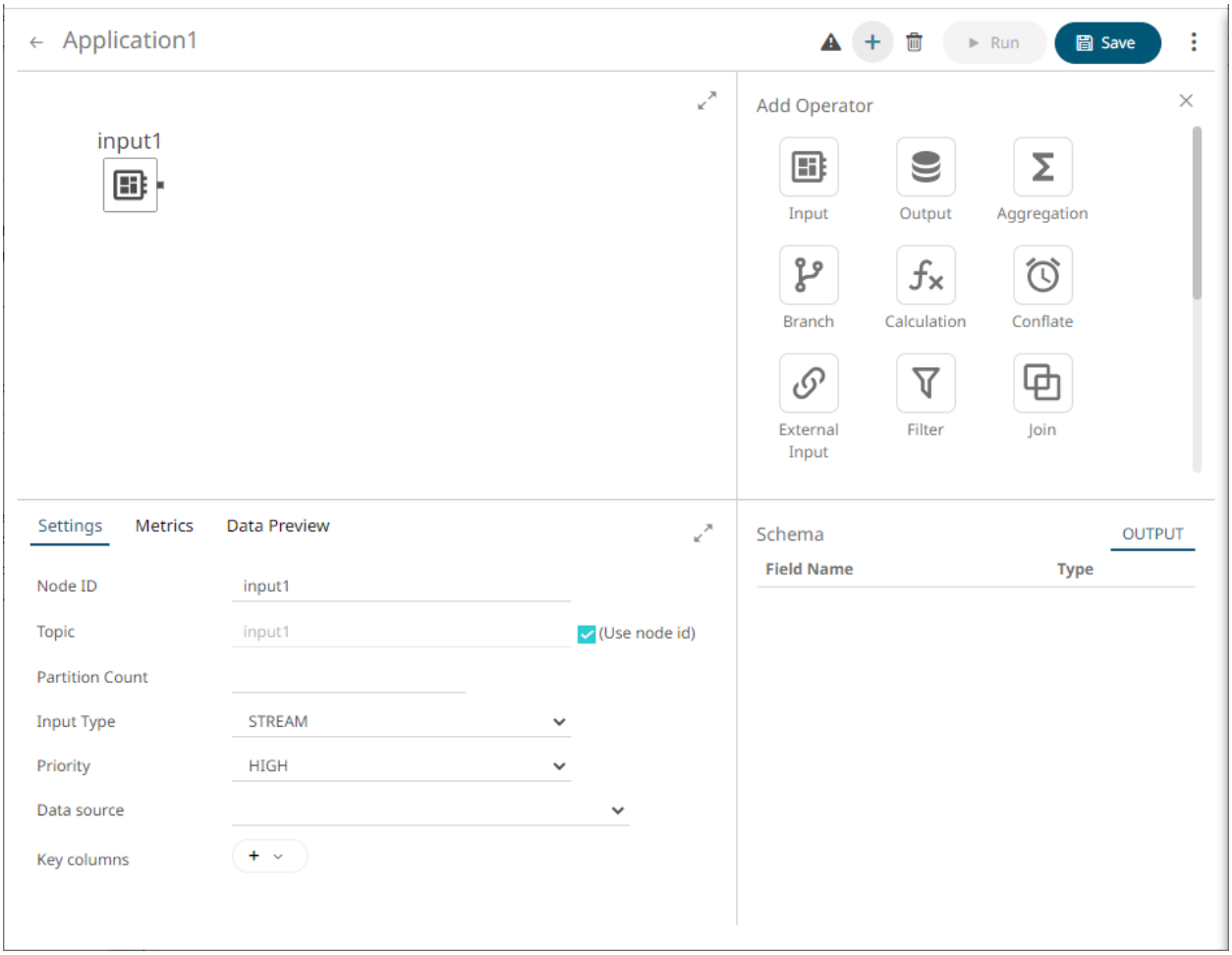
When adding an operator, its node will be displayed in the *Graph* panel.



NOTE

- **The edges (inbound and/or outbound) will depend on the operator.**
- **For best practice, start by adding Input operators (i.e., Input, External Input, or Metronome) and end with the Output operator.**

Also, the corresponding *Operator Properties* and *Schema* are displayed in the *Application* page.

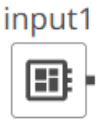


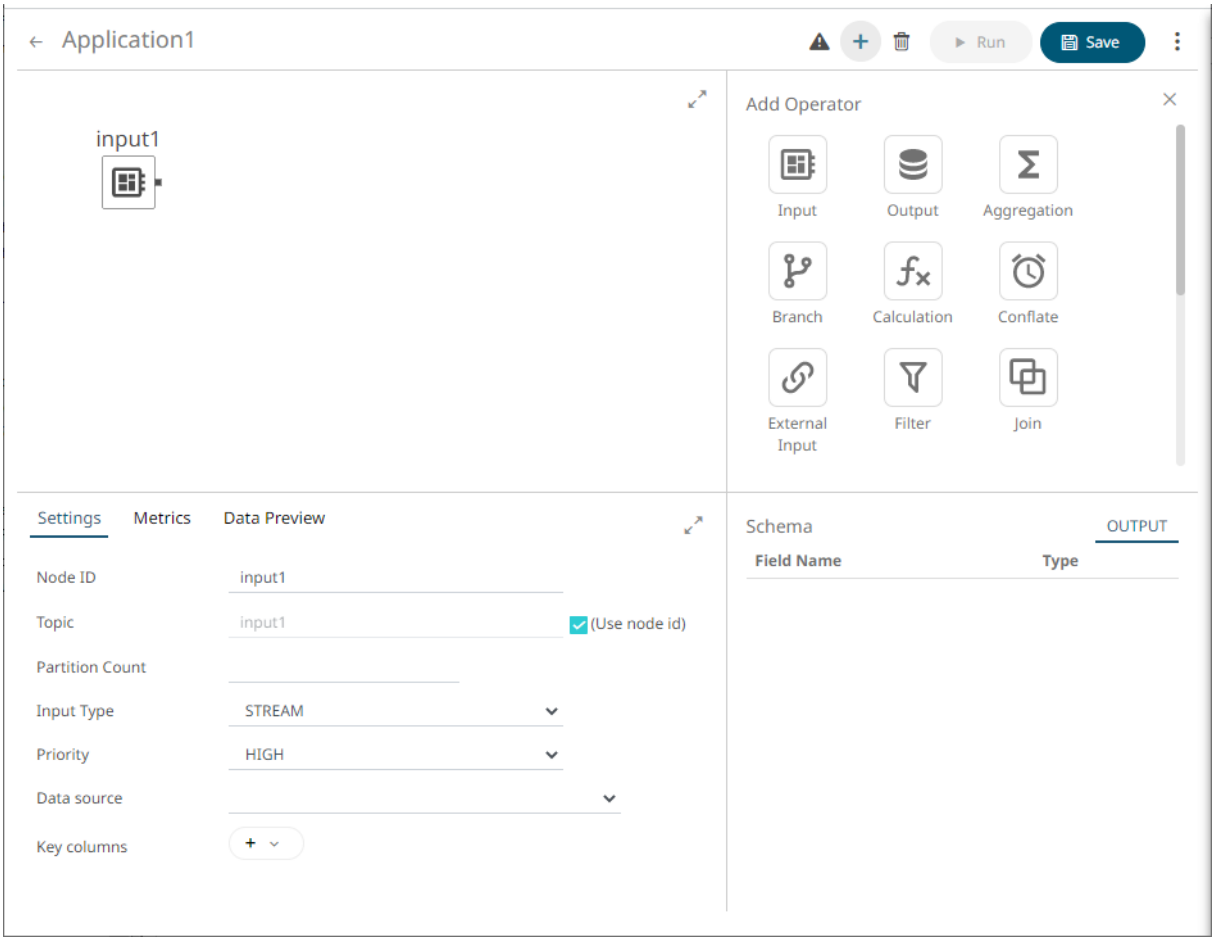
Adding an Input Operator

Used to define the input data for the application model.

Steps:

1. In the *Application* page, click + and select **Input** in the *Add Operator* pane.

   The **Input** node  icon displays in the *Graph* pane, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



This operator serves as the initial source of the data in the application. The right (outbound) edge allows you to connect to other operators.

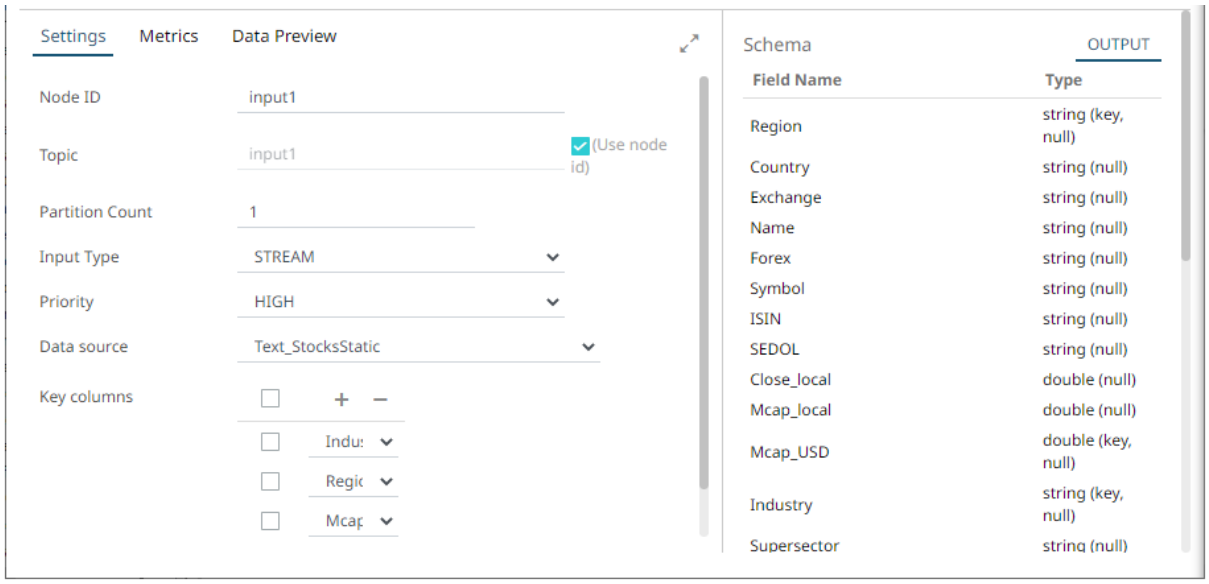
2. In the *Operator Settings* panel, define or select the following properties:

Property	Description
Node ID	The ID of the input operator.
Topic	The stream of records or input you will be subscribed to. Check the *Use Node ID* box to use the value entered in the *Input* ID. Otherwise, uncheck the box and enter a new *Topic* ID. When adding Topic IDs, ensure they: • must be unique across an application • must be specified • must start with a letter (a to Z) or an underscore. Also, it can only contain letters (a to Z), numbers (0 to 9), and underscores
Partition Count	Enter the number of partitions for the Kafka topics that will be created for the Input operator. Partitions allow you to parallelize a topic by splitting the data in a particular topic across multiple brokers wherein, each partition can be placed on a separate machine to allow for multiple consumers to read from a topic in parallel.

Input Type	Select the input type: **STREAM**, **TABLE**, or **GLOBAL_TABLE**. STREAM will treat incoming data as a stream of records while TABLE creates a "materialized view" or snapshot table, representing the latest state of received key/value pairs. GLOBAL_TABLE can be seen as a materialized view that is distributed across all partitions. This is useful for keeping small, relatively static, data sets that needs to be joined with streaming data.
Priority	Select the priority of the node's startup: • **APPLICATION** – running and successful completion of the node is critical in the application startup. • **HIGHEST** – highest priority but not critical. • **HIGH** (Default) – high priority but not critical. • **STANDARD** – standard priority. • **LOW** – low priority.
Data Source	Select the data source. **NOTES:** • It is recommended to upload the data source first so they will be available for selection. • Selecting a non-streaming data source displays the *Refresh Period* (ms) property. Enter the refresh period for the data. This value determines when to periodically reload the data (from the beginning). • Setting the *Refresh Period* to any value less than or equal to zero will disable automatic data reload. The preview of the data (OUTPUT) is displayed in the *Schema* panel.
Key Columns	In Kafka, all messages are processed in a key/value fashion where the value represents the actual data payload or record. The key is used to determine how the key/value pairs are distributed across available partitions. If the key is null a round-robin approach is used to determine partition. For the TABLE and GLOBAL_TABLE input type, key is also essential for defining how records are segregated (keyed) in the table. Not providing a key will result in a single-row table. Post input, keying of records can be changed by using either the Rekey or Aggregation operators.

NOTE ***Node ID, Topic, Input Type*, and *Data Source* properties are required.**

3. In the *Key Columns* section, click + to add a key column from the data source schema. Repeat to add more.



You can also delete a key column in the list by checking its box and clicking − .

4. Save the changes.

Example

```
<input>
    <id>Input</id>
    <topic>Input</topic>
    <dataProducer>
        <id>StocksStatic</id>
        <refreshPeriod>1000</refreshPeriod>
        <keyColumns>
            <field>Id</field>
        </keyColumns>
    </dataProducer>
    <inputType>TABLE</inputType>
</input>
```

Adding An Aggregation Operator

The aggregation operator aggregates the data based on a grouping key and a set of aggregated fields.

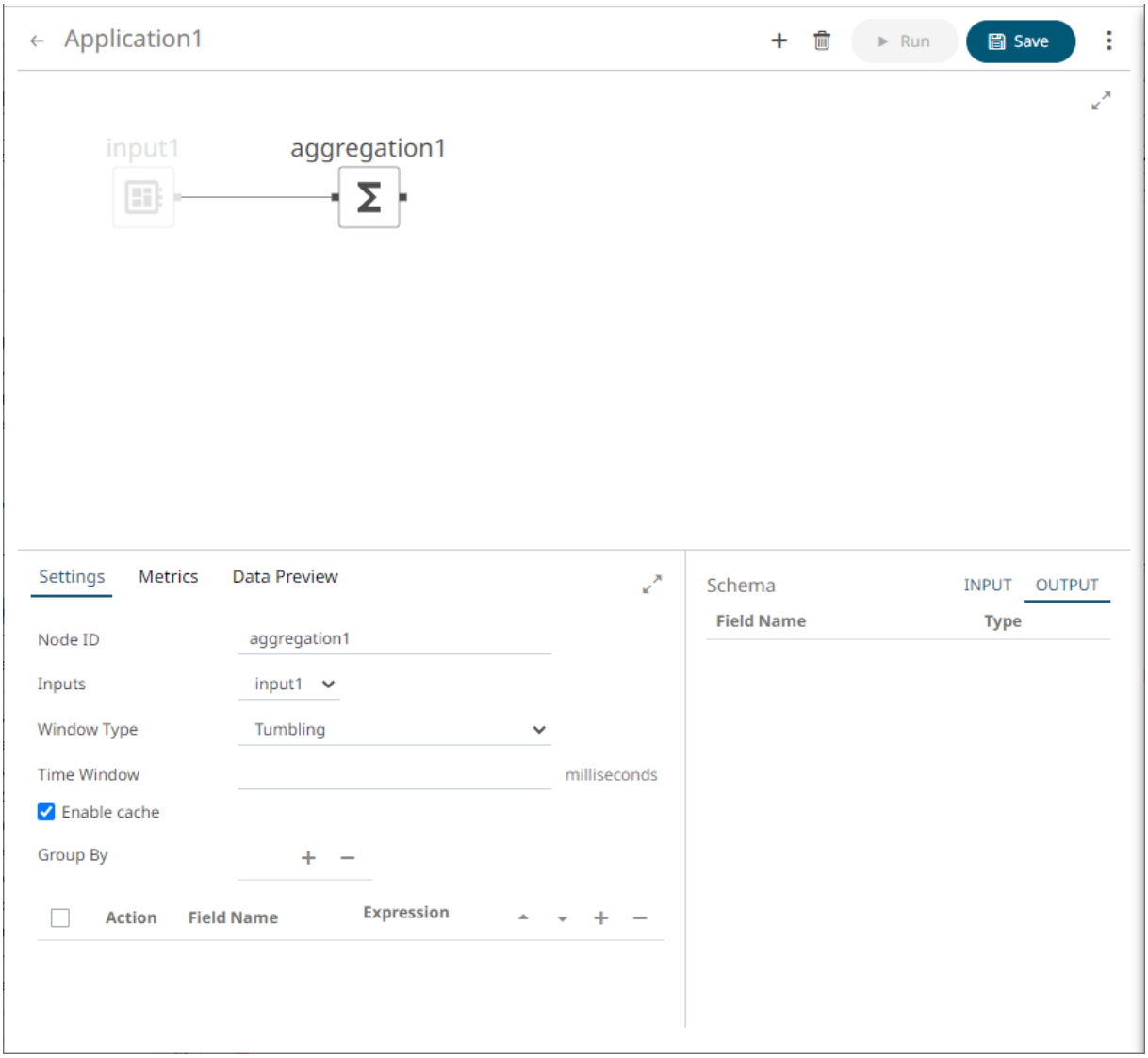
Steps:

1. In the *Application* page, click + and select **Aggregation**

in the *Add Operator* pane.

The **Aggregation** node

icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



This operator has left (inbound) and right (outbound) edges that allow connection to other operators in the application.

2. In the *Operator Settings* panel, define or select the following required properties:

Property	Description
Node ID	The ID of the aggregation operator.
Inputs	Automatically connects to the currently selected operator. You can select another ID of the operator that will be the source of the data in the *Inputs* drop-down list. The preview of the data (INPUT) is displayed in the *Schema* panel.

Window Type	Select either: • **Tumbling** (default) A series of fixed-sized, non-overlapping, and adjoining time intervals. This window type is a moving window whose window size (*Time Window*) is equal to its advance interval. Since tumbling windows never overlap, a data record will belong to only one window. • **Hopping** This window type models fixed-sized, scheduled overlapping windows. Defined by the window's size (*Time Window*) and advance interval (*Advance Period*).
Time Window	The window's size (in milliseconds).
Advance Period	The advance interval for the *Hopping Window* (in milliseconds). Specifies by how much a window moves forward relative to the previous one.
Enable Cache	Specifies whether to start or stop caching tables. When caching is stopped, it gets every event input into the table to produce an output event. This is necessary for delta/prev aggregates.
Group By	The name/IDs of the columns that the data will be grouped by. (Proceed to step 3.)
Fields List	A set of aggregated fields (with actions and expressions). (Proceed to step 5.)

3. In the *Group By* section, click + . A column is added in the list. Click the drop-down list to select another column.
4. Select a column that will be used to group the data.

   The INPUT and OUTPUT schema are displayed.

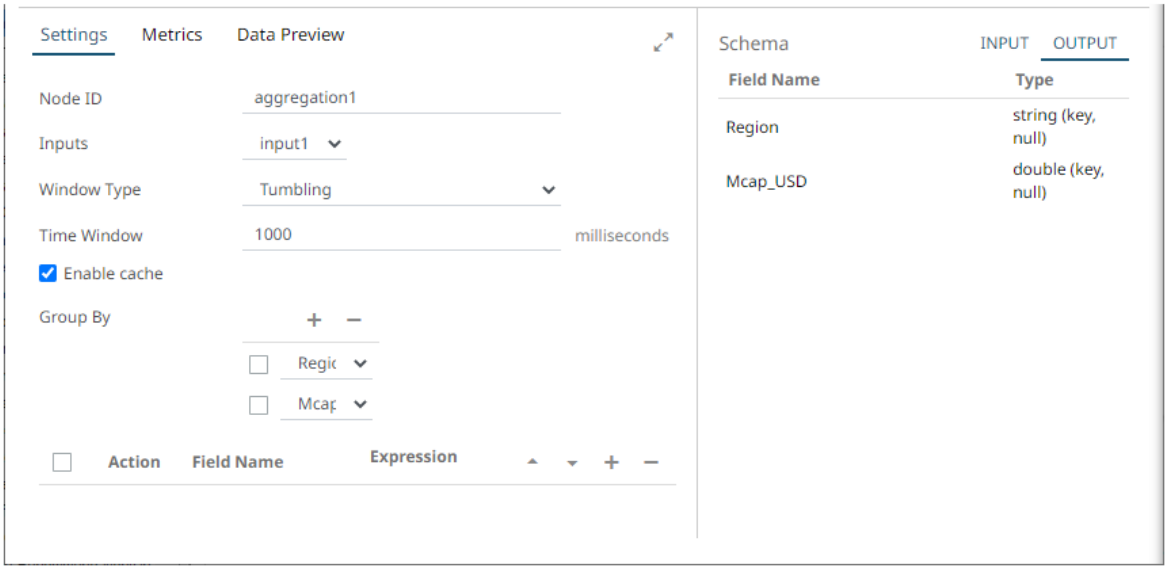
Schema INPUT OUTPUT

Field Name	Type
Region	string (key, null)
Country	string (null)
Exchange	string (null)
Name	string (null)
Forex	string (null)
Symbol	string (null)
ISIN	string (null)
SEDOL	string (null)
Close_local	double (null)
Mcap_local	double (null)
Mcap_USD	double (key, null)
Industry	string (key, null)

Schema INPUT OUTPUT

Field Name	Type
Region	string (key, null)

Repeat steps 3 and 4 to add more.



You can also delete a column in the *Group By* list by checking its box and clicking .

5. In the *Field List* section, click . A new field entry displays.

6. Enter the *Field Name* and the *Expression* that will be evaluated for each incoming record.

   Example:

   Field Name: Count

   Expression: count()

7. Select the **Add** action.

   Repeat steps 6 and 7 to add more aggregated fields.

Action	Field Name	Expression
Add	Count	count()
Add	Samples	samples(Mcap_USD)
Add	Sum_Mcap_USD	sum(Mcap_USD)
Add	First_Close_local	first(Close_local)
Add	Last_Close_local	last(Close_local)
Add	Min_One_Day_Change	min(One_Day_Change)
Add	Max_One_Day_Change	max(One_Day_Change)
Add	Avg_One_Day_Change	avg(One_Day_Change)
Add	Varp_One_Day_Change	varp(One_Day_Change)
Add	Vars_One_Day_Change	vars(One_Day_Change)

You can also:

- check the topmost box to select all of the fields
- change the order of the fields by checking a field's box and clicking either the ▲ or ▼ button
- delete a field entry in the Field List by checking its box and clicking −

The OUTPUT schema is updated based on the added aggregations.

Schema INPUT OUTPUT

Field Name	Type
Industry	string (key, null)
Count	long (not null)
Sum_Mcap_USD	double (not null)
First_Close_local	double (null)
Last_Close_local	double (null)
Min_One_Day_Change	double (null)
Max_One_Day_Change	double (null)
Samples	long (not null)
Avg_One_Day_Change	double (null)
Varp_One_Day_Change	double (null)
Vars_One_Day_Change	double (null)
Sdevp_One_Day_Change	double (null)
Sdevs_One_Day_Change	double (null)

8. Save the changes.

Example

```
<aggregation>
    <id>Aggregation</id>
    <fields>
        <field>
            <id>ColumnId</id>
            <action>ADD</action>
            <expression>Sum(Mcap_USD)</expression>
        </field>
    </fields>
    <groupBy>
        <field>Industry</field>
    </groupBy>
</aggregation>
```

Building the Expression

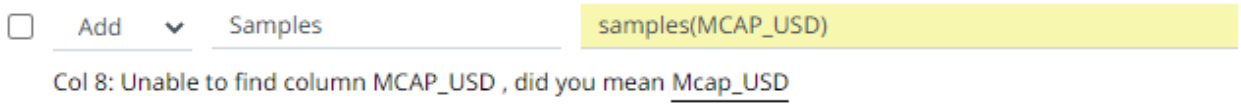
To build the expression, you can either:

- manually enter into *Expression* text box

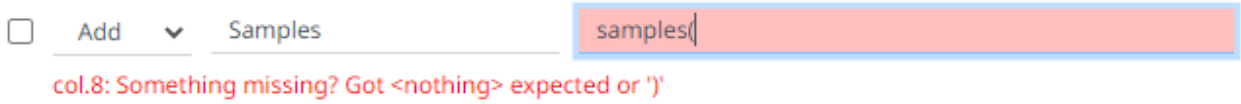
  Take note that the column name is case sensitive.

  A validation error displays with a suggestion to help build the expression.

  Examples:



Click the link (e.g., [**Mcap(USD)**] to apply the correct entry.



Complete the expression as necessary.

- use the Expression Builder

Using the Expression Builder

Create new fields using data from existing columns in your input operator and supported aggregation or calculation functions and operators.

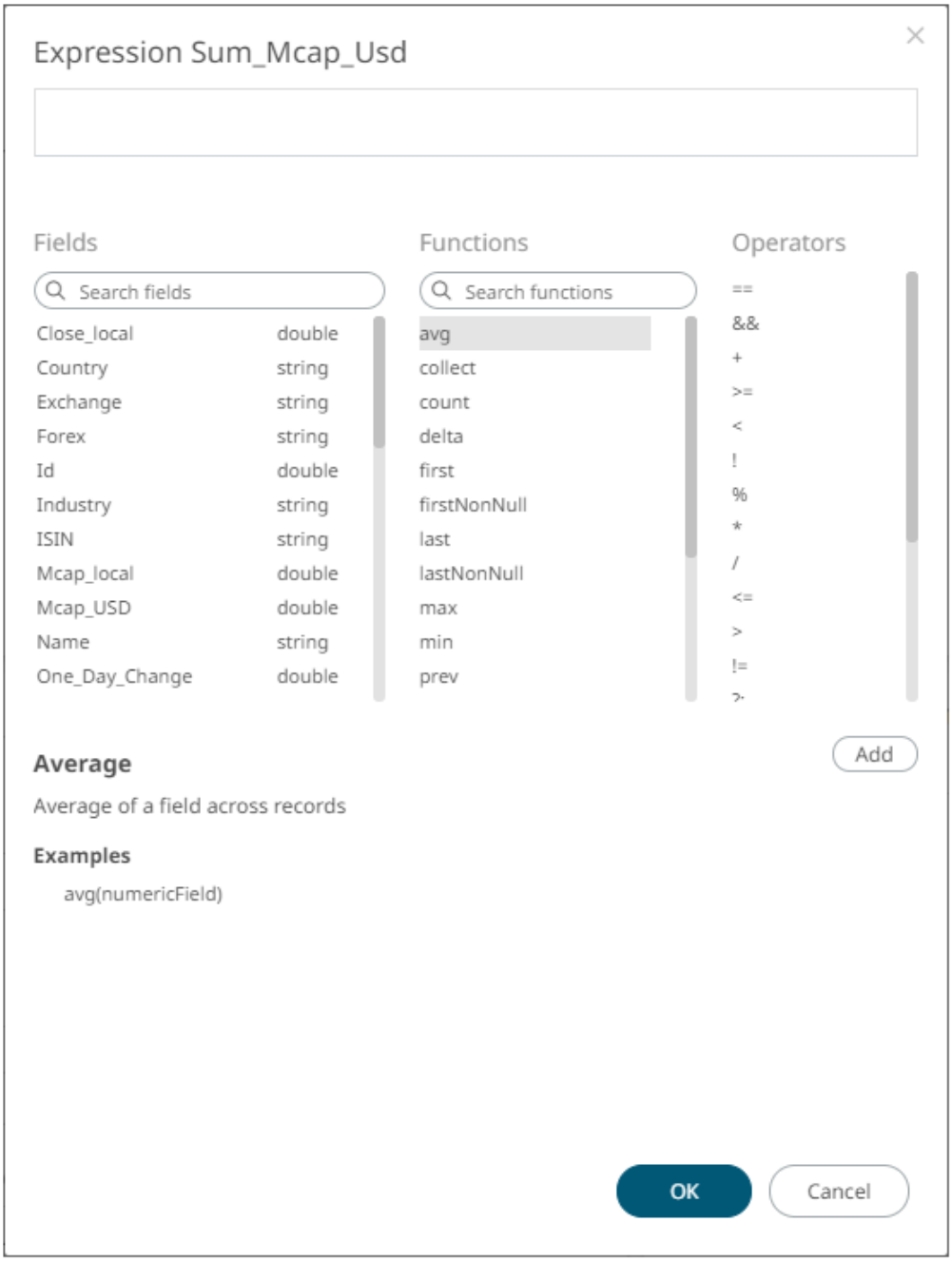
Steps:

1. On the *Fields List* section, click + to add a new field instance.

2. Enter the *Field Name*.

3. Click the **New Expression** icon.

   The *Expression <Field Name>* dialog displays.

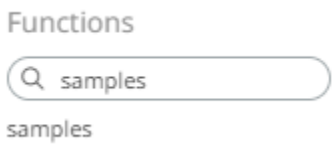


4. Build an expression by double-clicking in the list of *Functions*, *Fields*, and *Columns*.

   You can also click on a function and operator then

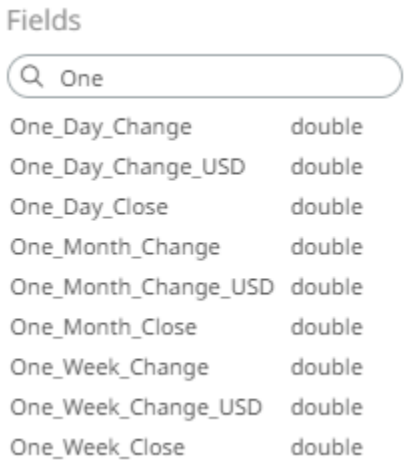
.

   To search for a particular column or function, enter it in the *Search Fields/Search Functions* box.



Or enter one or more characters/alphabets into the *Search Fields* box and the suggested list of columns that matched the entries will be displayed.

For example, after entering **One**, the list will be displayed such as below:



5. Click

. The new expression is added to the field instance.

Supported Aggregation Functions

This section lists the aggregation functions that are only supported in aggregation operator expressions.

NOTE

Panopticon Streams also supports nullability where:

- **a field may or may not allow null/empty/missing/NA values.**
- **functions or operators may or may not allow null arguments (e.g., you can't divide seven by null).**

Aggregation Function	Description	Example	Nullability
avg(X)	Average of a field across records.	avg(numericField)	✔
collect(X)	Returns last n values of a field across records. **collect(fieldName,valuesCount)** Parameters: • fieldName – Field name whose last n values should be retained. • valuesCount – Number of values to be retained.	collect(symbol,3)	✔
count(X)	Counts the number of records.	count()	
delta(X)	Returns the difference of the last and second last value of the integer field.	delta(numericField)	✔
first(X)	First value of a field.	first(fieldName)	✔
firstNonNull(X)	First non-null value of a field.	firstNonNull(fieldName)	✔
last(X)	Last value of a field.	last(fieldName)	✔
lastNonNull(X)	Last non-null value of a field.	lastNonNull(fieldName)	✔
max(X)	Maximum of an integer field across records.	max(fieldName)	✔
min(X)	Minimum of an integer field across records.	min(fieldName)	✔
prev(X)	Previous values of a field record.	prev(fieldName)	✔
samples(X)	Count of non-null values of field records.	samples(fieldName)	✔
sdevp(X)	Returns the standard deviation of an entire population.	sdevp(numericField)	✔
sdevs(X)	Estimates standard deviation based on a sample.	sdevs(numericField)	✔
sum(X)	Sums a field across records.	sum(numericField)	✔
varp(X)	Returns the variance in an entire population.	varp(numericField)	✔
vars(X)	Returns the variance based on a sample.	vars(numericField)	✔
wavg(X)	Weighted average of a field across records. **wavg(score,weight)** Parameters: • score – Numeric field for score. • weight – Numeric field for weightage.	wavg(marks,weight)	✔

NOTE **The following aggregates work with a time window (can subtract): count, samples, sum, sdev, var, and avg**

Converting Timestamp to/from Integer

Allows you to convert Timestamp values to/from Integer which include the following examples:

- from posix to timestamp
- from posixmillis to timestamp
- from timestamp to posix
- from timestamp to posixmillis

The conversion uses the expression: `to([typename],[expression],[format]`

Examples:

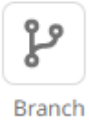
- to(int, timefieldname, 'POSIX')
- to(time, intfieldname, 'POSIX')

Adding a Branch Operator

The branch operator will split a stream into one or more branches. The path for a stream is determined by a configured predicate within the branch operator.

The predicate expression will be evaluated for each incoming record. A record will be routed to the first branch with a matching predicate.

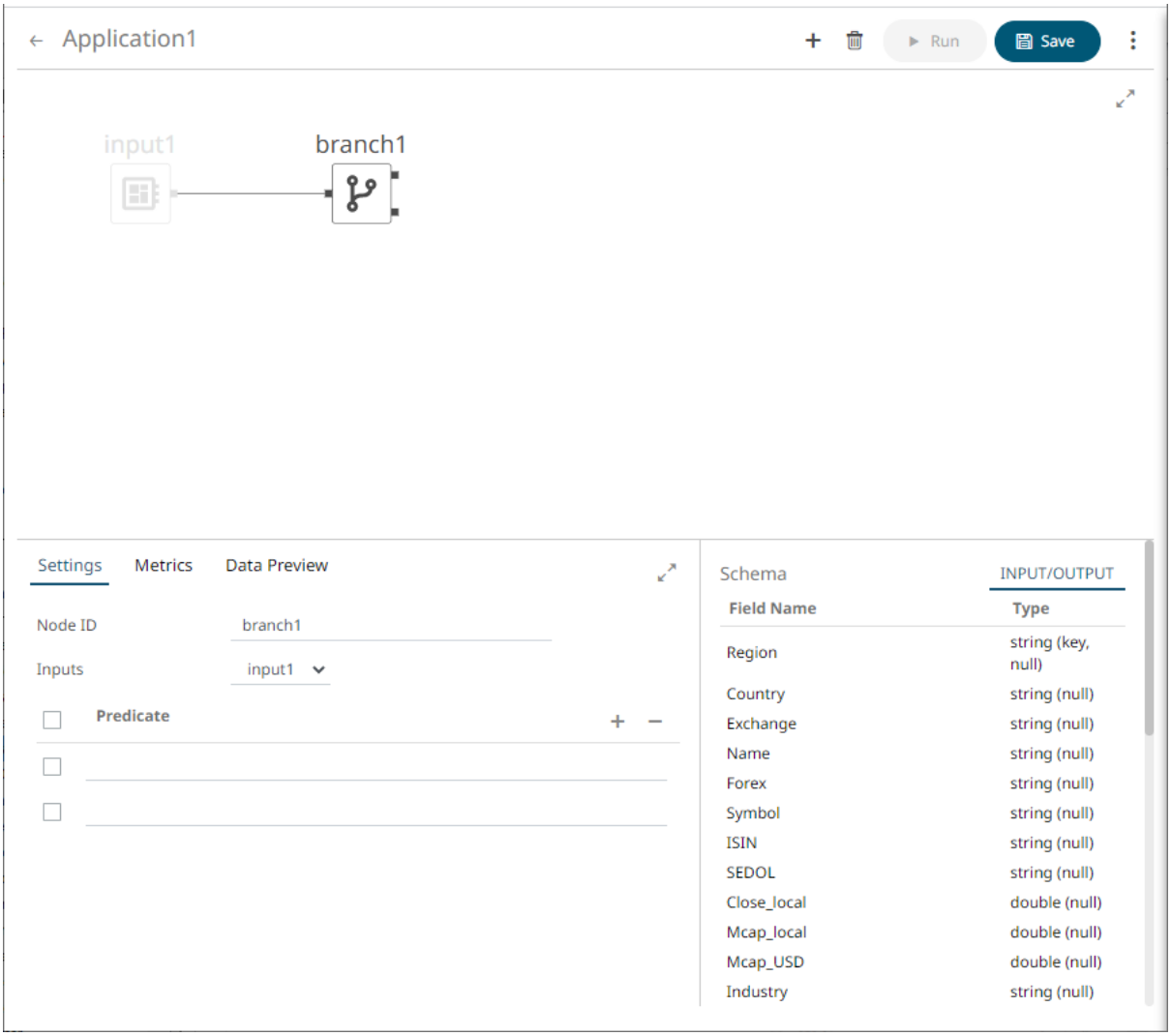
Steps:

1. In the *Application* page, click + and select **Branch** 

in the *Add Operator* pane.

   The **Branch** node

icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



The left (inbound) edge allows you connect to an input data or operator. The right (outbound) edges allow you to add more streams.

2. In the *Operator Settings* panel, define or select the following required properties:

Property	Description
Node ID	The ID of the branch operator.
Inputs	Automatically connects to the currently-selected operator. You can select another ID of the operator that will be the source of the data in the *Inputs* drop-down list. The preview of the data (INPUT/OUTPUT) is displayed in the *Schema* panel.
Predicate	A list of predicates. Each predicate contains an expression that will be evaluated for each record.

3. To add more predicates, click + . A new predicate entry displays.

   Enter at least two expressions.

	Predicate
☐	One_Day_Change < 0
☐	One_Day_Change >= 0

You can also:

- check the topmost box to select all of the fields
- delete a field entry in the Field List by checking its box and clicking −

4. Save the changes.

Example

```
<branch>
    <id>Branch</id>
    <predicates>
        <!-- One_Day_Change < 0 -->
        <predicate>One_Day_Change < 0</predicate>
        <!-- One_Day_Change >= 0 -->
        <predicate>One_Day_Change >= 0</predicate>
    </predicates>
</branch>
```

Example 2

```
<streams>
    <stream>
        <source>Input</source>
        <sink>
            <operator>Branch</operator>
        </sink>
    </stream>
    <stream>
        <source>Branch</source>
        <port>1</port>
        <sink>
            <operator>Output1</operator>
        </sink>
    </stream>
    <stream>
        <source>Branch</source>
        <port>2</port>
        <sink>
            <operator>Output2</operator>
        </sink>
    </stream>
</streams>
```

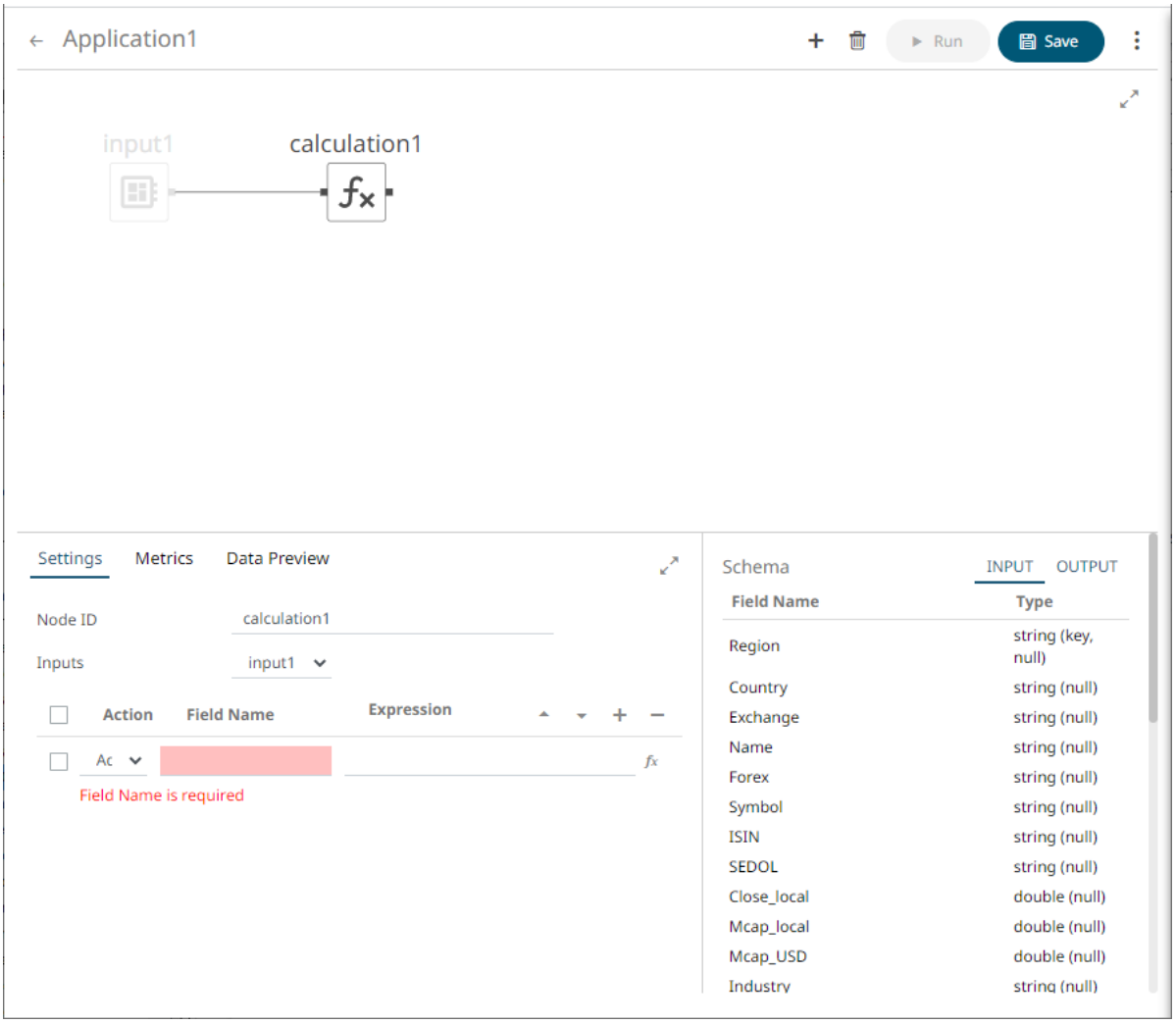
Adding a Calculation Operator

The calculation operation will calculate a field and add the result as an additional field. Usually, input fields pass through an operation, but calculations can also be set to replace existing fields or simply remove them.

Steps:

1. In the *Application* page, click + and select **Calculation** Calculation in the *Add Operator* pane.

   The **Calculation** node calculation1 icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



   This operator has left (inbound) and right (outbound) edges that allow connection to other operators in the application.

2. In the *Operator Settings* panel, define or select the following required properties:

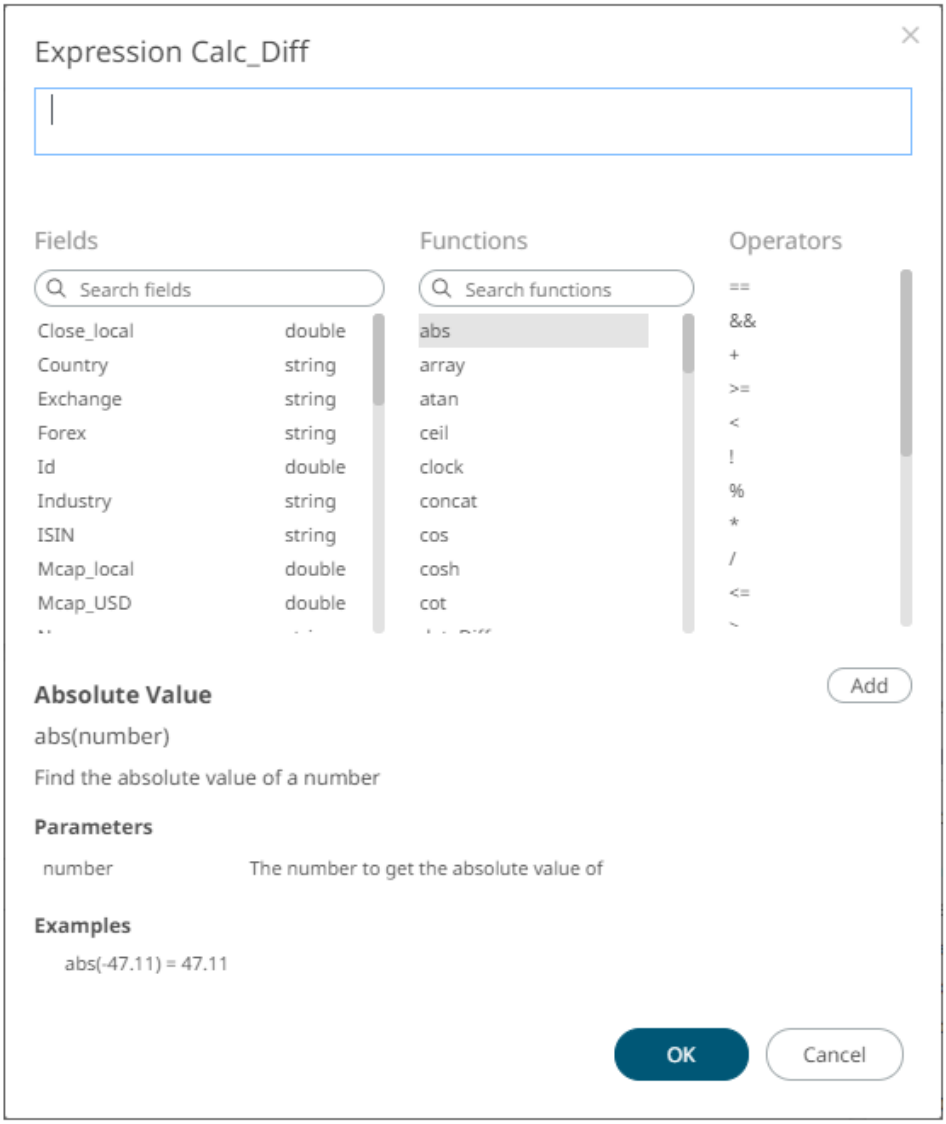
Property	Description
Node ID	The ID of the calculation operator.
Inputs	Automatically connects to the currently-selected operator.

	You can select another ID of the operator that will be the source of the data in the *Inputs* drop-down list. The preview of the data (INPUT and OUTPUT) are displayed in the *Schema* panel.
Fields List	Set of fields (with actions and expressions). Enter at least one calculated field. Proceed to step 3.

3. Enter the *Field Name* and the *Expression* that will be evaluated for each incoming record.

   To use the expression builder, click the **New Expression** icon.

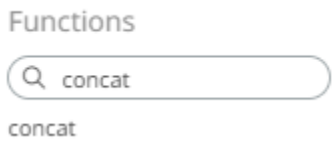
   The *Expression <Field Name>* dialog displays.



   Build an expression by double-clicking in the list of *Functions*, *Fields*, and *Columns*.

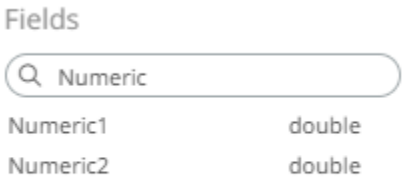
   You can also click on a function and operator then Add .

To search for a particular column or function, enter it in the *Search Fields/Search Functions* box.



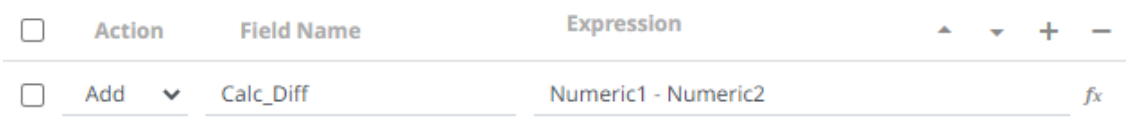
Or enter one or more characters/alphabets into the *Search Fields* box and the suggested list of columns that matched the entries will be displayed.

For example, after entering **Numeric**, the list will be displayed such as below:

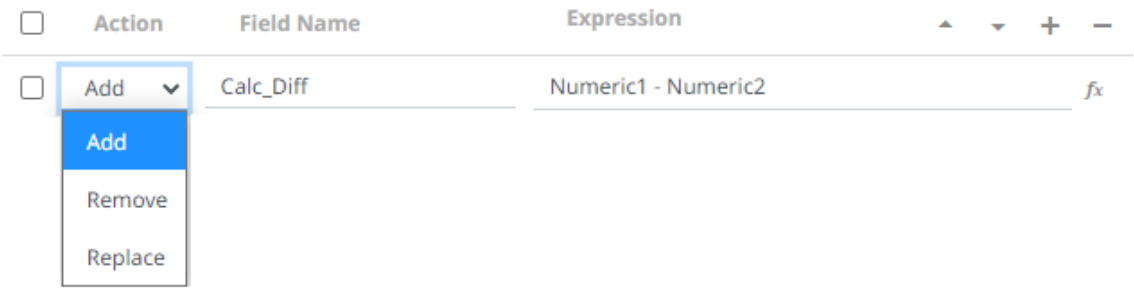


Click

. The new expression is added to the field instance.

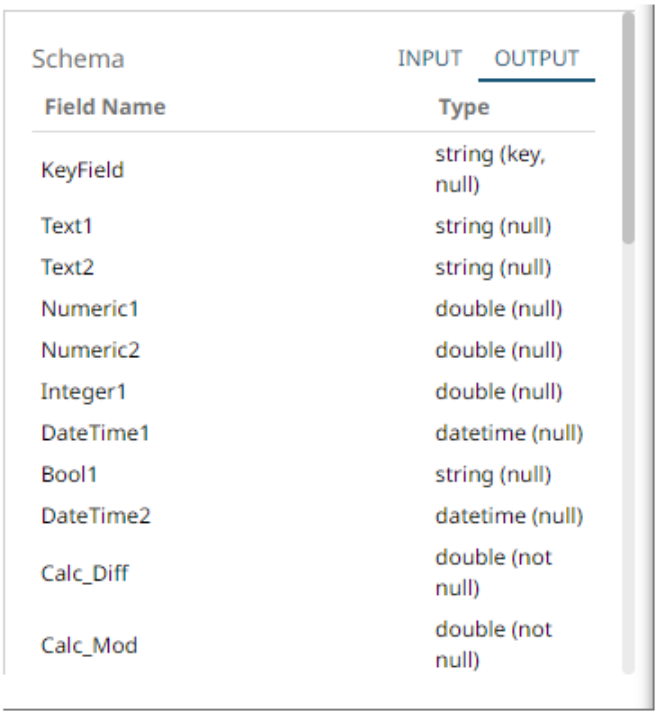


4. Select any of the following actions: **Add, Replace**, or **Remove**.



5. Click + to add a new field entry and repeat steps 3 and 4.

   The OUTPUT schema is updated based on the added calculations.



Schema INPUT OUTPUT

Field Name	Type
KeyField	string (key, null)
Text1	string (null)
Text2	string (null)
Numeric1	double (null)
Numeric2	double (null)
Integer1	double (null)
DateTime1	datetime (null)
Bool1	string (null)
DateTime2	datetime (null)
Calc_Diff	double (not null)
Calc_Mod	double (not null)

You can also:

- check the topmost box to select all of the fields
- change the order of the fields by checking a field's box and clicking either the ▲ or ▼ button
- delete a field entry in the Field List by checking its box and clicking −

6. Save the changes.

Supported Operators and Calculation Functions

This section lists the supported operators and calculation functions in Panopticon Streams.

NOTE

Panopticon Streams supports nullability where:

- **a field may or may not allow null/empty/missing/NA values**
- **functions or operators may or may not allow null arguments (e.g., you can't divide seven by null)**

Supported Operators

These are typically the operators that are used to create aggregation operator and calculation operator expressions.

Operator	Name	Description	Example	Nullability
!	Logical NOT	! boolean Reverse arguments or results. Parameter:		

		• boolean – A value of logical expression that can be evaluated as TRUE or FALSE.		
!=	Not Equal To	Tests if one value is not equals to another.	12.3 != 47.11 = true	✔
%	Modulo	number % divisor Gets the remainder from division. Parameters: • number – The Number to be divided. • divisor – The number to divide with.	17 % 5 = 2	
&&	Logical AND	Returns true only if both the conditions return true.	(7 > 5)&&(3 < 8) = true	
*	Multiplication	Multiply	9 * 5 = 45	
+	Addition	Adds two numbers or joins two or more text strings to produce a single piece of text.	47.11 + 9.7 = 56.81	
-	Subtraction	Subsctracts two numbers.	47 – 11 = 36	
/	Division	number / divisor Parameters: • number – The number to be divided. • divisor – The number to divide with.	11 / 5 = 2	
<	Less than	Tests if one value is less (smaller) than another.	4 < 7 = true	
<=	Less Than or Equal To	Tests if one value is less than or equal to another.	7 <= 4 = false	
==	Equal To	Tests if one value is equals to another.	9 == 5 = false	✔
>	Greater Than	Tests if one value is greater (larger) than another.	7 > 4 = true	
>=	Greater Than or Equal To	Tests if one value is greater or equal to another.	7 >= 4 = true	
?:	Ternary IF	Provides branching capability. If condition is true, then it returns exprIfTrue, else returns exprIfFalse. condition ? exprIfTrue : exprIfFalse Parameters: • condition – A value or logical expression that can be evaluated as true or false. • exprIfTrue – The value to return when condition evaluates to true. • exprIfFalse - The value to return when condition evaluates to false.		✔
^	Exponentiation	Get the exponential value of the number.	(2.0) ^ 7.0 = 128.0	
like	Like	Tests if the pattern exists in the text.	‘olle’ like pattern(‘.ll.’) = true	
||	Logical OR	Returns true if any of the conditions return true.	(7 > 5) || (3 > 8) = true	

-	Negation	Negation of the number.	-(1 + 2) = -3	
+	Positivity	Positivity of the number.		
()	Cast Operator	Explicit data type conversion. Valid casts are: • 'int' • 'real' • 'text' • 'time' • 'bool'	(int)sqr(Numeric1)	

Supported Calculation Functions

Operator	Name	Description	Example	Nullability
abs	Absolute Value	**abs(number)** Find the absolute value of a number. Parameter: • number – The number to get the absolute number of.	abs(-47.11) = 47.11	
array	Array	**array(size,value)** Create an array of specified size and initialize with the given value. Parameters: • size – Size of the array. • value – Array elements.	array(5, 1.1)	✔
atan	Arc Tangent	**atan(ordinate, abscissa)** Get the inverse tangent of a number. Parameters: • ordinate – The ordinate coordinate. • abscissa – The abscissa coordinate (optional).	atan(1.0) = 0.785398	
ceil	Ceiling	**ceil(number)** Returns the smallest double value that is greater than or equal to the argument and is equal to a mathematical integer. Parameter: • number – The number to get the ceiling value of.	ceil(4.7) = 5.0	
clock	Clock	Returns the current system Date/Time value.	clock()	
concat	Concatenate	**concat(text1,text2)** Joins two text values. Parameters: • text1 – First text to join.	concat('olle','pelle') = 'olle pelle'	

		• text2 – Second text to join.		
cos	Cosine	**cos(number)** The natural logarithm (base e) of a real value. Parameter: • number – A number to take the natural logarithm of .	cos(0.0) = 1.0	
cosh	Hyperbolic Cosine	**cosh(number)** Get the hyperbolic cosine of the number. Parameter: • number – The number to get the hyperbolic cosine of.	cosh(0.0) = 1.0	
cot	Cotangent	**cot(number)** Get the cotangent of the number. Parameter: • number – The number to get the cotangent of.		
dateDiff	Date Difference	**dateDiff(startDatetime,endDatetime,unit)** Calculates the difference in whole units between two time values, the return value is positive if endDatetime comes after startDatetime, and is negative if endDatetime comes before startDatetime. Valid units are ‘DAYS’, ‘HOURS’, ‘MINUTES’, ‘SECONDS, ‘’MILLISECONDS’, ‘MICROSECONDS’, AND ‘NANOSECONDS’. Parameters: • startDatetime – The first (later) Date/Time value. • endDatetime – The second (earlier) Date/Time value. • unit – The time unit to use.	dateDiff(#2019-06-17#,#2019-06-14#,’DAYS’) = 3	
dateDiff2	Date Difference	Calculates the difference in fraction units between two time values, the return vallue is positive if endDatetime comes after startDatetime, and is negative if endDatetime comes before startDatetime. Valid units are ‘DAYS’, ‘HOURS’, ‘MINUTES’, ‘SECONDS, ‘’MILLISECONDS’, ‘MICROSECONDS’, AND ‘NANOSECONDS’. Parameters: • startDatetime – The first (later) time value. • endDatetime – The second (earlier) time value. • unit – The time unit to use.	dateDiff2(#2019-06-17T12:00:00#,#2019-06-14T00:00:00#,’DAYS’) = 3.5	
datePart	Date Part	Returns a specified part of a time value, the result is an integer value.	datePart(#1973-07-23#,’YEARS’) = 1973	

		Valid units are ‘DAYS’, ‘HOURS’, ‘MINUTES’, ‘SECONDS, ‘’MILLISECONDS’, ‘MICROSECONDS’, AND ‘NANOSECONDS’. Parameters: • datetime – The Date/Time value. • part – The time part to get.		
dateTrunc	Date Truncate	**dateTrunc(datetime,datetimePart)** Truncates the specified time value to the accuracy specified by the time_part. Valid units are ‘DAYS’, ‘HOURS’, ‘MINUTES’, ‘SECONDS, ‘’MILLISECONDS’, ‘MICROSECONDS’, AND ‘NANOSECONDS’. Parameters: • datetime – The Date/Time value. • datetimepart – The Date/Time part to be truncated.	dateTrunc(#1973-07-23T12:34:56#,’YEARS’) = #1973-01-01T00:00:00#	
exp	Exponential	**exp(number)** Find the value of e raised to the power of a number. Parameter: • number – The power that e is raised to.	exp(0.0) = 1.0	
find	Find	**find(findText,withinText,startPosition)** Returns the starting position of one string within another string, given a starting position. Parameters: • findText – The text to find. • withinText – The text to search witihin. • startPosition – Start the search from this position (optional).	find(‘ab’,’drabant’) = 3	
floor	Floor	**floor(number)** Returns the largest real value that is less than or equal to the argument and is equal to a mathematical integer. Parameter: • number – The number to get the flloor value of.	floor(4.7) = 4.0	
get	Get	**get(array,position)** Gets the nth element of the array. Parameters: • array – Array of items. • position – Array element at this position. **NOTE:** Index of the array starts with 0.	get(array(5,1.1),1) = 1.1	✔
if	If	**if(condition,exprIfTrue,expIfFalse)** The function provides branching capability. If condition is true, then it returns exprIfTrue, else it returns exprIfFalse.	if(a < b, a,b)	✔

		Parameters: • condition – A value or logical expression that can be evaluated as true or false. • exprIfTrue – The value to return when condition evaluates to true. • exprIfFalse – The value to return when condition evaluates to false.		
ifNull	ifNull	**ifNull(expression,altValue)** The ifNull function returns the specified value if the expression is null, otherwise returns the expression. Parameters: • expression – The expression to test whether it is null. • altValue – The value to return if the expression is null.	ifNull(null, 'b') = 'b'	✔
index	Index	**index(array,text)** Sorts the input array and outputs a lookup index. Parameters: • array – Array of items. • text – The order which the array should be sorted, the valid texts are 'asc' for ascending order or 'desc' for descending order.	index(array(3,#1973-07-23#),'asc')	
intpow	Integral Power	**intpow(number,power)** Raise a number to a power. Parameters: • number – Number to raise a power. • power - The power to raise a number to.	intpow(2.0, 3.9) = 8.0	
invert	Invert	Inverts a lookup index. Since the index function returns an inverse permutation, you can apply the invert function which will turn it into a forward permutation (or rank).		
left	Left	**left(text,numofChars)** Returns the leftmost characters from a string producing a new string. Parameters: • text – The text from which to extract characters. • numofChars – Number of characters to be picked from the left.	left('olle', 3) = 'oll'	
len	Length	**len(value)** Returns the number of characters in a string or the number of elements in an array. Parameter: • value – String or array to find the length of.	len('olle') = 4	

ln	Logarithm	**ln(number)** The natural logarithm (base e) of a real value. Parameter: • number – A number to take the natural logarithm of.	ln(1) = 0	
log	Logarithm	**log(number)** Logarithm with base 10. Parameter: • number – Number of which you want the logarithm.	log(1000.0) = 3.0	
logn	Logarithm	**logn(number,logBase)** Returns the Log Based N of Input. Parameters: • number – Number of which you want the logarithm. • logBase – Base of the logarithm.	logn(4711.0,4711.0) = 1.0	
lower	Lower	**lower(text)** Convert text to lower case. Parameter: • text – Text to change case to lower.	lower(‘OLLE’) = ‘olle’	✔
max	Maximum	Maximum of the two numbers.	max(11.0, 47.0) = 47.0	
mid	Mid	**mid(string,startPosition,numofChars)** Returns the characters from the middle of a text string, given a starting position and length. Parameters: • string – The original string. • startPosition – Starting position in string. • numofChars – Length of the substring.	mid(‘olle’, 2,2) = ‘ll’	
min	Minimum	Minimum of the two numbers.	min(47.0, 11.0) = 11.0	
pow	Power	**pow(number,power)** Raise a number to a power. Parameters: • number – Number to raise a power. • power – The power to raise a number to.	pow(-2.0, 7.0) = -128.0	
proper	Proper	**proper(text)** Converts a text to proper case; the first letter in each word in uppercase, and all other letters in the lower case. Parameter: • text – The text to make as a proper case.	proper(‘olle asp’) = ‘Olle Asp’	✔
random	Random	**random(minimumValue,maximumValue)** Returns a random number with a positive sign.	random(12.0) get a floating-point random number greater than	

		Parameters: • minimumValue – Minimum value or a random number (optional). • maximumValue - Maximum value or a random number (optional).	or equal to 0.0 and less than 12.0	
replaceAll	Replace All	**replaceAll(string,oldText,newText)** Replaces all occurences of the pattern with the replacement string. Parameters: • string – The original string. • oldText – The string to be replaced. • newText – The new replacement string.	replaceAll(‘axa’, ‘a’, ‘b’) = ‘bxb’	✔
replaceFirst	Replace First	**replaceFirst(string,oldText,newText)** Replaces the first occurrence of the pattern with the replacement string. Parameters: • string – The original string. • oldText – The string to be replaced. • newText – The new replacement string.	replaceFirst(‘axa’, ‘a’, ‘b’) = ‘bxa’	✔
right	Right	**right(text,numofChars)** Returns the rightmost characters from a string producing a new string. Parameters: • text – The text from which to extract characters. • numofChars – Number of characters to be picked from the right.	right(‘olle’, 3) = ‘lle’	
rnd	Rnd	**rnd(scaleValue)** Returns a random number with a positive or negative sign depending on scale value. Parameter: • scaleValue – Positive scaleValue will result in a number that is maximum up to it and the negative scaleValue will result in a number that is minimum to it.		
round	Round	**round(number,digits)** Round a number to a given number of digits. Parameters: • number – The number to round. • digits – The place at which number should be rounded (optional).	round(47.11) = 47.0	
set	Set	**set(array,position,newvalue)** Sets the nth element of the array. Parameters: • array – Array of items. • position – Array element at this position.	set(array(5, 1.1), 1, 2.2)	✔

		• newvalue – New value of the element. **NOTE:** Index of first element starts with 0.		
sign	Sign	**sign(number)** Get the sign of a number, returns one if positive, negative one if negative, and zero if zero. Parameter: • number – The number to get the sign of.	sign(7) = 1	
sin	Sine	**sin(number)** Get the sine of the number. Parameter: • number – The number to the sine of.		
sinh	Sinus Hyperbolic	**sinh(number)** Get the sinus hyperbolic of the number. Parameter: • number – The number to the sinus hyperbolic of.	sinh(0.0) = 0.0	
sort	Sort	Applies a lookup index to an array.		
sqr	Square	**sqr(number)** Returns square of the number. Parameter: • number – The number to get the square of.	sqr(3) = 9	
sqrt	Square Root	**sqrt(number)** Returns square root of the number. Parameter: • number – The number to get the square root of.	sqrt(9.0) = 3.0	
tan	Tangent	**tan(number)** Get the tangent of the number. Parameter: • number – The number to get the tangent of.	tan(0.0) = 0.0	
trim	Trim	**trim(text)** Get the input text stripped of leading or following spaces. Parameter: • text – The text to be stripped of leading or following spaces.	trim(‘ olle ’) = ‘olle’	✔
trunc	Truncate	**trunc(number,digits)** Truncate a number to a given precision. Parameters: • number – The number to truncate.	trunc(47.11) = 47.0	

		• digits – The precision of the truncation (optional and the default is 0).		
upper	Upper	**upper(text)** Convert text to upper case. Parameter: • text – Text to change case to upper.	upper(‘olle’) = ‘OLLE’	

Example

```
<calculation>
    <id>Calculation</id>
    <fields>
        <field>
            <id>SquareRoot_ColumnA</id>
            <action>ADD</action>
            <expression>SquareRoot(ColumnA)</expression>
        </field>
    </fields>
</calculation>
```

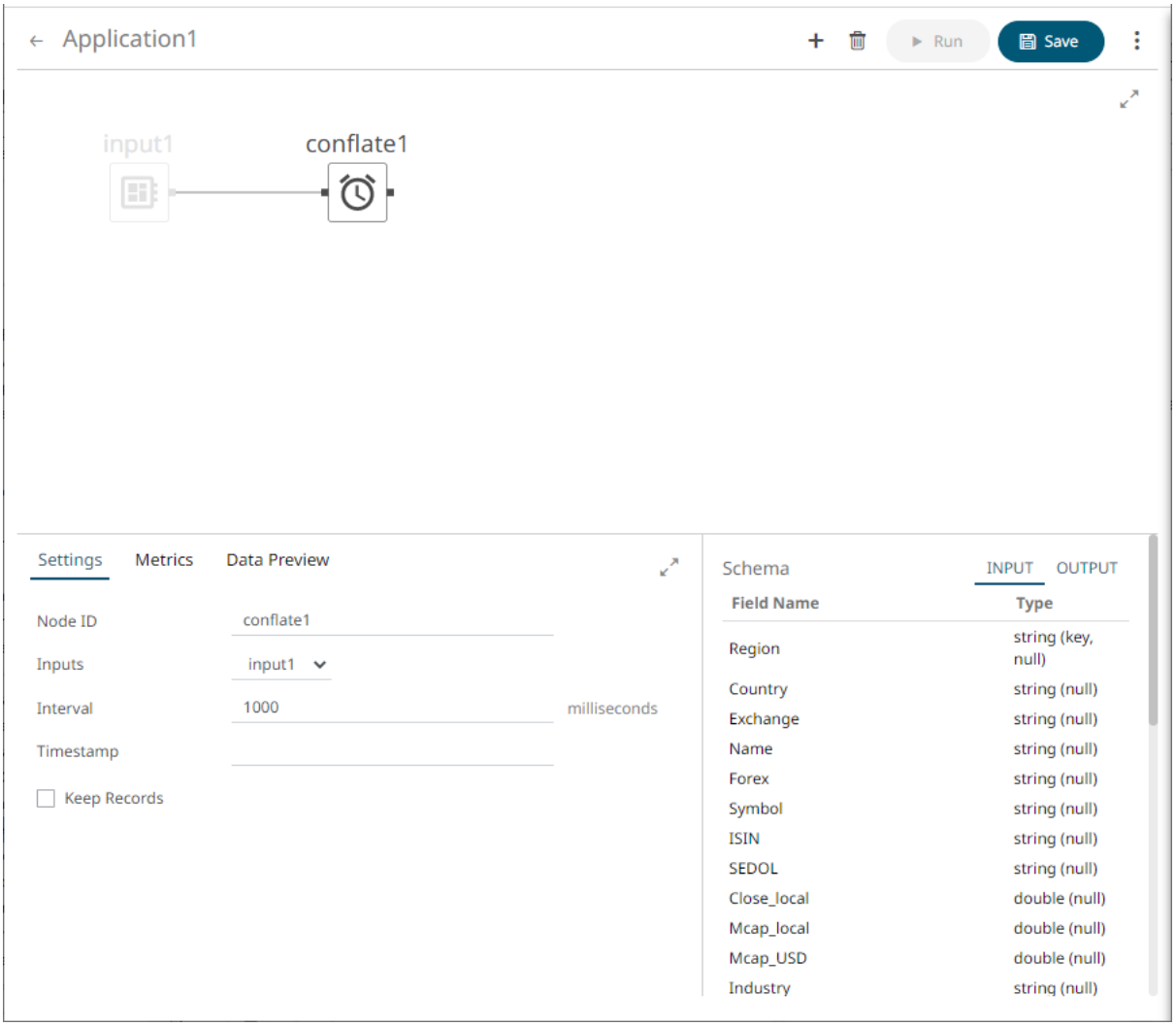
Adding a Conflate Operator

The conflate operation is used to lower the frequency of updates. The conflate will retain the last records seen on the input and push them to the output stream on a fixed interval. For example, if the input is producing a high frequency data throughput, instead of processing all of these updates, a configured conflate will only push through a small set of records on a fixed interval.

Steps:

1. In the *Application* page, click + and select **Conflate** in the *Add Operator* pane.

   The **Conflate** node icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



This operator has left (inbound) and right (outbound) edges that allow connection to other operators in the application.

2. In the *Operator Settings* panel, define or select the following properties:

Property	Description
Node ID	The ID of the conflate operator.
Inputs	Automatically connects to the currently-selected operator. You can select another ID of the operator that will be the source of the data in the *Inputs* drop-down list. The preview of the data (INPUT and OUTPUT) are displayed in the *Schema* panel.
Interval	The interval of which the data should be published to the output stream (in milliseconds).
Timestamp	The timestamp.
Keep Records	Check to retain or not remove flushed elements. This means the entire set of records will be flushed at each interval.

NOTE ***Node ID, Inputs*, and *Interval* are required properties.**

3. Save the changes.

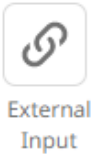
Example

```
<conflate>
    <id>Conflate </id>
    <interval>10000</interval>
</conflate>
```

Adding an External Input

Sources data directly from a Kafka topic.

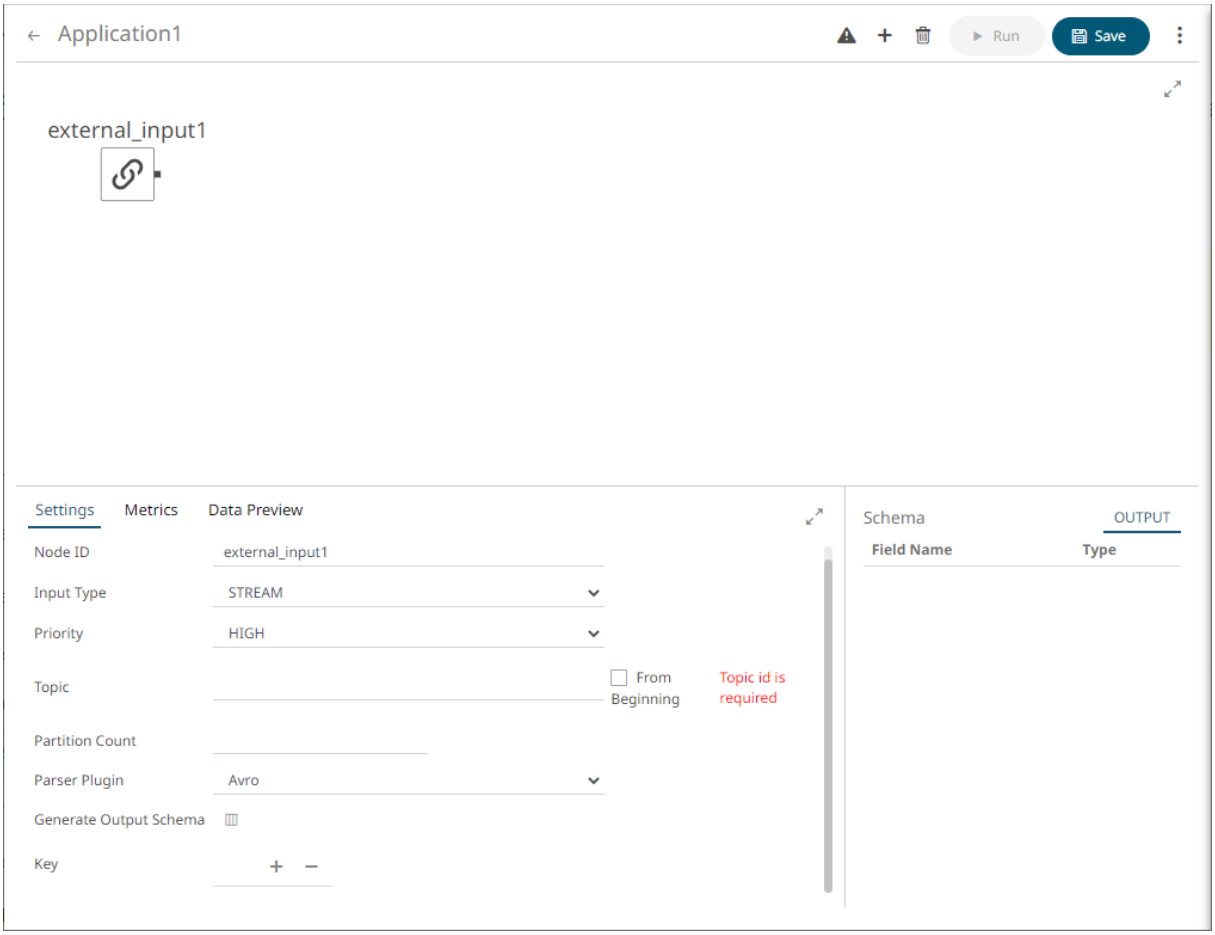
Steps:

1. In the *Application* page, click + and select **External_input** 

in the *Add Operator* pane.

   The **External Input** node

icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



This operator serves as the initial source of the data in the application. The right (outbound) edge allows you to connect to other operators.

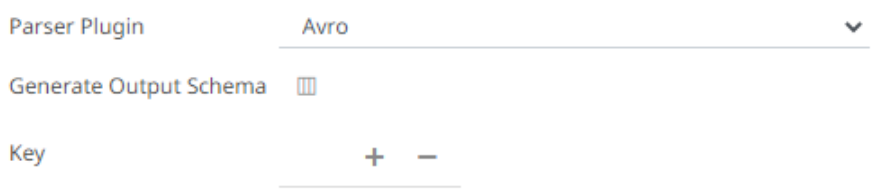
2. In the *Operator Settings* panel, define or select the following properties:

Property	Description
Node ID	The ID of the external input operator.
Input Type	Select the input type: **STREAM**, **TABLE**, or **GLOBAL_TABLE**.
Priority	Select the priority of the node's startup: • **APPLICATION** – running and successful completion of the node is critical in the application startup. • **HIGHEST** – highest priority but not critical. • **HIGH** (Default) – high priority but not critical. • **STANDARD** – standard priority. • **LOW** – low priority.
Topic	The stream of records or input you will be subscribed to.
From Beginning	Check to retrieve full history (from beginning to the latest) of the topic. If un-checked, only the latest messages after the application has started will be retrieved.

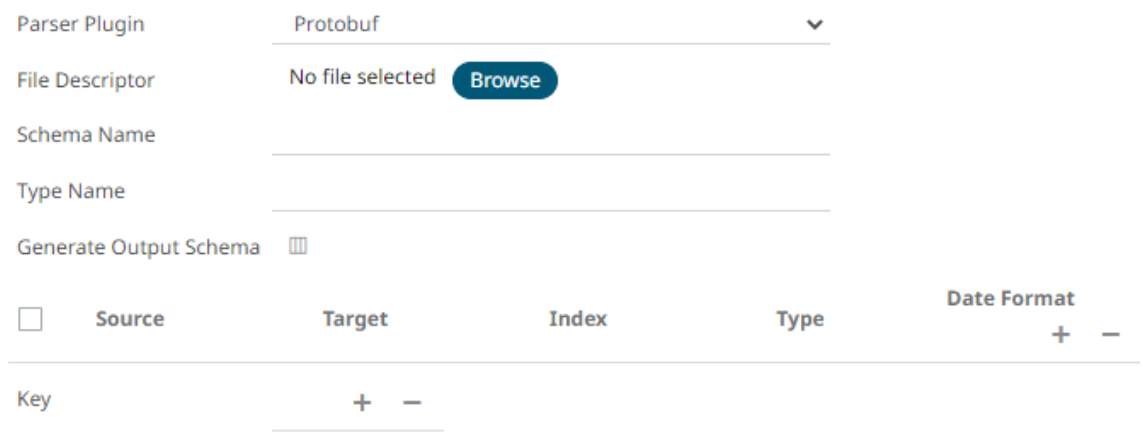
Partition Count	Enter the number of partitions for the Kafka topics that will be created for the External Input operator. Partitions allow you to parallelize a topic by splitting the data in a particular topic across multiple brokers wherein, each partition can be placed on a separate machine to allow for multiple consumers to read from a topic in parallel. **NOTE:** The External Input topic pulls the default partition count from the provided topic meta with generate schema call.

NOTE ***Node ID*, *Input Type*, and *Topic* properties are required.**

3. Select the *Parser Plugin*:
   - Avro



   - Protobuf



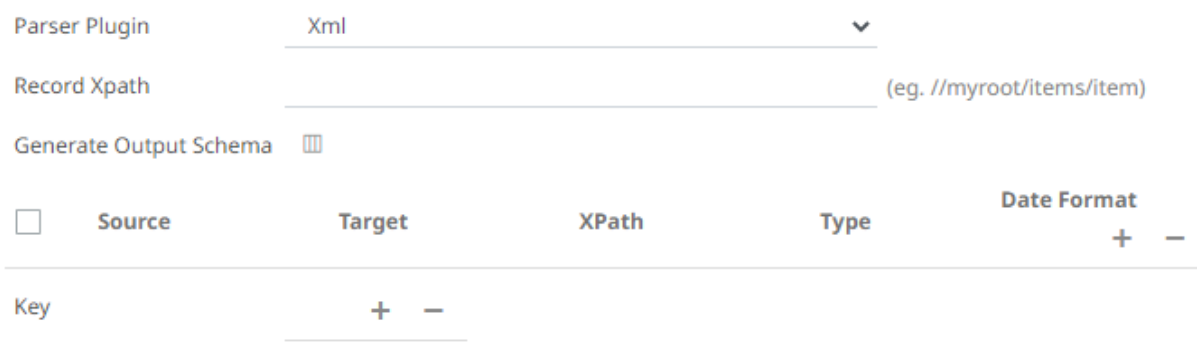
Click

to select the **File Descriptor** (`.desc` file) in the *Open* dialog.

Then enter the *Schema Name* and *Type Name*.

Property	Description
Schema Name	The Protobuf schema.
Type Name	The message of Protobuf type that will be sent to Kafka.
File Descriptor	The `FileDescriptorSet` which: • is an output of the protocol compiler. • represents a set of `.proto` files, using the `--descriptor_set_out` option.

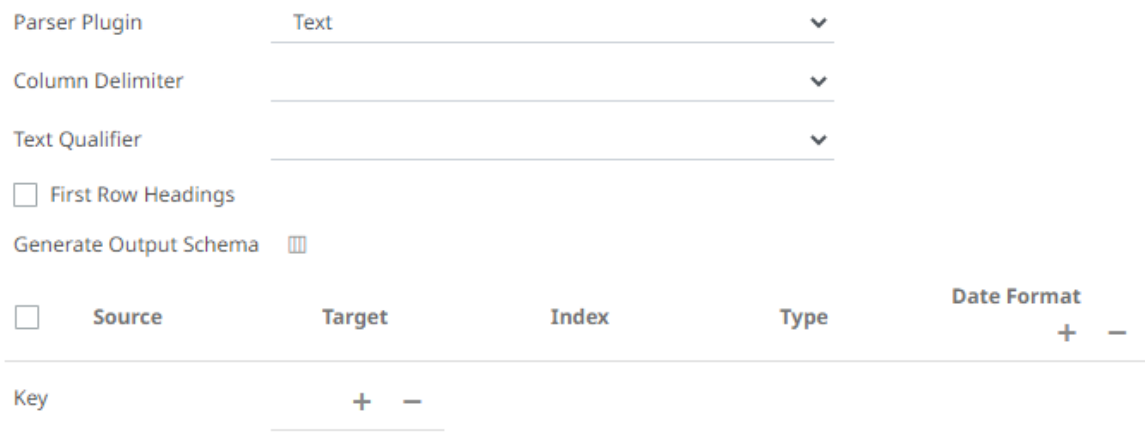
- XML



- JSON

- Text

If **Text** has been selected, confirm the **Column Delimiter and Text Qualifier**, and if the first row of the message includes column headings.



4. You can also click the following icons:

Button	Description
	Fetch the schema of the output topic. This populates the list of columns, with the data type found from inspecting the first 'n' rows of the file.
+	Add a new field entry.
−	Check a box of a field entry and click − to delete.

5. Enter or select the following properties:

Property	Description
Source	The column name of the source schema.
Target	The column name of the target schema.
XPath/Json Path/Index	The column name of the target schema.
Type	The data type of the column. Can be: BOOLEAN, DATE, DATETIME, DOUBLE, FLOAT, INT, LONG, STRING, TIME.
Date Format	Date/Time format when the data type is **DATE, DATETIME**, or **TIME**.

6. In the *Key* section, click + to add a key column from the data source schema. Repeat to add more.

   You can also delete a key column in the list by checking its box and clicking − .

7. Save the changes.

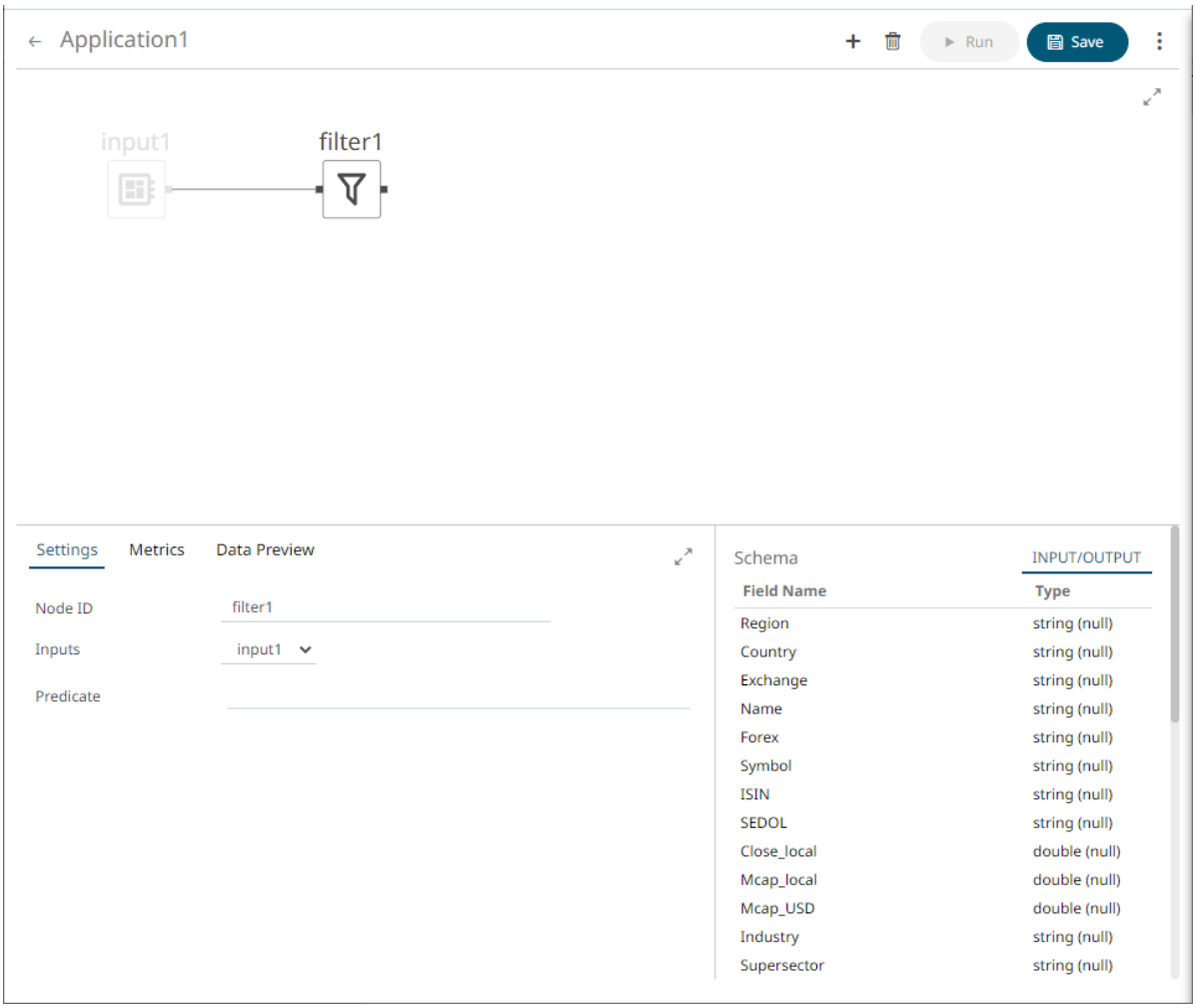
Adding a Filter Operator

Used to filter a data source based on a predicate.

Steps:

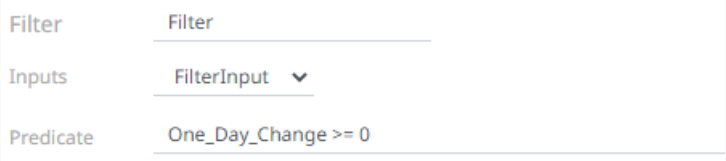
1. In the *Application* page, click + and select **Filter** in the *Add Operator* pane.

   The **Filter** node icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



   This operator has left (inbound) and right (outbound) edges that allow connection to other operators in the application.

2. In the *Operator Settings* panel, define or select the following required properties:

Property	Description
Node ID	The ID of the filter operator.
Inputs	Automatically connects to the currently-selected operator. You can select another ID of the operator that will be the source of the data in the *Inputs* drop-down list. The preview of the data (INPUT/OUTPUT) is displayed in the *Schema* panel.
Predicate	Determines whether the input record will be included or excluded. The records that will not match the predicate will be filtered out and will not be part of the output result. Example: 

3. Save the changes.

Example

```
<filter>
    <id>Filter</id>
    <!-- One_Day_Change >= 0 -->
    <predicate>One_Day_Change >= 0</predicate>
</filter>
```

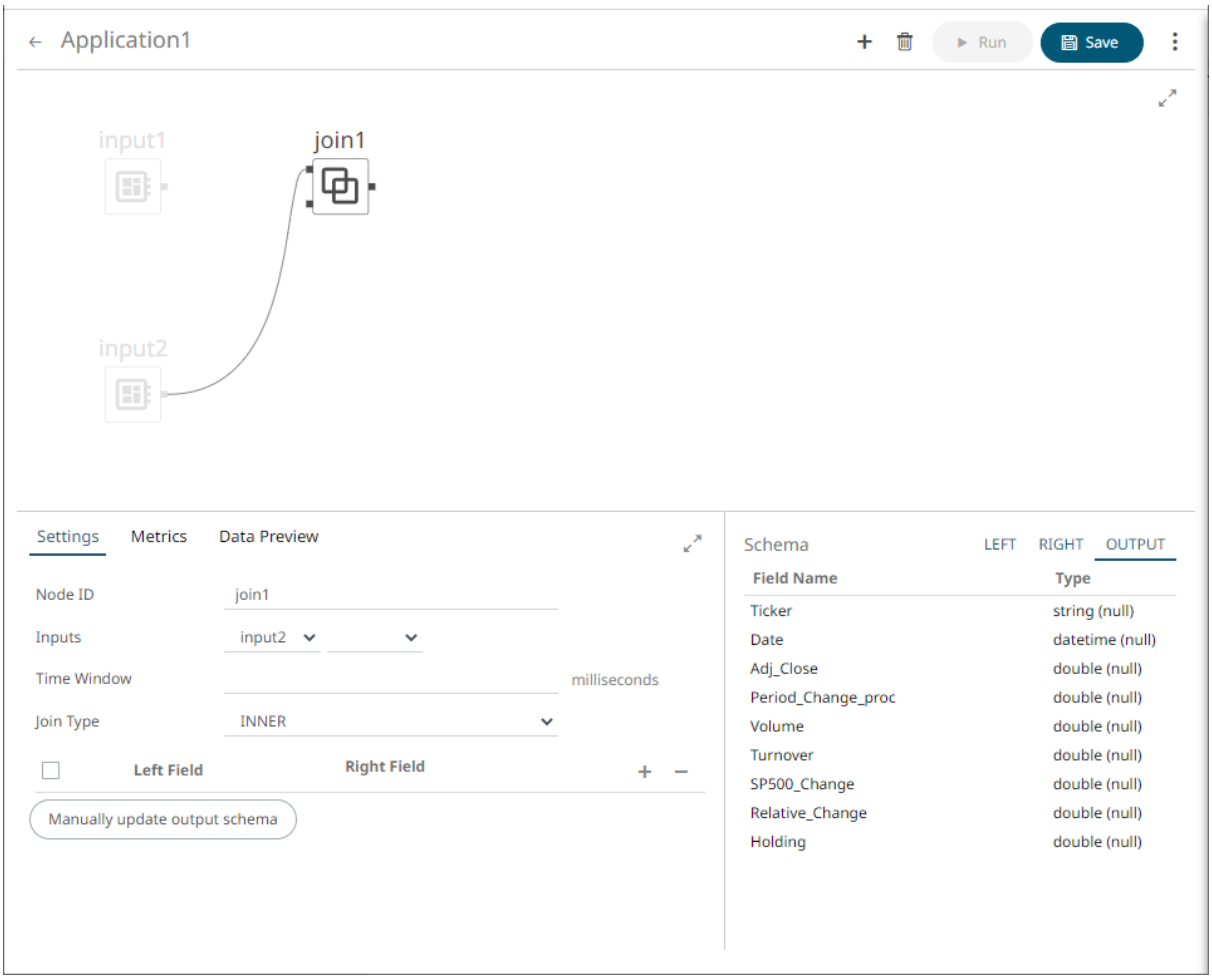
Adding a Join Operator

Used to join data sources using common keys.

Steps:

1. In the *Application* page, click + and select **Join** in the *Add Operator* pane.

   The **Join** node icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



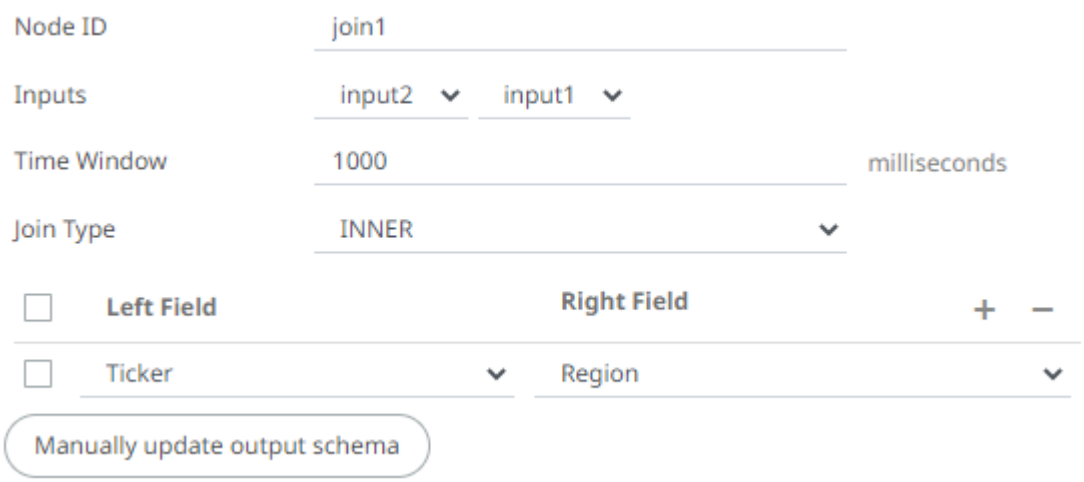
The left (inbound) edges allow you to select the input sources or operators that will be joined. The right (outbound) edge allows you to connect to other operators.

2. In the *Operator Settings* panel, define or select the following properties:

Property	Description
Node ID	The ID of the join operator.
Inputs	The left input automatically connects to the currently-selected operator. You can select another ID of the operator that will be the source of the data in the *Inputs* drop-down list. Then select the right input. The preview of the data (LEFT, RIGHT, and OUTPUT) are displayed in the *Schema* panel.
Time Window	The time window for the join operation (in milliseconds).
Join Type	The type of the join: **INNER**, **LEFT**, or **OUTER**
Left Field	The columns from the left source that will be used to join with.
Right Field	The columns from the right source that will be used to join with.

NOTE ***Node ID, Inputs, Join Type*, and *Left Field* with *Right Field* properties are required.**

3. In the *Fields List* section, click + . The key columns of the left and right sources are automatically displayed.

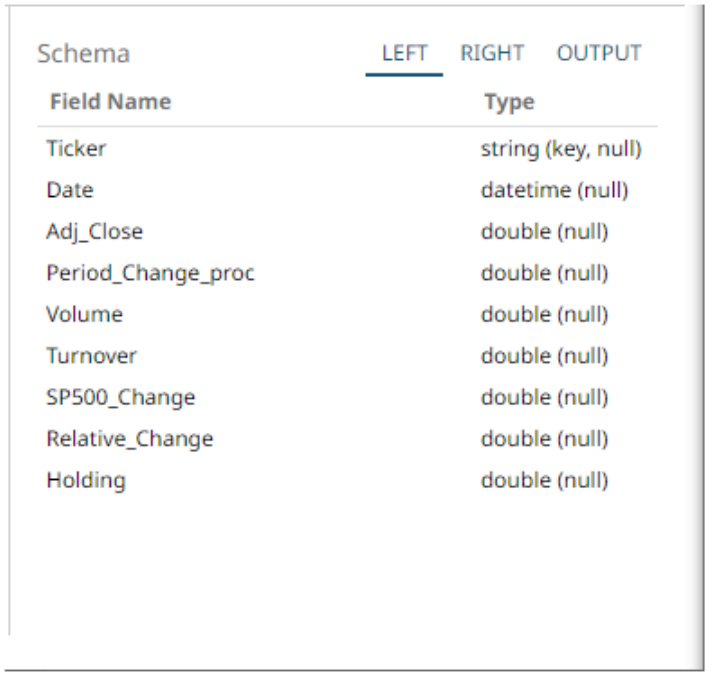


Repeat step 3 to add more columns.

You can also:

- check the topmost box to select all of the fields
- delete a field entry in the Field List by checking its box and clicking −

The LEFT, RIGHT, and OUTPUT schema are displayed.



Schema LEFT RIGHT OUTPUT

Field Name	Type
Region	string (null)
Country	string (null)
Exchange	string (null)
Name	string (null)
Forex	string (null)
Symbol	string (null)
ISIN	string (null)
SEDOL	string (null)
Close_local	double (null)
Mcap_local	double (null)
Mcap_USD	double (null)
Industry	string (null)

Schema LEFT RIGHT OUTPUT

Field Name	Type
Ticker	string (key, null)
Date	datetime (null)
Adj_Close	double (null)
Period_Change_proc	double (null)
Volume	double (null)
Turnover	double (null)
SP500_Change	double (null)
Relative_Change	double (null)
Holding	double (null)
Country	string (null)
Exchange	string (null)
Name	string (null)

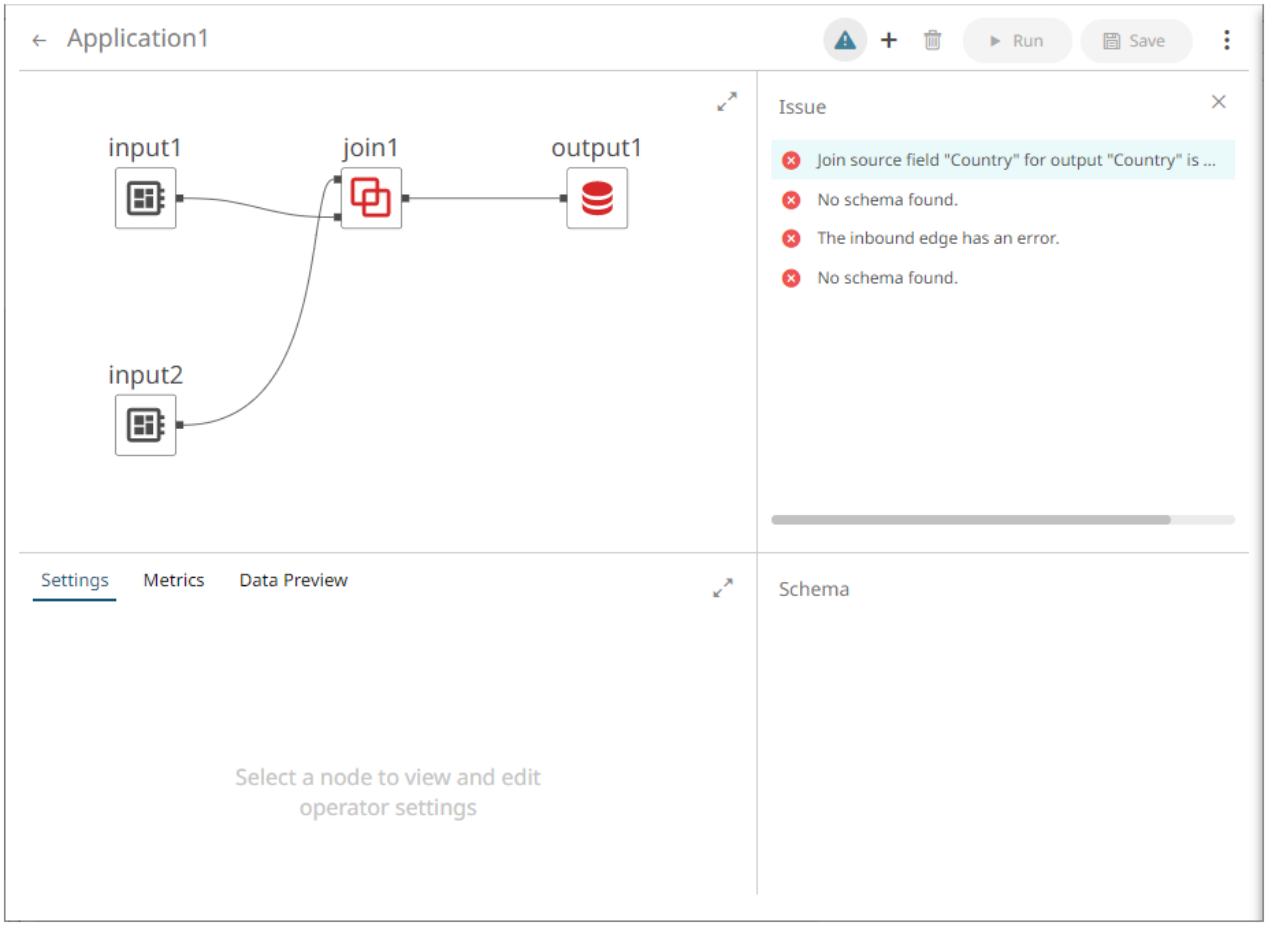
4. Save the changes.

Example

```
<join>
    <id>Join</id>
    <joinType>INNER</joinType>
    <left>
        <field>Ticker</field>
    </left>
    <right>
        <field>Ticker</field>
    </right>
    <timeWindow>1000</timeWindow>
</join>
```

Fixing Broken Joins

Changes in the input data sources may cause issues in the Join operator of an application like broken joins and output schema.



For example, if the original data source contains **Brand** and **Country** columns:

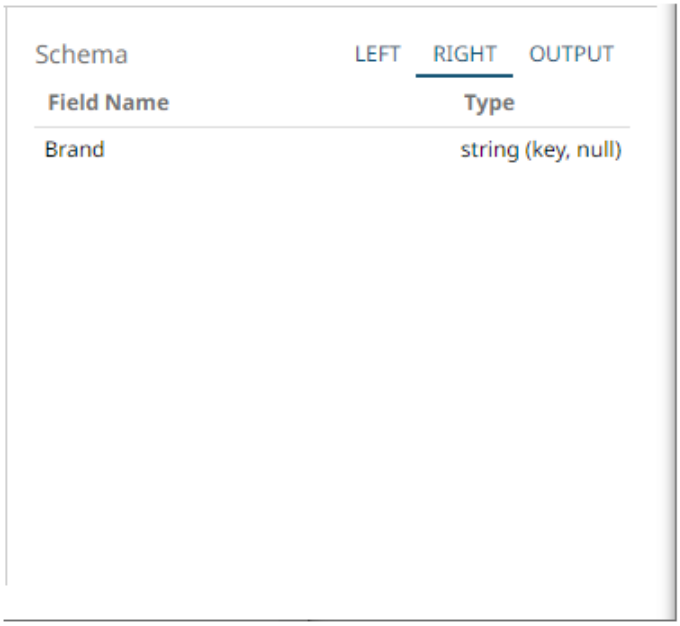
Schema LEFT RIGHT OUTPUT

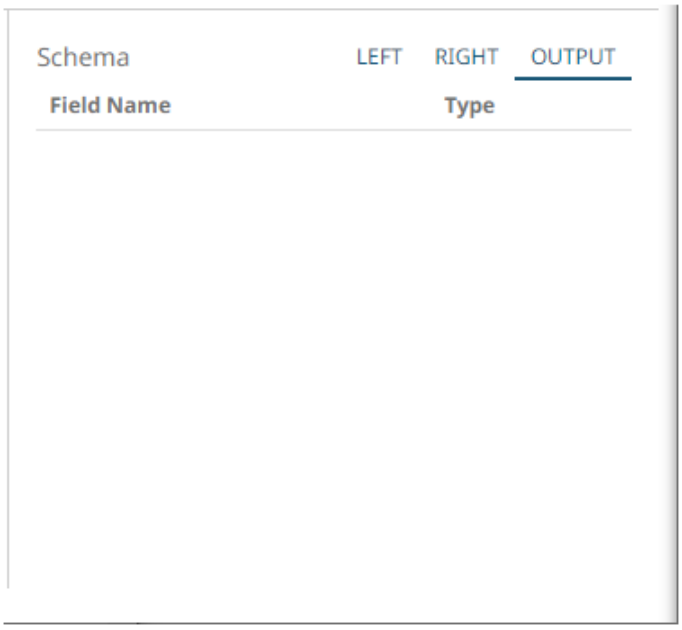
Field Name	Type
Brand	string (key, null)
Country	string (null)

Schema LEFT RIGHT OUTPUT

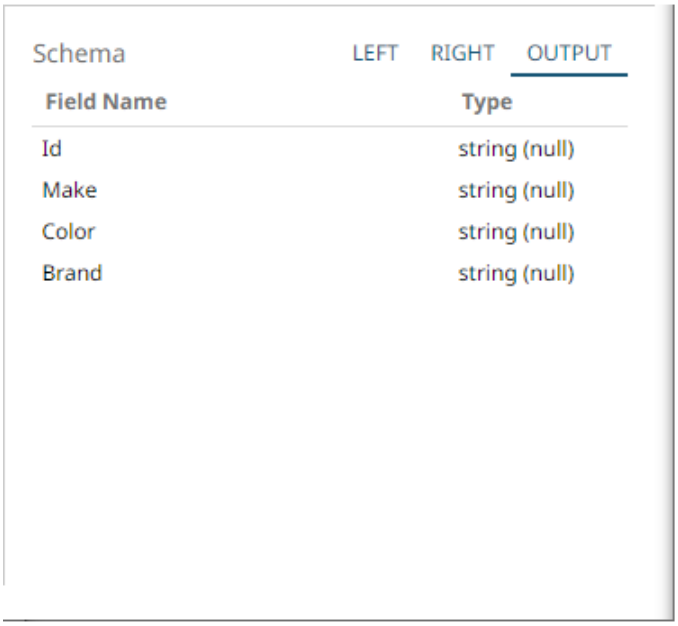
Field Name	Type
Id	string (key, null)
Make	string (null)
Color	string (null)

And if eventually the column **Country** is deleted in the data source, opening the application again will display:





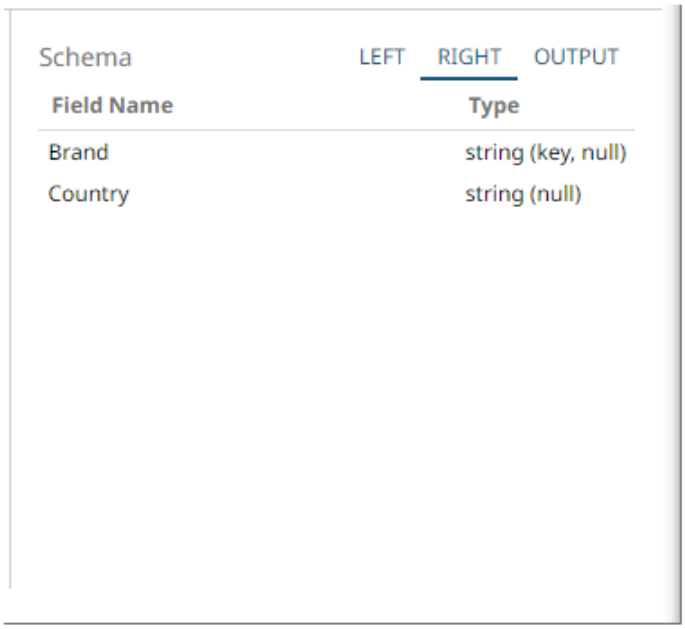
Click **Manually Update Output Schema** to fix this error. Note that Country is no longer in the list of the Output schema.

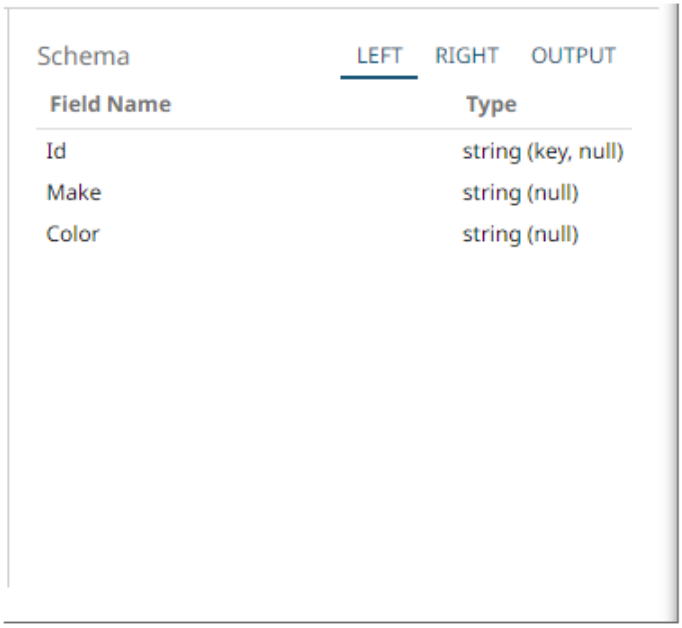


Click

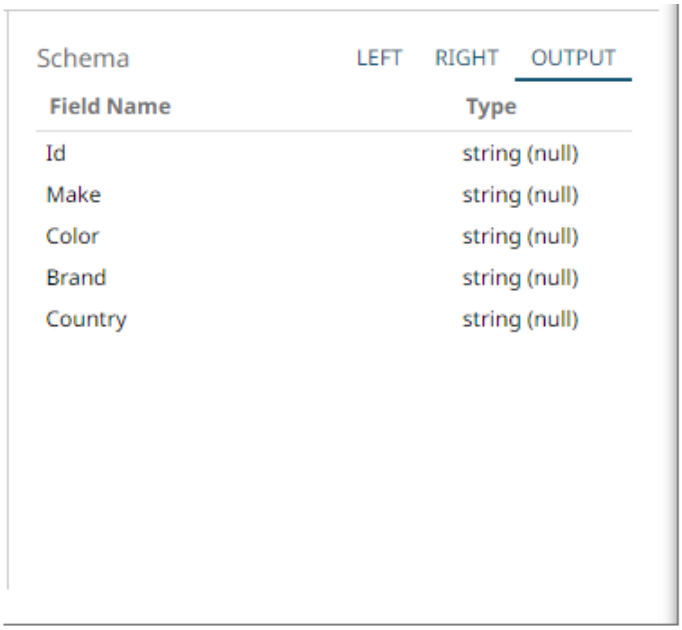
to apply the changes.

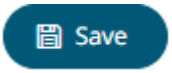
On the other hand, if new columns are added in the data source (e.g., if the **Country** column is added in the data source again), opening the application will display:





Click **Manually Update Output Schema**. Note that Country is added in the Output schema.



Click 

to apply the changes.

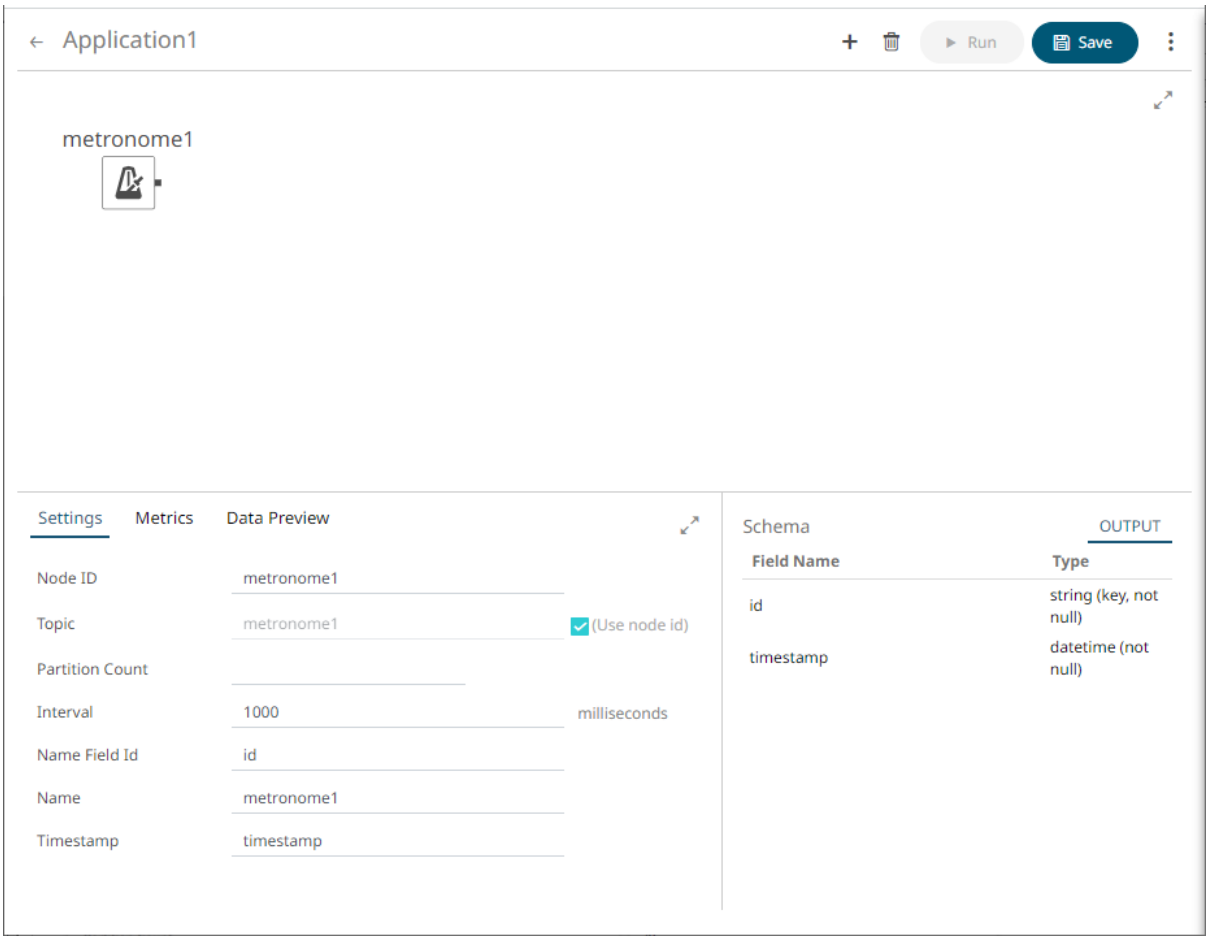
Adding a Metronome Input Operator

Similar with a synthetic input, this operator acts as a single timestamp field schema generator.

Steps:

1. In the *Application* page, click + and select **Metronome** Metronome in the *Add Operator* pane.

   The **Metronome** node metronome1 icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.

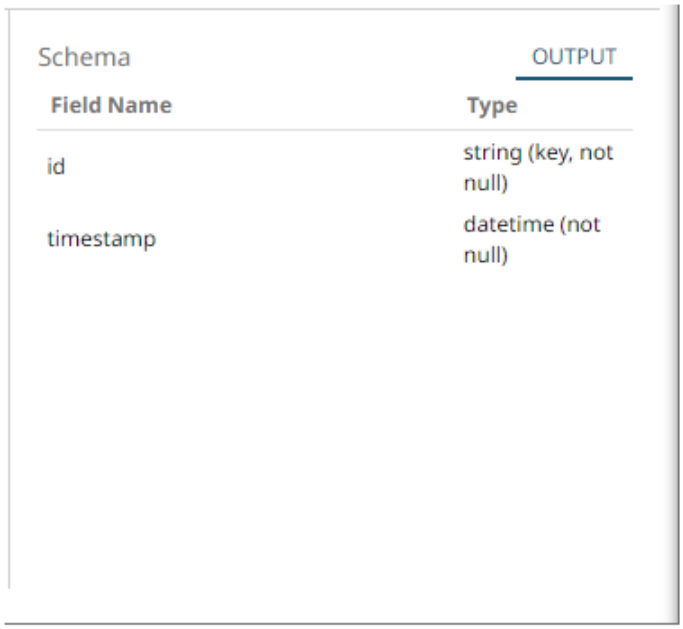


   The right (outbound) edge allows you to connect to the other operators.

2. In the *Operator Settings* panel, define or select the following required properties:

Property	Description
Node ID	The ID of the metronome operator.
Topic	The stream of records or input you will be subscribed to. Check the *Use Node ID* box to use the value entered in the *Input* ID. Otherwise, uncheck the box and enter a new *Topic* ID. When adding Topic IDs, ensure they: • must be unique across an application • must be specified • must start with a letter (a to Z) or an underscore. Also, it can only contain letters (a to Z), numbers (0 to 9), and underscores
Partition Count	Enter the number of partitions for the Kafka topics that will be created for the Metronome operator. Partitions allow you to parallelize a topic by splitting the data in a particular topic across multiple brokers wherein, each partition can be placed on a separate machine to allow for multiple consumers to read from a topic in parallel.
Interval	The interval of which the data should be published to the output stream.
Name Field Id	The ID of the name field.
Name	The name of the ID.
Timestamp	The name of the new column that will include the timestamp.

The preview of the data (OUTPUT) is displayed in the *Schema* panel.



3. Save the changes.

Example

```
<metronome>
    <id>Metronome</id>
    <topic>Metronome</topic>
    <dataProducer>
        <bufferSize>0</bufferSize>
        <id>Metronome</id>
        <keyColumns/>
        <refreshPeriod>100</refreshPeriod>
    </dataProducer>
    <inputType>STREAM</inputType>
    <interval>100</interval>
    <name>Metronome</name>
    <nameFieldId>ID</nameFieldId>
    <timestampFieldId>Timestamp</timestampFieldId>
</metronome>
```

Adding a Python Transform Operator

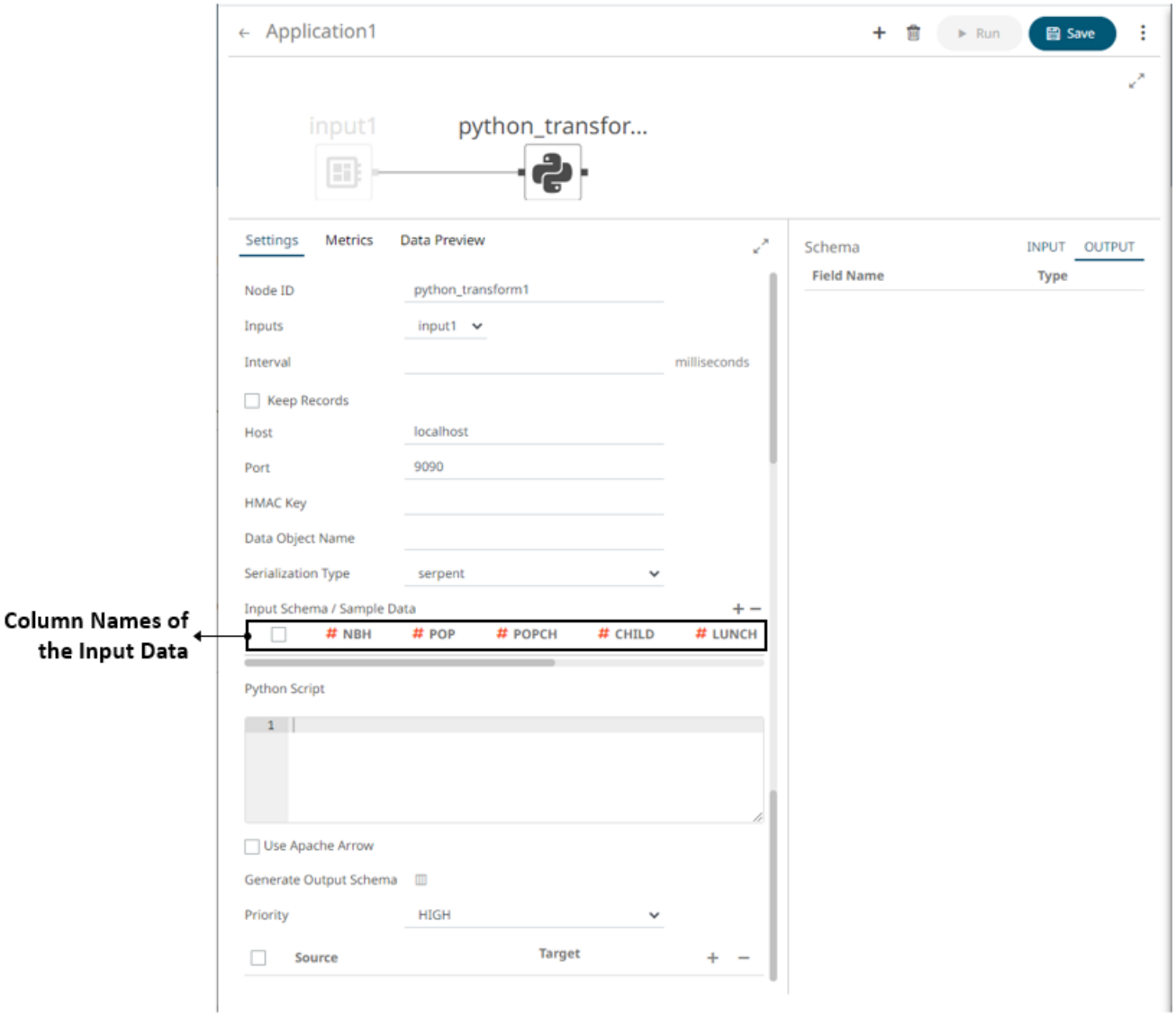
A Python script can be executed as a data transformation step in the data pipeline.

Steps:

1. In the *Application* page, click + and select **Python Transform** in the *Add Operator* pane.

   The **Python Transform** node

   icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



The right (outbound) edge allows you to connect to the other operators.

2. In the *Operator Settings* panel, define or select the following required fields:

Field	Description
Node ID	The ID of the Python Transform operator.
Inputs	The stream of records or input you will be subscribed to.
Interval	The interval of which the data should be published to the output stream (in milliseconds).
Keep Records	Check to retain or not remove flushed elements. This means the entire set of records will be flushed at each interval.
Host	Host of the Python Pyro instance.
Port	Port of the Python Pyro instance.
HMAC Key	The HMAC key that will be used to connect to the Python Pyro instance.

Data Object Name	The data structure (array of dictionaries) that Panopticon will produce, and then will be utilized by the Python Script.
Serialization Type	The serialization type: **Serpent** or **Pickle** • simple serialization library based on `ast.literal_eval` • faster serialization but less secure

NOTE **The *Host*, *Port*, *HMAC Key*, and *Serialization Type* fields will be hidden if their corresponding properties are set in the `Streams.properties` file.**

Field	Corresponding Property in Streams.properies
Host	`connector.python.host`
Port	`connector.python.port`
HMAC Key	`connector.python.password`
Serialization Type	`connector.python.serializertype`

3. Enter the required *Python Script* to execute on the active Pyro instance.
4. Check the **Use Apache Arrow** box to enable fast serialization of data frames in the Python transform.
5. On the *Input Schema/Sample Data* section, the column names of the Input data source are displayed. In cases where there are no rows from the input data source and the Python script is not handling zero rows, you can add sample data to ensure transform is applied.

   To add or manage the sample data, you can use the following icons:

Button	Description
+	Add sample data for the input column names.
−	Check a box of a sample data row and click − to delete, or check the topmost box and click − to delete all of the sample data rows.

6. In the *Generate Output Schema* section, click **Generate Output Schema** to fetch the schema of the output topic. This populates the list of columns, with the data type found from inspecting the first 'n' rows of the file.
7. Select the *Priority* of the node's startup:

Priority	Description
APPLICATION	Running and successful completion of the node is critical in the application startup.
HIGHEST	Highest priority but not critical.
HIGH (Default)	High priority but not critical.
STANDARD	Standard priority.
LOW	Low priority.

8. You can also click the following icons:

Button	Description
	Fetch the schema of the output topic. This populates the list of columns, with the data type found from inspecting the first ‘n’ rows of the file.
+	Add a new field entry.
−	Check a box of a field entry and click − to delete.

9. Save the changes.

Example

```
<operators>
    <transform>
        <id>python_transform1</id>
        <transformPlugin>Python</transformPlugin>
        <transformPluginSettings/>
        <interval>1000</interval>
        <columns>
            <type>STRING</type>
        </columns>
        <maxRowsCount>0</maxRowsCount>
    </transform>
    <input>
        <id>input1</id>
        <topic>input1</topic>
        <globalTopic>UntitledApplication_0.input1</globalTopic>
        <dataProducer>
            <id>StreamSimulator_StocksStatic</id>
            <keyColumns>
                <field>Region</field>
            </keyColumns>
            <refreshPeriod>0</refreshPeriod>
        </dataProducer>
        <inputType>STREAM</inputType>
    </input>
    <output>
        <id>output1</id>
        <topic>output1</topic>
        <globalTopic>UntitledApplication_0.output1</globalTopic>
        <dataConsumer>TextOutput</dataConsumer>
    </output>
</operators>
<streams>
    <stream>
        <source>python_transform1</source>
        <sink>
            <operator>output1</operator>
        </sink>
    </stream>
    <stream>
        <source>input1</source>
        <sink>
            <operator>python_transform1</operator>
        </sink>
    </stream>
</streams>
```

Additional Best Practice Recommendations in Using Python with Panopticon

With a Python transform or the Python connector in Panopticon, it is fairly quick and easy to enter some short code snippet and use the result. However, as a project grows, and if a solution is moved into production and becomes business critical, you need more structure in your use of Python with Panopticon:

- Code should be made into functions, even if used only in one place and even if the code content is very brief. Thereby, the operations performed by each function will be contained and you avoid the risk of naming conflicts and contamination in the global environment.
- Ensure you handle exceptions in the code you write. For example, when applying a Python transform to data, you can do an initial check in your code to see if the dataset is either a zero-row or has any rows. In which case, you want to terminate and just return the empty dataset. You should also use try-except clauses, whereby in the event of an error, you could, for example, insert the error message into the designated column in your dataset

and then return it to Panopticon. As long as there is no error, the same column could contain a plain "OK" or similar as an indicator of a no-errors result.

- Functions should ideally be turned into a package. The benefit of that is mainly about the possibility of adding unit testing and automating dependency package imports.
- Your package should have unit tests that are run when building the package.
- Your package should import any other packages that you have a dependency on.
- Developing, Testing, and Debugging the package should happen in a proper IDE, where proper debugging tools and full error messages can be monitored easily. For testing and debugging, some boiler-plate code snippets and parameter input data can be prepared, to mimic the input which could come from Panopticon parameters when the code is used via Panopticon.
- In Panopticon, the code field of the transform or connector should contain an absolute mininum of code; perhaps as little as a single function call, where the function takes the necessary arguments coming from Panopticon parameters.

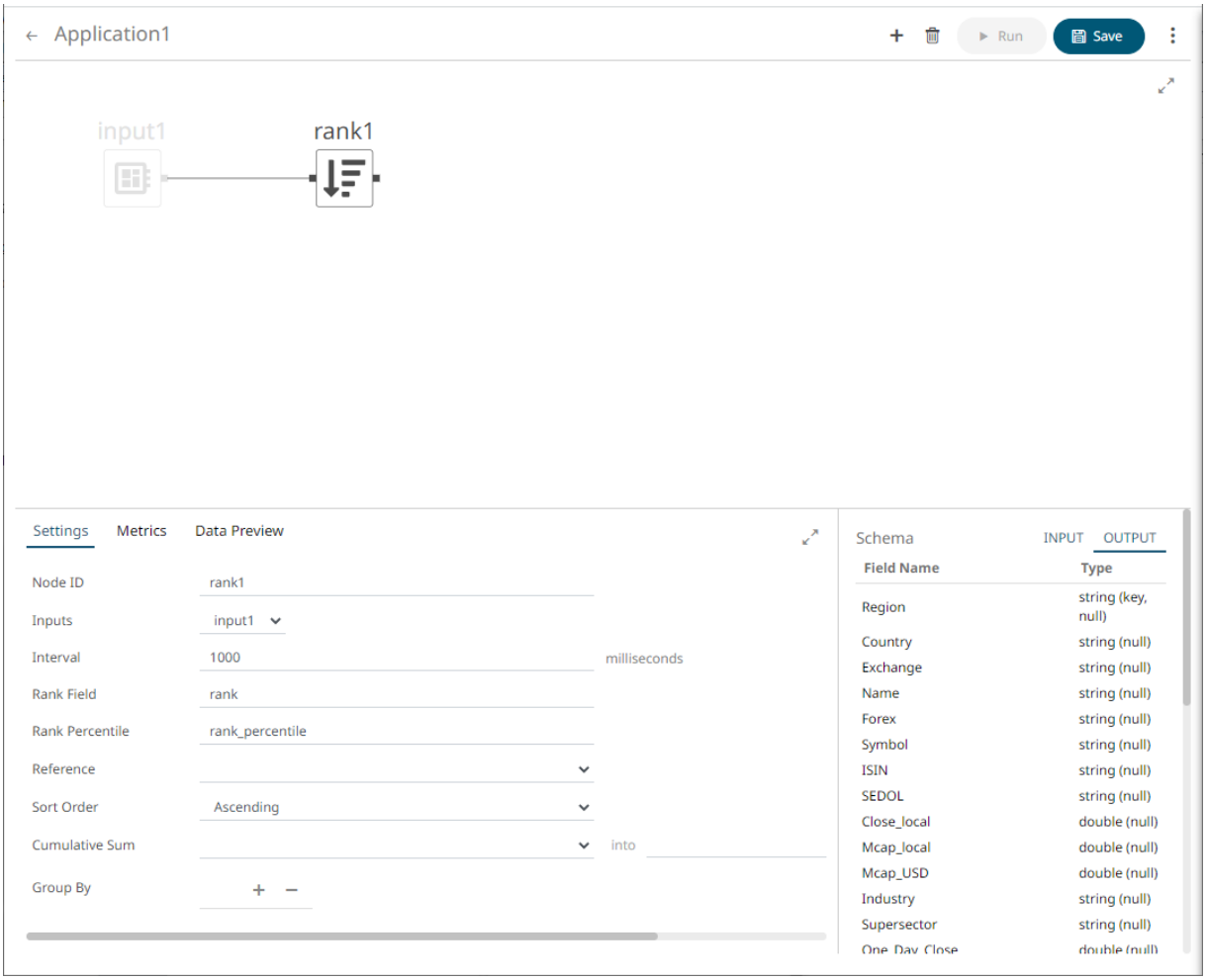
Adding a Rank Operator

Assign a rank number to records in the same group.

Steps:

1. In the *Application* page, click + and select **Rank** in the *Add Operator* pane.

   The **Rank** node icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



The right (outbound) edge allows you to connect to the other operators.

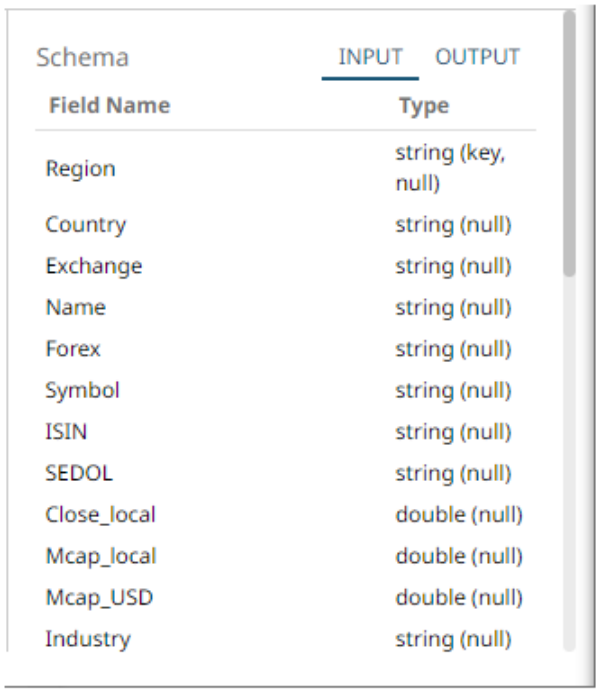
2. In the *Operator Settings* panel, define or select the following required properties:

Property	Description
Node ID	The ID of the rank operator.
Inputs	The stream of records or input you will be subscribed to.
Interval	How often the collected data should be sorted, ranked, and output (in milliseconds)
Rank Field	The name of the rank number field in the output.
Rank Percentile	The name of the percentile field in the output. This is the rank number divided by the number of records in the group.
Reference	The input field to sort records on when ranking them.
Sort Order	The order to sort the records: ASCENDING (the lowest value gets rank one) or DESCENDING (the highest value gets rank one).
Cumulative Sum	The cumulative sum based on the currently applied sort order for each Reference value. You can opt to specify a new value in the *Into* field. This column will be added in the Output schema.

Group By	The name/IDs of the fields that the data will be grouped by. Records are ranked within each group. (Proceed to step 3.)

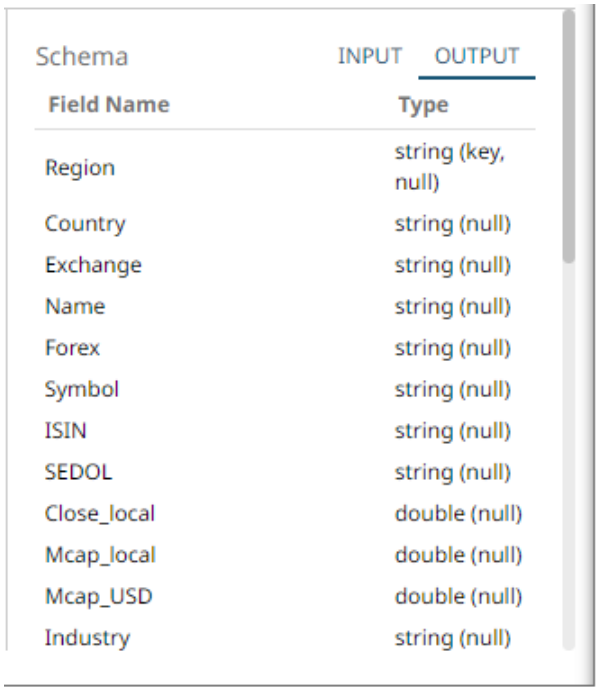
3. In the *Group By* section, click + . A column is added in the list. Click the drop-down list to select the column that will be used to group the data.

   The INPUT and OUTPUT schema are displayed.



Schema INPUT OUTPUT

Field Name	Type
Region	string (key, null)
Country	string (null)
Exchange	string (null)
Name	string (null)
Forex	string (null)
Symbol	string (null)
ISIN	string (null)
SEDOL	string (null)
Close_local	double (null)
Mcap_local	double (null)
Mcap_USD	double (null)
Industry	string (null)



Schema INPUT OUTPUT

Field Name	Type
Region	string (key, null)
Country	string (null)
Exchange	string (null)
Name	string (null)
Forex	string (null)
Symbol	string (null)
ISIN	string (null)
SEDOL	string (null)
Close_local	double (null)
Mcap_local	double (null)
Mcap_USD	double (null)
Industry	string (null)

   Repeat step 3 to add more.

Two columns are added in the Output schema: <Rank Field> and the <Rank Percentile>. For example:

One_Day_Change	double (null)
One_Day_Change_USD	double (null)
One_Week_Change	double (null)
One_Week_Change_USD	double (null)
Two_Week_Change	double (null)
Two_Week_Change_USD	double (null)
One_Month_Change	double (null)
One_Month_Change_USD	double (null)
Two_Month_Change	double (null)
Two_Month_Change_USD	double (null)
Three_Month_Change	double (null)
Three_Month_Change_USD	double (null)
RecScore	double (null)
Id	double (null)
rank	int (null)
rank_percentile	double (null)

If you set a name for the *Cumulative Sum*, it will also be added in the Output schema.

One_Day_Change_USD	double (null)
One_Week_Change	double (null)
One_Week_Change_USD	double (null)
Two_Week_Change	double (null)
Two_Week_Change_USD	double (null)
One_Month_Change	double (null)
One_Month_Change_USD	double (null)
Two_Month_Change	double (null)
Two_Month_Change_USD	double (null)
Three_Month_Change	double (null)
Three_Month_Change_USD	double (null)
RecScore	double (null)
Id	double (null)
rank	int (null)
rank_percentile	double (null)
CumulativeSumSize	double (null)

You can also delete a column in the *Group By* list by checking its box and clicking .

4. Save the changes.

Example

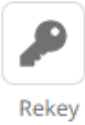
```
<rank>
    <id>rank1</id>
    <interval>1000</interval>
    <groupBy>
        <field>Region</field>
        <field>Country</field>
    </groupBy>
    <rankField>rank</rankField>
    <rankPercentileField>rank_percentile</rankPercentileField>
    <reference>Mcap_USD</reference>
    <sortOrder>ASCENDING</sortOrder>
    <cumSumSourceField>One_Week_Change</cumSumSourceField>
</rank>
```

Adding a Rekey Operator

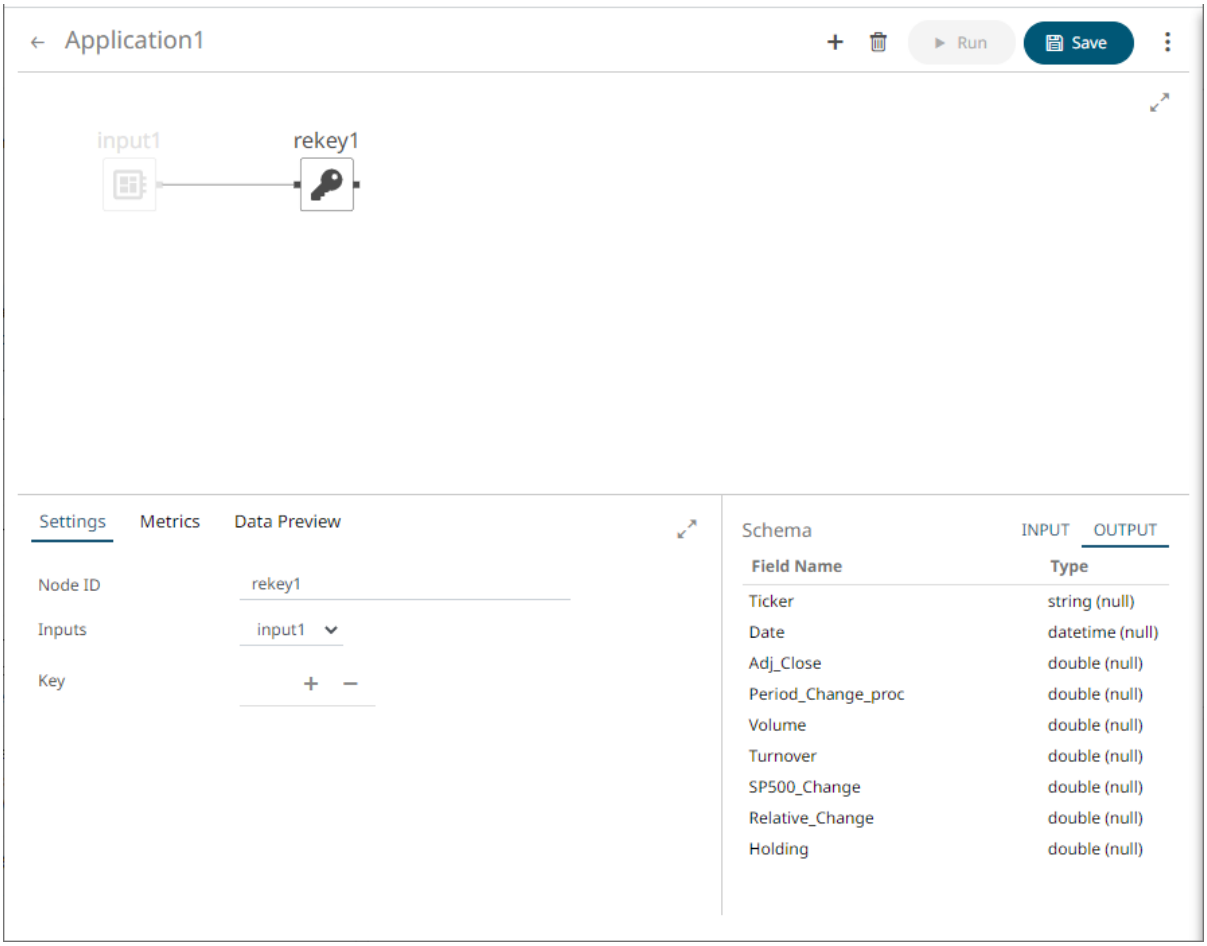
Takes a stream data and changes its key. The new key can be any subset of fields from the stream.

> **NOTE** **This operator can never be applied to a table since tables require keys to be unique and therefore, you need to specify how multiple records with the same key should be aggregated.**

Steps:

1. In the *Application* page, click + and select **Rekey**  in the *Add Operator* pane.

   The **Rekey** node icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



The right (outbound) edge allows you to connect to the other operators.

2. In the *Operator Settings* panel, define or select the following required properties:

Property	Description
Node ID	The ID of the rekey operator.
Inputs	The stream of records or input you will be subscribed to.
Key	The key column. Proceed to step 3.

NOTE ***Node ID, Inputs, and Key*** **properties are required.**

3. In the *Key* section, click + to select the new key column in the drop-down list box from the data source schema. Repeat to add more.

   You can also delete a key column in the list by checking its box and clicking − .

   The preview of the data (OUTPUT) is displayed in the *Schema* panel.

Schema INPUT OUTPUT

Field Name	Type
Ticker	string (null)
Date	datetime (null)
Adj_Close	double (null)
Period_Change_proc	double (null)
Volume	double (null)
Turnover	double (null)
SP500_Change	double (null)
Relative_Change	double (null)
Holding	double (null)

4. Save the changes.

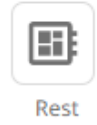
Example

```
<rekey>
     <id>rekey1</id>
     <key>
         <field>Ticker</field>
         <field>Volume</field>
     </key>
</rekey>
```

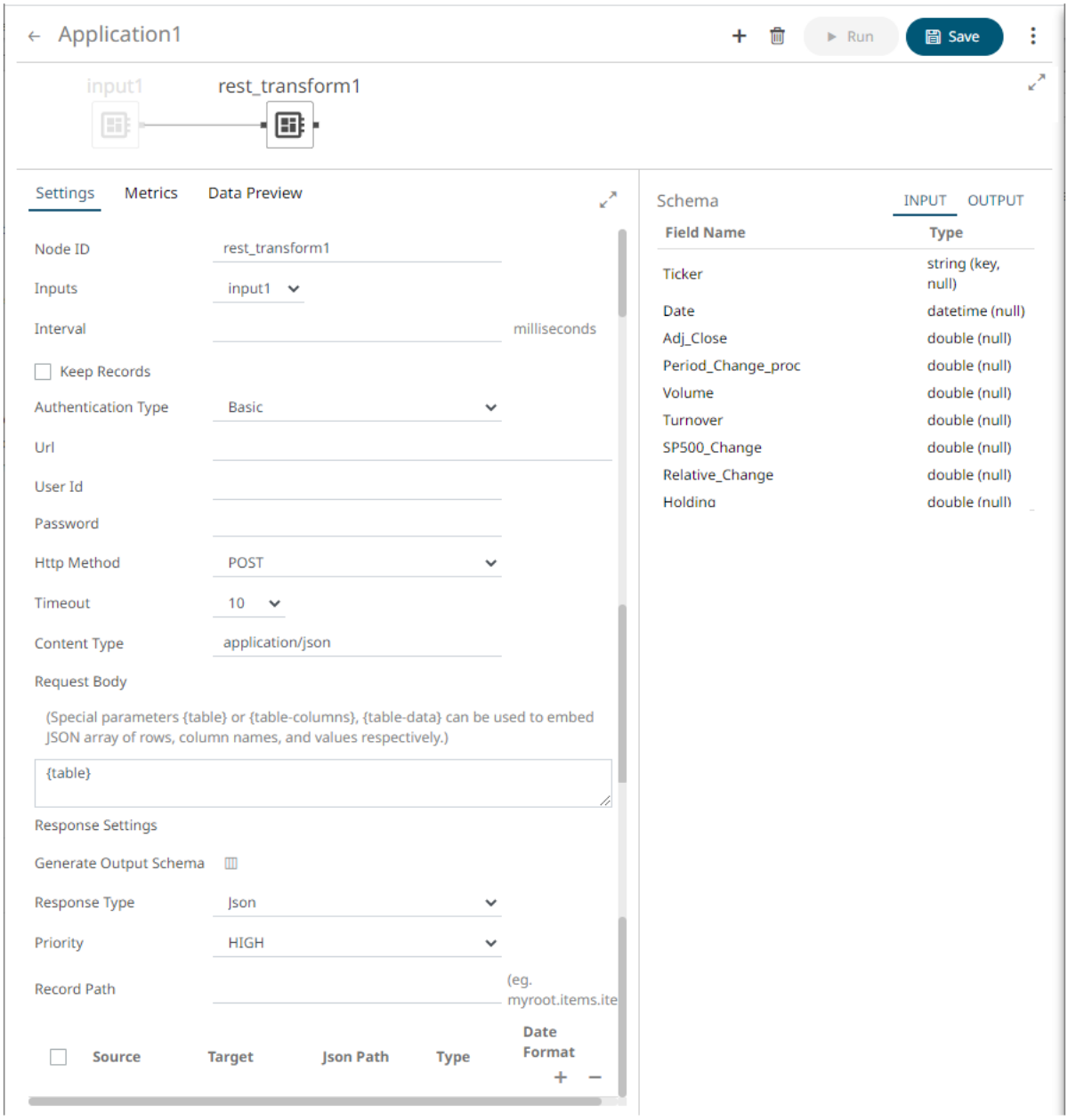
Adding a REST Transform Operator

Takes an input data frame, executes a REST call, and interprets the result which gets passed upstream.

Steps:

1. In the *Application* page, click + and select **Rest Transform**  in the *Add Operator* pane.

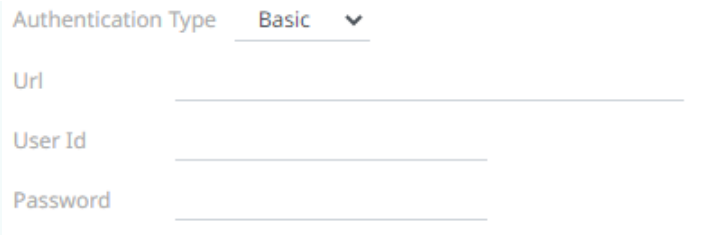
   The **REST Transform** node rest_transform1 icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



The right (outbound) edge allows you to connect to the other operators.

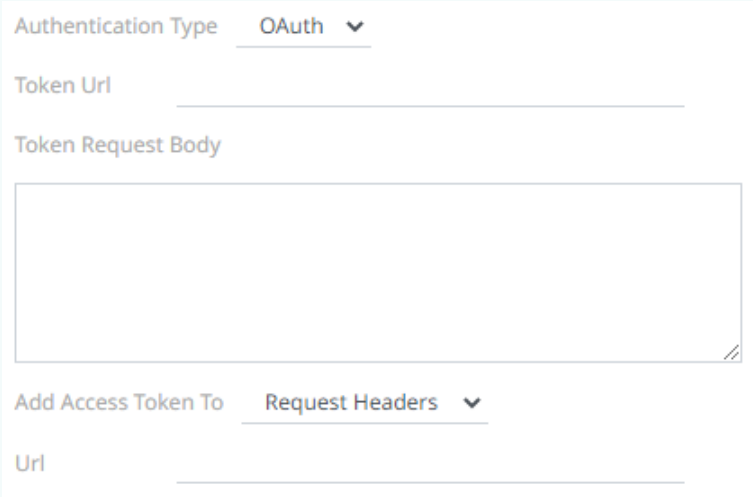
5. In the *Operator Settings* panel, define or select the following required properties:

Property	Description
Node ID	The ID of the REST Transform operator.
Inputs	The stream of records or input you will be subscribed to.
Interval	The interval of which the data should be published to the output stream (in milliseconds).
Keep Records	Check to retain or not remove flushed elements. This means the entire set of records will be flushed at each interval.
Authentication Type	• **Basic**



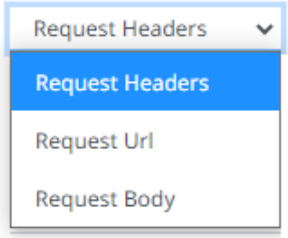
Enter the *URL* of the REST API. Then enter the *User Id* and the *Password* that will be used to the connect to the REST API.

- **OAuth**



Then enter the following settings:

- **Token URL** – The URL to retrieve the access token from.
- **Token Request Body** – The request body used for access token requests.
- **Add Access Token To** - The Access token retrieved from the *Token URL* can be added to headers, URL or request body, depending on how the REST endpoint needs the token.



  - Request Header - A header is automatically added to the REST API request.
  - Request URL - The URL needs to be manually parameterised with a {access_token} parameter, before calling the REST API, the parameter is replaced with the actual token.
  - Request Body - The Request Body needs to be manually parameterised with a {access_token} parameter, before calling the REST API, the parameter is replaced with the actual token.
- URL – The URL of the REST API.

HTTP Method	Select the appropriate HTTP method for the request from the following options:

	 • GET – retrieve data • POST – add new data • PUT – replace existing data • DELETE – remove existing data
Timeout	The length of time to wait for the server response (10 to 300). Default is **10**.
Content Type	The required Content Type. Default is **application/json.**
Request Body	The Request Body for HTTP POST. You can also click to generate the Request Body.

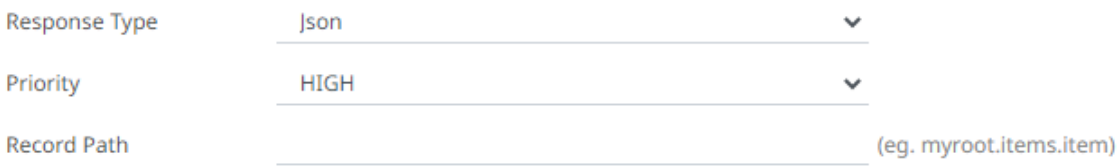
6. Check the **Multiple Records** box to send unnamed array.

7. In the *Response Settings* section, click **Generate Output Schema** to fetch the schema of the output topic. This populates the list of columns, with the data type found from inspecting the first 'n' rows of the file.

8. Select the *Response Type*:

   - XML
   - JSON

     If **JSON** is selected, enter the *Record Path* which allows the identification of multiple records within the JSON document.

   - Text

     If **Text** is selected, confirm the **Column Delimiter**, **Text Qualifier**, and if the first row of the message includes column headings.

Response Type	Text
Priority	HIGH
Column Delimiter	Comma {,}
Text Qualifier	<none>
First Row Headings	☑

8. Select the *Priority* of the node's startup:

Priority	Description
APPLICATION	Running and successful completion of the node is critical in the application startup.
HIGHEST	Highest priority but not critical.
HIGH (Default)	High priority but not critical.
STANDARD	Standard priority.
LOW	Low priority.

9. Enter or select the following properties:

Property	Description
Source	The column name of the source schema.
Target	The column name of the target schema.
XPath/Json Path/Index	The column name of the target schema.
Type	The data type of the column. Can be: BOOLEAN, DATE, DATETIME, DOUBLE, FLOAT, INT, LONG, STRING, TIME.
Date Format	Date/Time format when the data type is **DATE, DATETIME**, or **TIME**.

9. You can also click the following icons:

Button	Description
+	Add a new field entry.
−	Check a box of a field entry and click − to delete.

10. Save the changes.

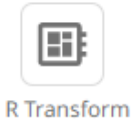
Adding an R Transform Operator

An R script can be executed as a data transformation step in the data pipeline. Specifically:

- Data is retrieved from an underlying source.
- The returned data table is translated into an R data frame.
- The R data frame and supplied R Script are passed to an external R process running Rserve.
- The external Rserve process returns a resulting R data frame.
- The returned data frame is translated into a Panopticon table for visualization rendering.

For this to occur, both R and Rserve must be installed, and initialized.

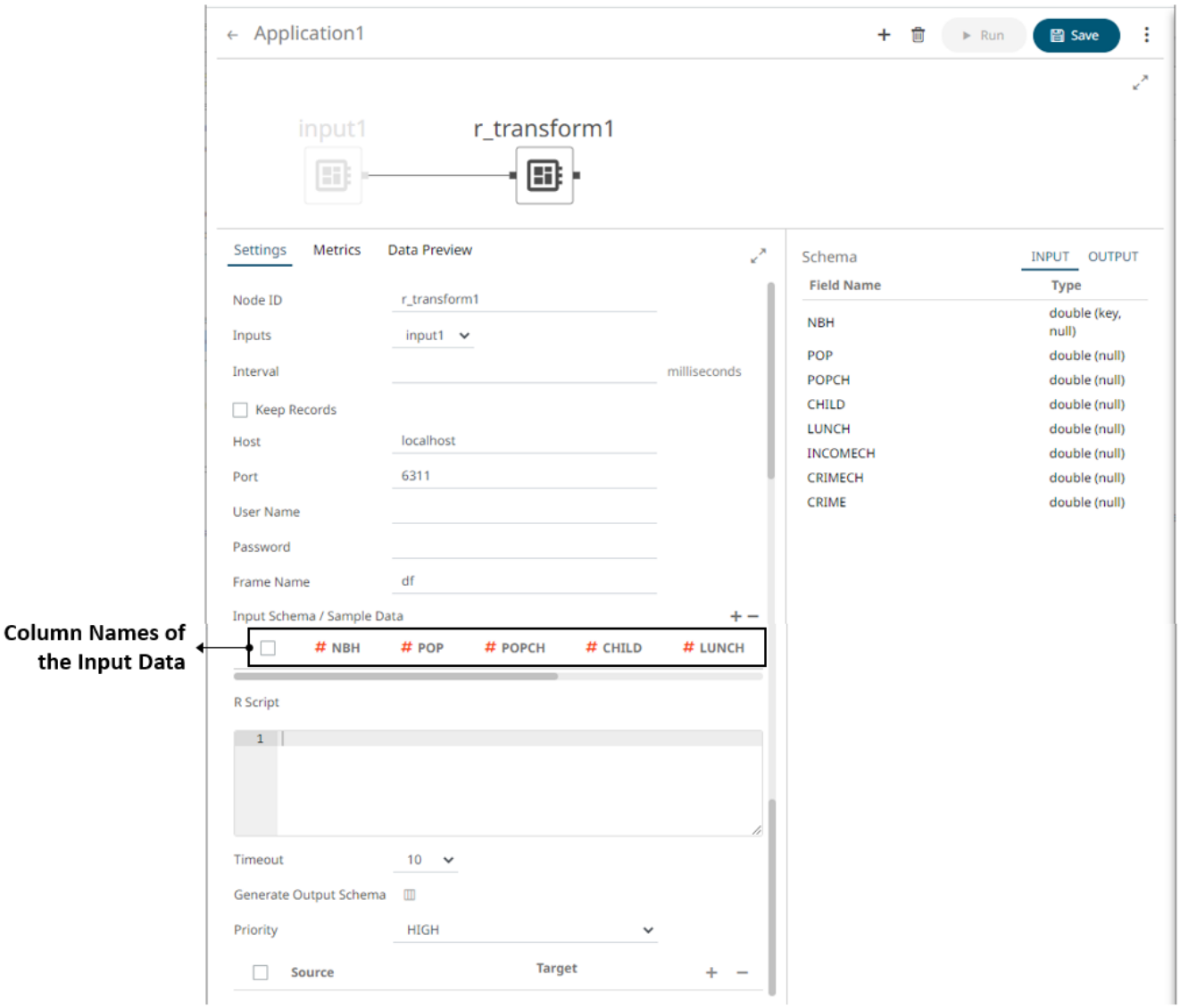
Steps:

1. In the *Application* page, click + and select **R Transform** 

in the *Add Operator* pane.

   The **R Transform** node

icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



The right (outbound) edge allows you to connect to the other operators.

2. In the *Operator Settings* panel, define or select the following required properties:

Property	Description
Node ID	The ID of the R Transform operator.
Inputs	The stream of records or input you will be subscribed to.
Interval	The interval of which the data should be published to the output stream (in milliseconds).
Keep Records	Check to retain or not to remove flushed elements. This means the entire set of records will be flushed at each interval.
Host	Host of the Rserve instance.
Port	Port of the Rserve instance.
User Name	The user Id if authentication is enabled on the Rserve process.
Password	The password if authentication is enabled on the Rserve process.
Frame Name	The *Frame Name* that Panopticon Streams will produce that will be utilized by the R scripts. The default is **df**.
R Script	The R script that reference the input frame name. Returns a data frame.
Timeout	The length of time to wait for the server response (10 to 300). Default is **10**.

3. On the *Input Schema/Sample Data* section, the column names of the Input data source are displayed. In cases where there are no rows from the input data source and the R script is not handling zero rows, you can add sample data to ensure transform is applied.

   To add or manage the sample data, you can use the following icons:

Button	Description
+	Add sample data for the input column names.
−	Check a box of a sample data row and click − to delete, or check the topmost box and click − to delete all of the sample data rows.

4. In the *Generate Output Schema* section, click **Generate Output Schema** to fetch the schema of the output topic. This populates the list of columns, with the data type found from inspecting the first 'n' rows of the file.

5. Select the *Priority* of the node's startup:

Priority	Description
APPLICATION	Running and successful completion of the node is critical in the application startup.
HIGHEST	Highest priority but not critical.
HIGH (Default)	High priority but not critical.
STANDARD	Standard priority.
LOW	Low priority.

6. You can also opt to click the following icons:

Button	Description
+	Add a new field entry then enter/select the following properties: • Source – the column of the source schema. • Target – the column name of the target schema. • Type - The data type of the column. Can be: **BOOLEAN**, **DATE**, **DATETIME**, **DOUBLE**, **FLOAT**, **INT**, **LONG**, **STRING**, **TIME**.
−	Check a box of a field entry and click − to delete.

7. Continue adding the necessary operators then save the changes in the application.

Additional Best Practice Recommendations in Using R with Panopticon

With an R transform or the Rserve connector in Panopticon, it is fairly quick and easy to enter some short code snippet and use the result. However, as a project grows, and if a solution is moved into production and becomes business critical, you need more structure in your use of R and Rserve with Panopticon:

- Code should be made into functions, even if used only in one place and even if the code content is very brief. Thereby, the operations performed by each function will be contained and you avoid the risk of naming conflicts and contamination in the global environment.
- Ensure you handle exceptions in the code you write. For example, when applying an R transform to data, you can do an initial check in your code to see if the dataset is either zero-row or has any rows. In which case, you want to terminate and just return the empty dataset. You should also use tryCatch clauses, whereby in the event of an error or a warning, you could, for example, insert the error/warning message into the designated column in your dataset and then return it to Panopticon. As long as there is no error, the same column could contain a plain "OK" or similar as an indicator of a no-errors result.
- Functions should ideally be turned into a package. The benefit of that is mainly about the possibility of adding unit testing and automating dependency package imports.
- Your package should have unit tests that are run when building the package.
- Your package should import any other packages that you have a dependency on.
- Developing, Testing and Debugging the package should happen in a proper IDE, where proper debugging tools and full error messages can be monitored easily. For testing and debugging, some boiler-plate code snippets and parameter input data can be prepared, to mimic the input which could come from Panopticon parameters when the code is used via Panopticon.
- In Panopticon, the code field of the transform or connector should contain an absolute mininum of code; perhaps as little as a single function call, where the function takes the necessary arguments coming from Panopticon parameters.
- With R and Rserve, it should be configured to load (import) your packages on startup, which will avoid the overhead of repeated loading of the packages upon each call.

Adding a Scatter Operator

Given a record with array fields (must have the same length), the scatter operator will emit one record for each position in the array(s). This operator is similar with unpivot but on array positions instead of columns.

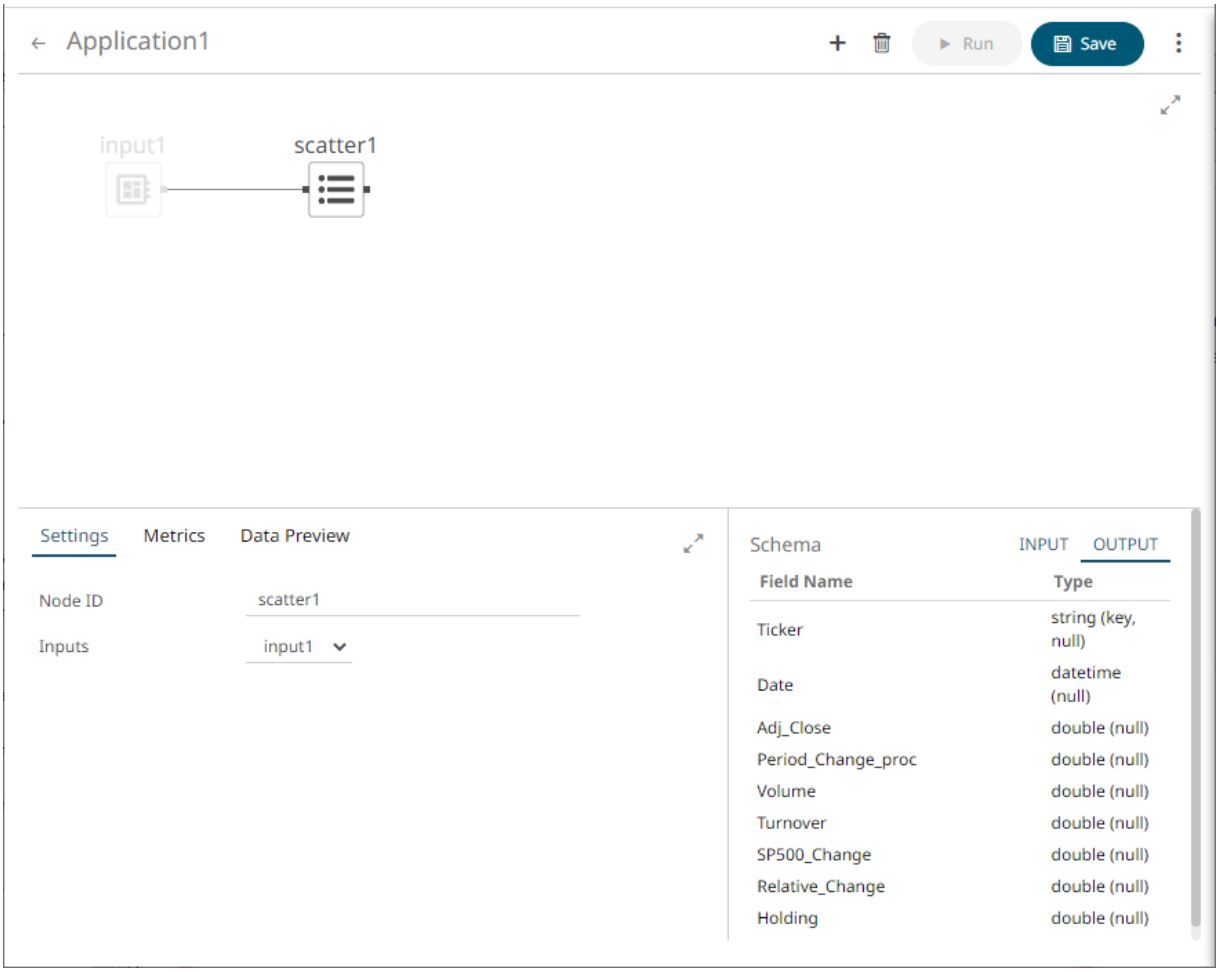
If the input record has an integer array field A of length N and text (non-array) field T, the operator will output N records with integer (non-array) field A and text (non-array) field T. For example, they will have values: `{ A[0], T }, { A[1], T }, …, { A[N-1], T }` (assuming zero-based indexing).

If the input has no array fields, the scatter operator is a no-op and will pass records through unchanged.

Steps:

1. In the *Application* page, click + and select **Scatter** in the *Add Operator* pane.

   The **Scatter** node icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



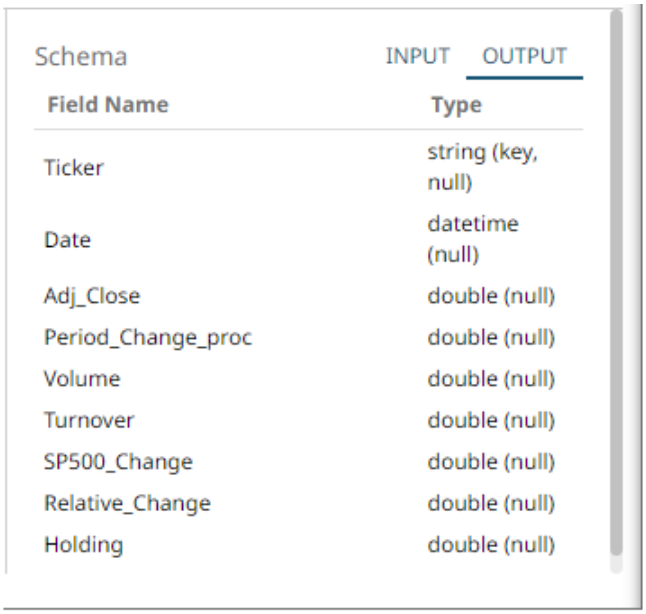
   The right (outbound) edge allows you to connect to the other operators.

2. In the *Operator Settings* panel, define or select the following required properties:

Property	Description
Node ID	The ID of the scatter operator.
Inputs	The stream of records or input you will be subscribed to.

NOTE ***Node ID and Inputs*** **properties are required.**

The preview of the data (OUTPUT) is displayed in the *Schema* panel.



Schema INPUT OUTPUT

Field Name	Type
Ticker	string (key, null)
Date	datetime (null)
Adj_Close	double (null)
Period_Change_proc	double (null)
Volume	double (null)
Turnover	double (null)
SP500_Change	double (null)
Relative_Change	double (null)
Holding	double (null)

3. Save the changes.

Example

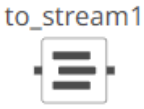
```
<scatter>
            <id>scatter1</id>
        </scatter>
    </operators>
    <streams>
        <stream>
            <source>input1</source>
            <sink>
                <operator>scatter1</operator>
            </sink>
    </stream>
</streams>
```

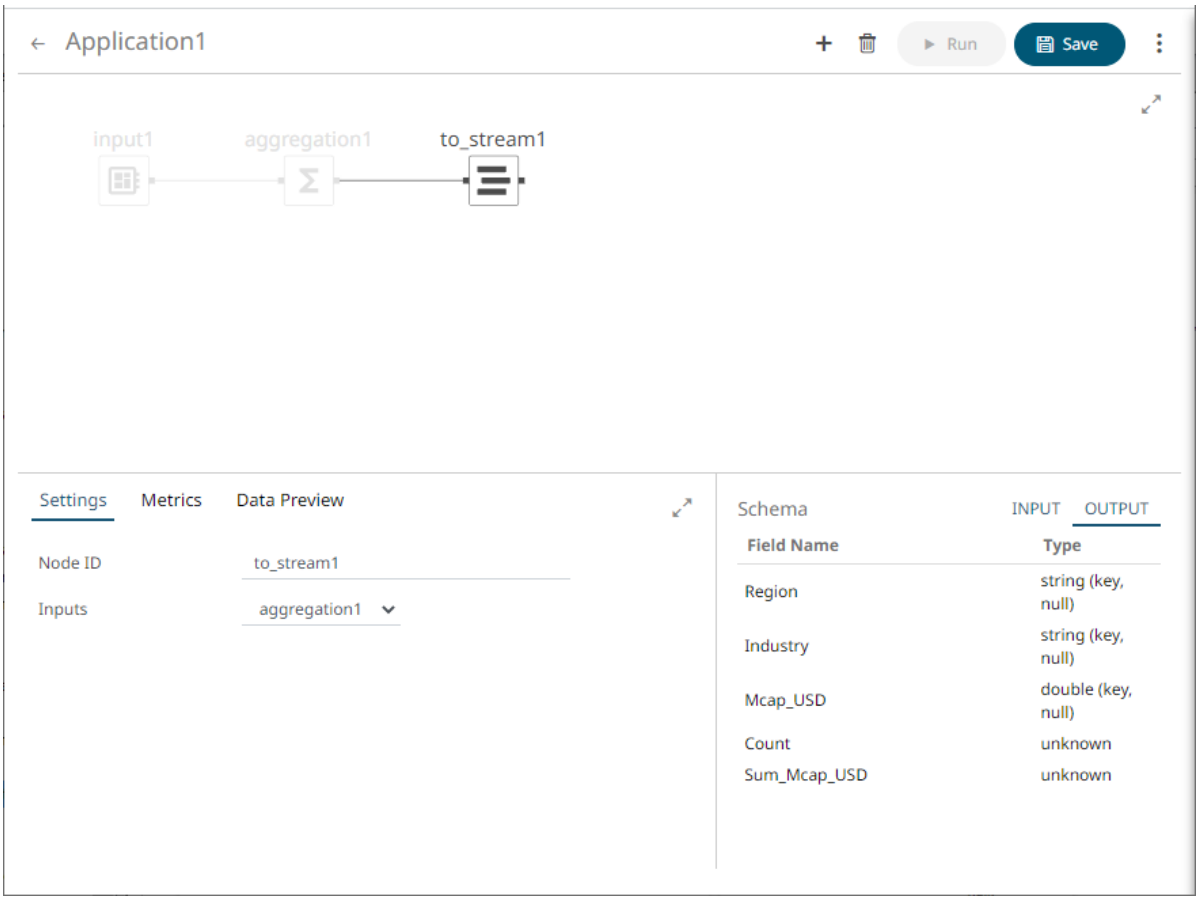
Adding a Table to Stream Operator

Aggregating on delta as a Table causes a change log, producing a single record. The Table to Stream operator morphs the single record back into stream.

Steps:

1. In the *Application* page, click + and select **To Stream** in the *Add Operator* pane.

   The **To_stream** node  icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



   The right (outbound) edge allows you to connect to the other operators.

2. In the *Operator Settings* panel, define or select the following required properties:

Property	Description
Node ID	The ID of the Table to Stream operator.

Inputs	The left input automatically connects to the currently-selected operator. You can select another ID of the operator that will be the source of the data in the *Inputs* drop-down list. Ideally, this should be an aggregation operator. The preview of the data (LEFT, RIGHT, and OUTPUT) are displayed in the *Schema* panel.

The preview of the data (OUTPUT) is displayed in the *Schema* panel.

Schema INPUT OUTPUT

Field Name	Type
Region	string (key, null)
Industry	string (key, null)
Mcap_USD	double (key, null)
Count	unknown
Sum_Mcap_USD	unknown

NOTE

The data types of the aggregated columns are still unknown. The new data type will be applied once the application is saved.

For example:

Schema INPUT OUTPUT

Field Name	Type
Region	string (key, null)
Industry	string (key, null)
Mcap_USD	double (key, null)
Count	long (not null)
Sum_Mcap_USD	double (not null)

3. Save the changes.

Example

```
<tostream>
     <id>to_stream1</id>
</tostream>
```

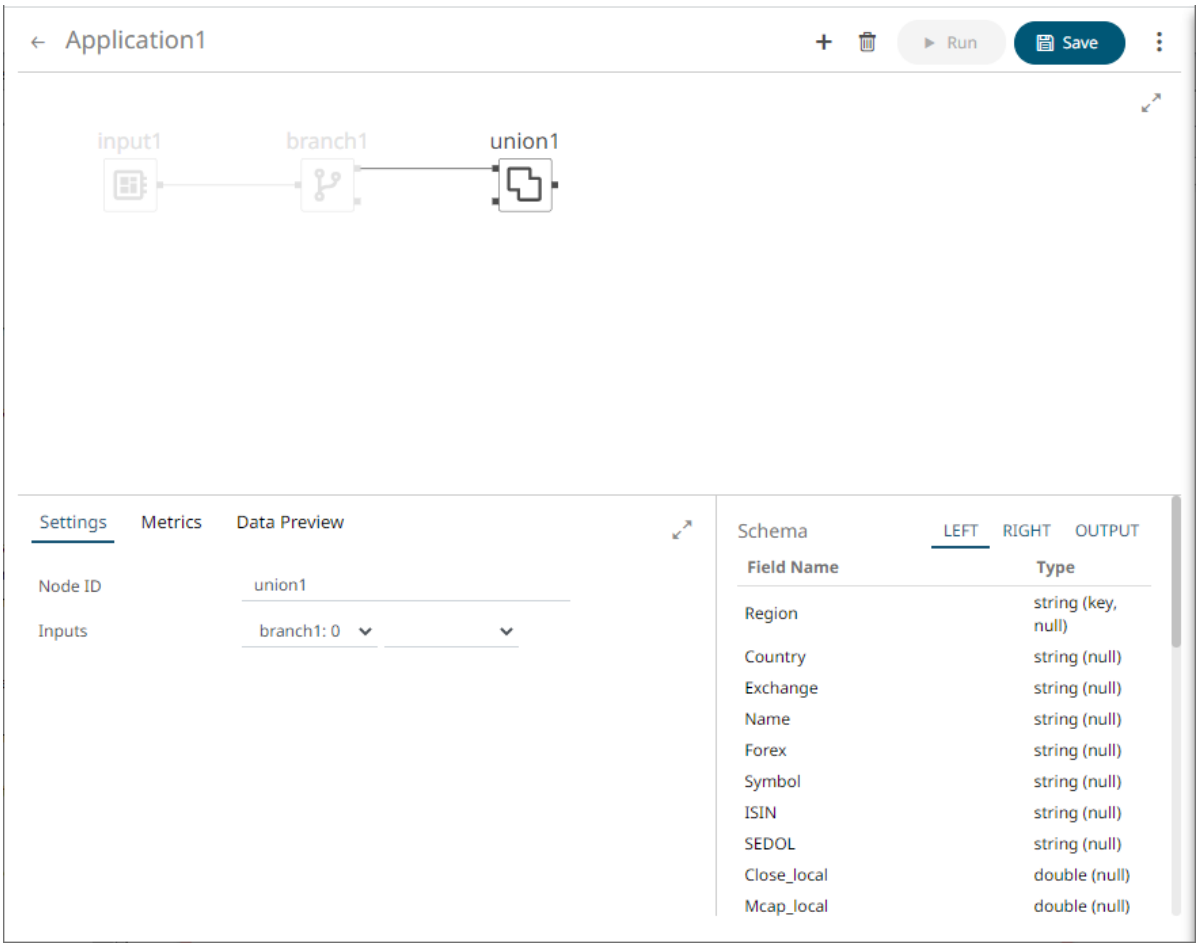
Adding a Union Operator

Used to perform a union of two streams. Both streams would need the same schema. Otherwise, the output would be the combination of both, with missing values returned as Null.

Steps:

1. In the *Application* page, click + and select **Union** in the *Add Operator* pane.

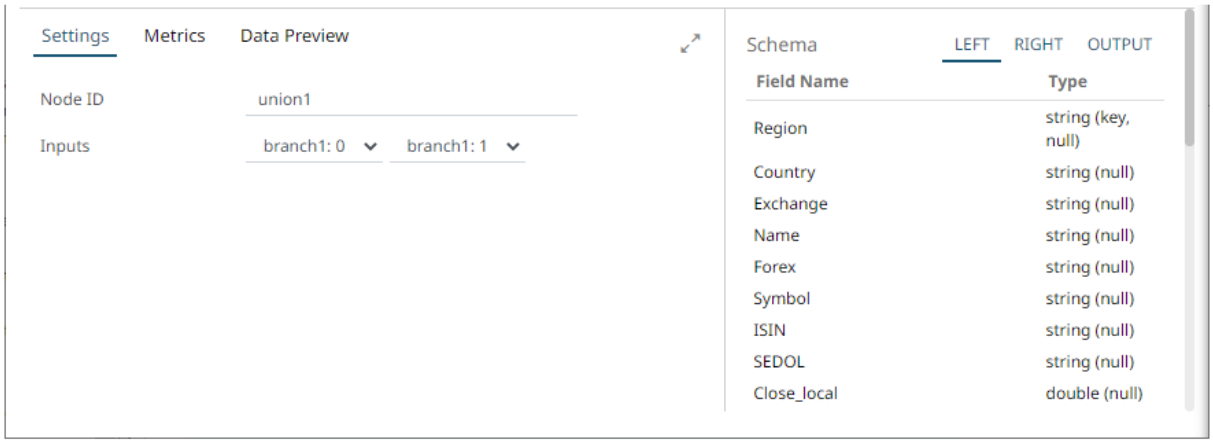
   The **Union** node icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



The left (inbound) edges allow you to select the input streams. The right (outbound) edge allows you to connect to the other operators.

2. In the *Operator Settings* panel, define or select the following required properties:

Property	Description
Node ID	The ID the union operator.
Inputs	The left input stream automatically connects to the currently-selected operator. You can select another ID of the operator that will be the source of the data in the *Inputs* drop-down list. Then select the right input stream. The preview of the data (LEFT, RIGHT and OUTPUT) is displayed in the *Schema* panel.



3. Save the changes.

Adding an Output Operator

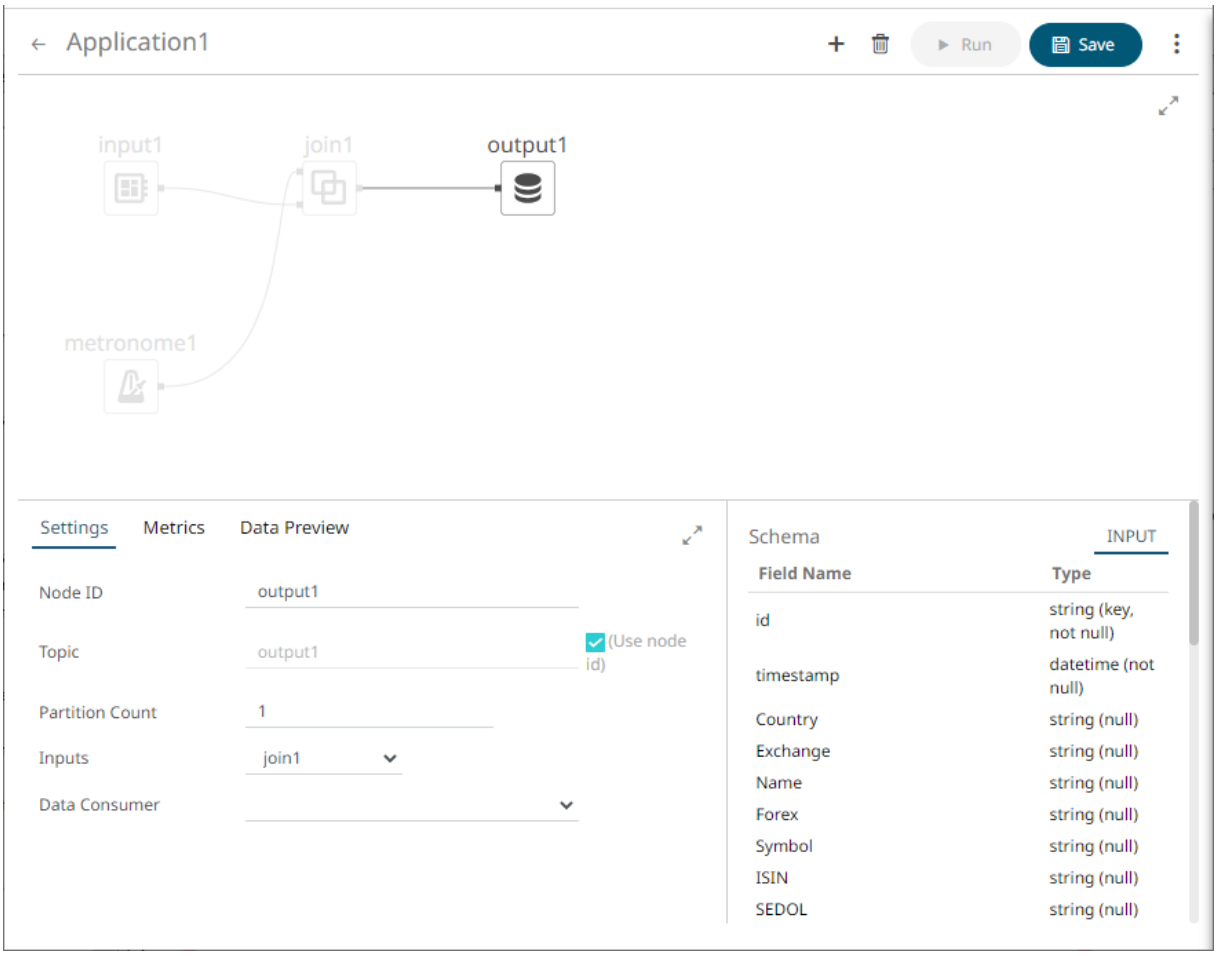
An output produces and publishes streams towards a Kafka topic or a data consumer.

Steps:

1. In the *Application* page, click + and select **Output** in the *Add Operator* pane.

The **Output** node

icon displays in the *Graph* panel, as well as the properties to be defined in the *Operator Settings* panel, and the preview of the data in the *Schema* panel.



The left (inbound) edge allows you to select the input source or operator.

2. In the *Operator Settings* panel, define or select the following properties:

Property	Description
Node ID	The ID of the output operator.
Topic	The stream of records or output you will be subscribed to. Check the *Use Node ID* box to use the value entered in the *Output* ID. Otherwise, uncheck the box and enter a new *Topic* ID. When adding Topic IDs, ensure they: must be unique across an application must be specified must start with a letter (a to Z) or an underscore. Also, it can only contain letters (a to Z), numbers (0 to 9), and underscores
Partition Count	Enter the number of partitions for the Kafka topics that will be created for the Output operator. Partitions allow you to parallelize a topic by splitting the data in a particular topic across multiple brokers wherein, each partition can be placed on a separate machine to allow for multiple consumers to read from a topic in parallel.

Inputs	The left input stream automatically connects to the currently-selected operator. You can select another ID of the operator that will be the source of the data in the *Inputs* drop-down list. The preview of the data (INPUT) is displayed in the *Schema* panel.
Data Consumer	Select the Data Consumer where the output will be produced or published. Currently, the following data consumers are supported: • Text • JDBC databases • InfluxDB • Email • Kx kdb+ • REST • Apache Kafka

3. Save the changes.

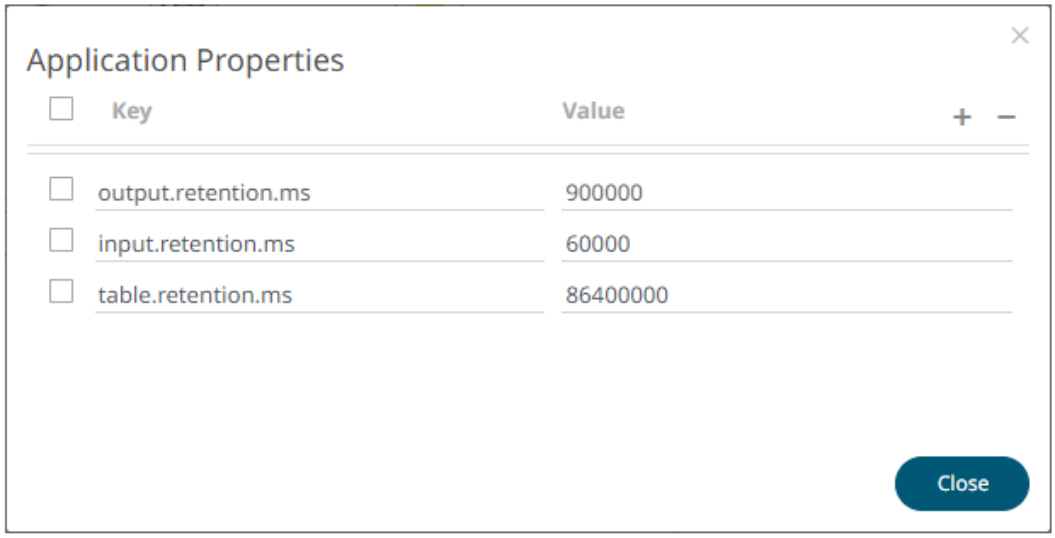
ADDING APPLICATION-SPECIFIC PROPERTIES

Panopticon Streams properties can be viewed and configured in `Streams.properties`. However, some of these server-wide properties can be overridden by adding and customizing them in an application.

Steps:

1. In the *Application* page, click then select **Properties**.

   The *Application Properties* dialog displays.



NOTE

- **Currently, the application properties are used to assign specific retention time (in milliseconds) for topic(s).**
- ***Partition Count* values that were added in operators in the application are displayed.**

2. To add a property, click .

   A new row for *Key* and *Value* entry displays.

NOTE **The *Keys* and *Values* must not be empty. Also, keys must be unique within the application property list.**

3. Enter the *Key*. This is the application property to be configured.
4. Enter the corresponding *Value* of the key.

   You can also opt to delete an application property entry by checking its box and clicking .

5. Click

.

Example

```
<properties>
    <!-- Keep tables alive one day -->
    <entry>
        <key>table.retention.ms</key>
        <value>86400000</value>
    </entry>
    <!-- Keep input and output streams for 1 second -->
    <entry>
        <key>input.retention.ms</key>
        <value>1000</value>
    </entry>
    <entry>
        <key>output.retention.ms</key>
        <value>1000</value>
    </entry>
    <!-- Custom retention time for InputStream topic -->
    <entry>
        <key>TimeSeries.retention.ms</key>
        <value>1111</value>
    </entry>
</properties>
```

Refer to **RetentionTimeExample** in the Example Applications section for more information

SAVING AN APPLICATION

Saved applications (.app) are available in the `PanopticonAppdata\CEP\Applications` folder (i.e., `c:\streamsserverdata\CEP\Applications`).

Steps:

1. In the *Application* page, you can either click:

   - the Save icon
   - the icon. The context menu displays with three saving options:
     - Save

       Click to save the changes made in the application.

     - Save as Copy

       Click to make a duplicate of the application. The original name is appended with **_Copy**.

       To change the *Application Name*, click on it to make it editable, then enter a new one and click .

NOTE	**The *Name* or ID must start with a letter (a to Z) or underscore. Also, it can only contain letters (a to Z), numbers (0 to 9), and underscores.**

     - Revert to Saved

       Click to revert to the previously-saved application settings.

NOTE	• **After saving, you can start the execution of the application. You can do this either in the *Application* page or on the Applications tab.** • **When saving an application, the color will indicate the status of the nodes:** ○ **Black – no issue** ○ **Yellow – no traffic on the topic** ○ **Red – there are definition issues. Refer to *Validating and Fixing Application Issues* for more information.**

EDITING AN APPLICATION

NOTE **Applications that are started or running cannot be edited.**

Steps:

1. On the **Applications** tab, click an application link to modify.

   The *Application* page displays.

2. To change the *Application Name*, click on it to make it editable, then enter a new one and click ✓ .

   NOTE **The *Name* or ID must start with a letter (a to Z) or underscore. Also, it can only contain letters (a to Z), numbers (0 to 9), and underscores.**

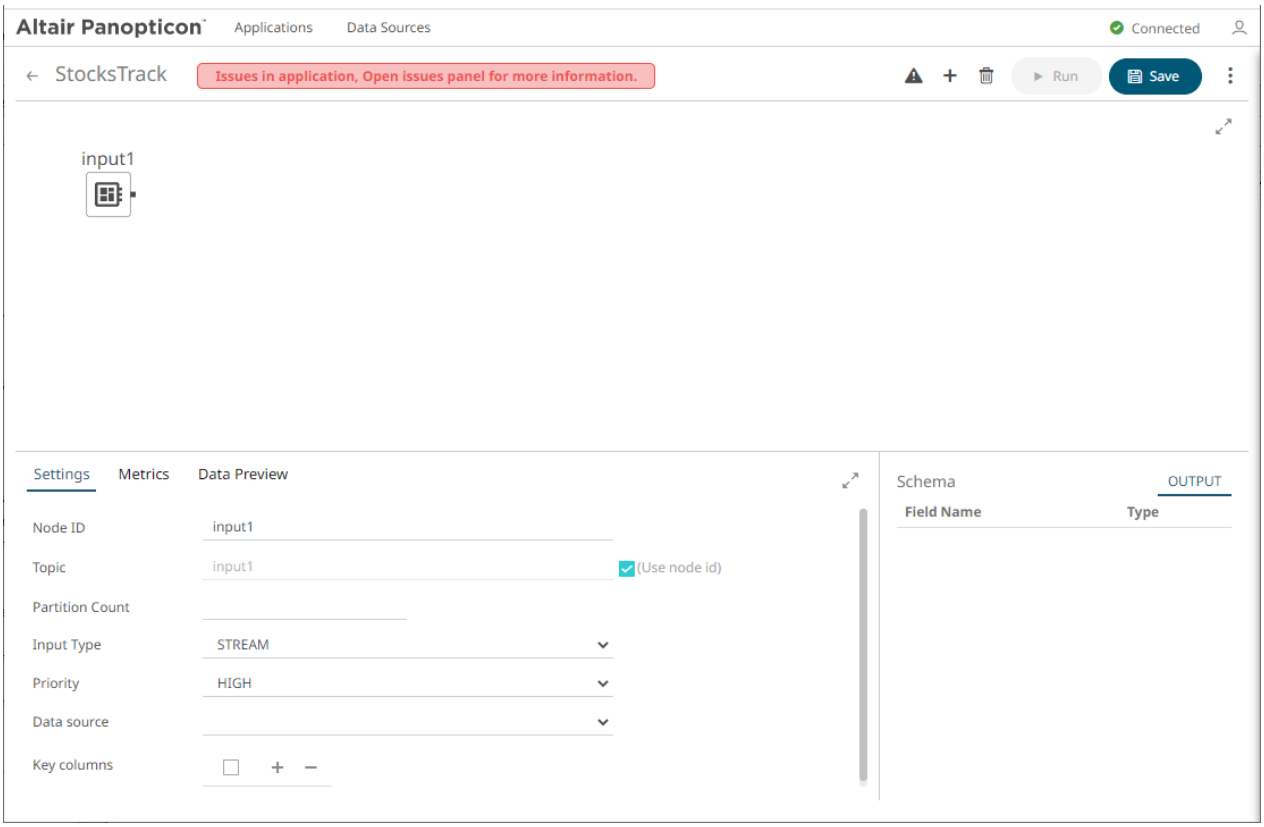
3. You can also modify or add the following:
   - operators
   - properties
4. Save the changes.

   To go back to the **Applications** tab, click ← beside the application name.

   NOTE **If there are changes that were not saved, a confirmation message displays asking if you will exit the *Application* page without saving. Click Cancel and then save.**

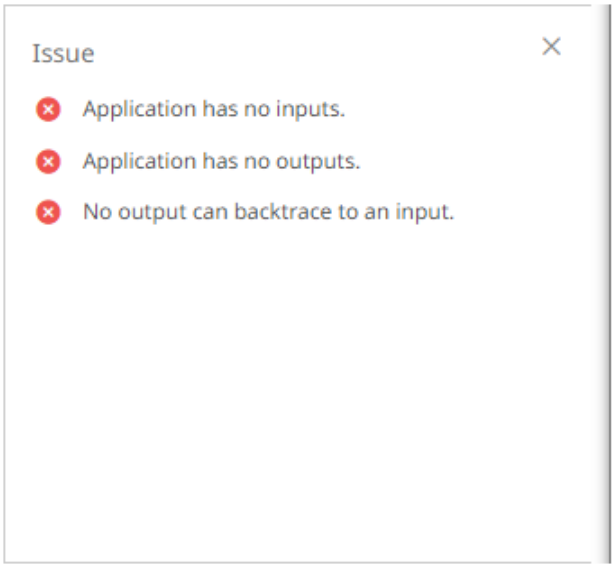
VALIDATING AND FIXING APPLICATION ISSUES

Panopticon Streams provides an error validation to help fix application issues.



Steps:

1. Click **Show Issues** . The list of *Issues* is displayed with the *Source* or operator with an error.



2. Click an issue to select.

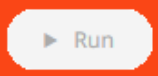
   Some possible issues include the following:

   - for the input nodes, the data source is not available
   - the application model parts are still not complete, or has invalid values
   - for all nodes except inputs, there are faulty input definition or missing input connection
   - for all nodes except outputs, there are faulty output definition or missing output connection

3. Apply the necessary changes and save.

STARTING AN APPLICATION

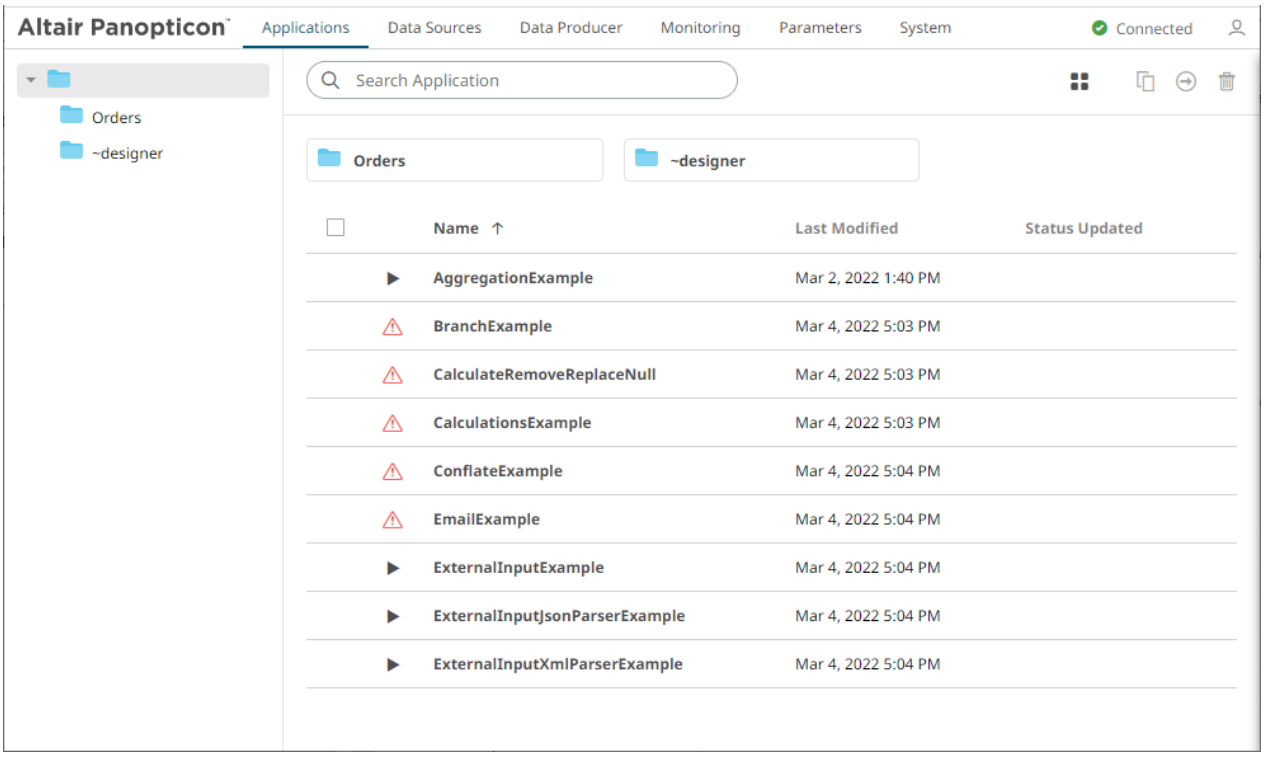
NOTE

- **Before starting an application, ensure:**
  - **the CEP engine has been started**
  - **the prerequisite data sources are uploaded on the Data Sources tab**
  - **the application model is defined correctly**
- **If the application is empty, the** 

icon is disabled. Refer to Creating a New Application for more information.

You can start an application either on the Applications tab or on the *Application* page.

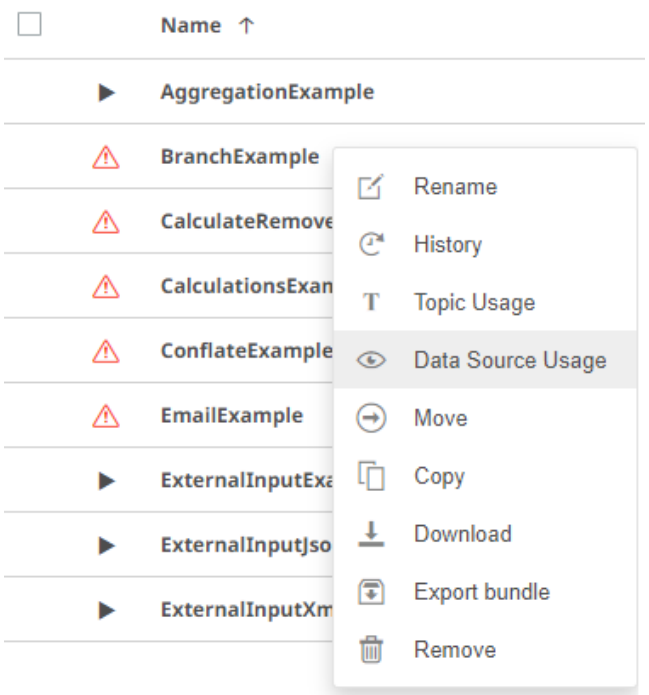
Starting an Application on the Applications Tab



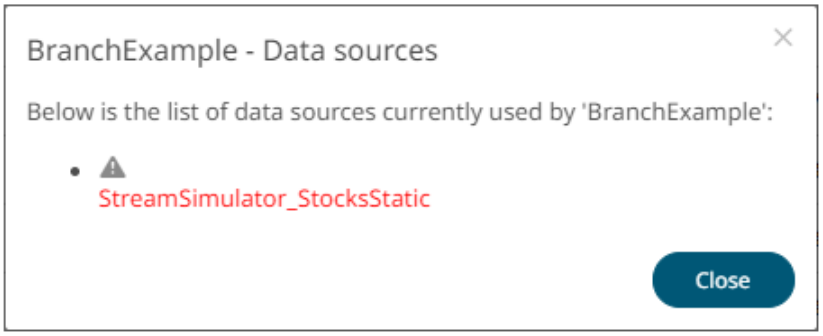
Steps:

1. To execute an application, ensure the ▶ icon displays before the *Name*. This means the necessary data sources are already uploaded.

   However, if ⚠ is displayed, right-click on the application and select **Data Source Usage** on the context menu.



The list of data sources used by the application is displayed. For example:



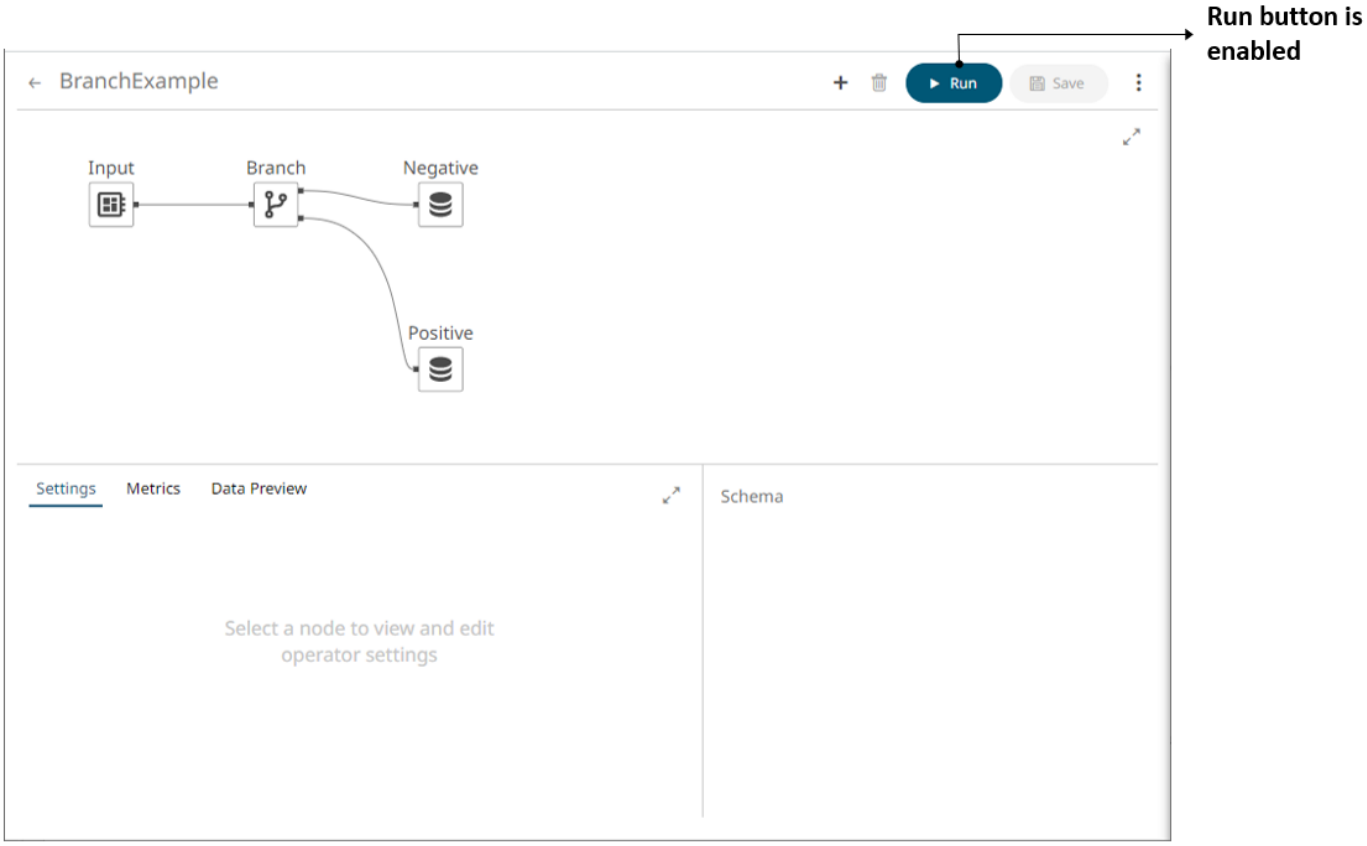
Refer to *Uploading Data Sources* or Creating a Data Source for more information.

2. Click ▶. The icon changes to ■ and the timestamp is displayed under the *Status Updated* column.

   The stream topics and the data producer are also generated.

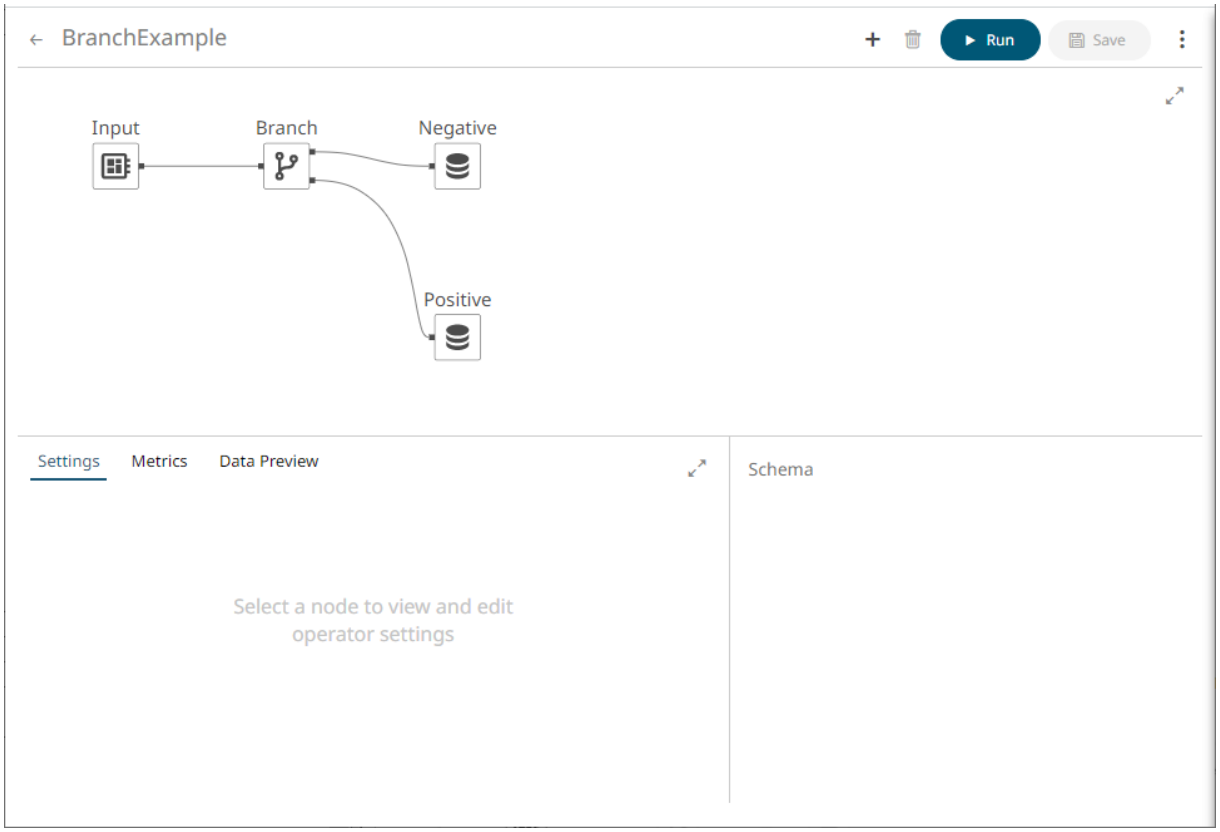
Starting an Application on the Application Page

Users with a Designer role have the ability to open and manage applications.



Steps:

1. On the **Applications** tab, click an application link to open and display it on the *Application* page.



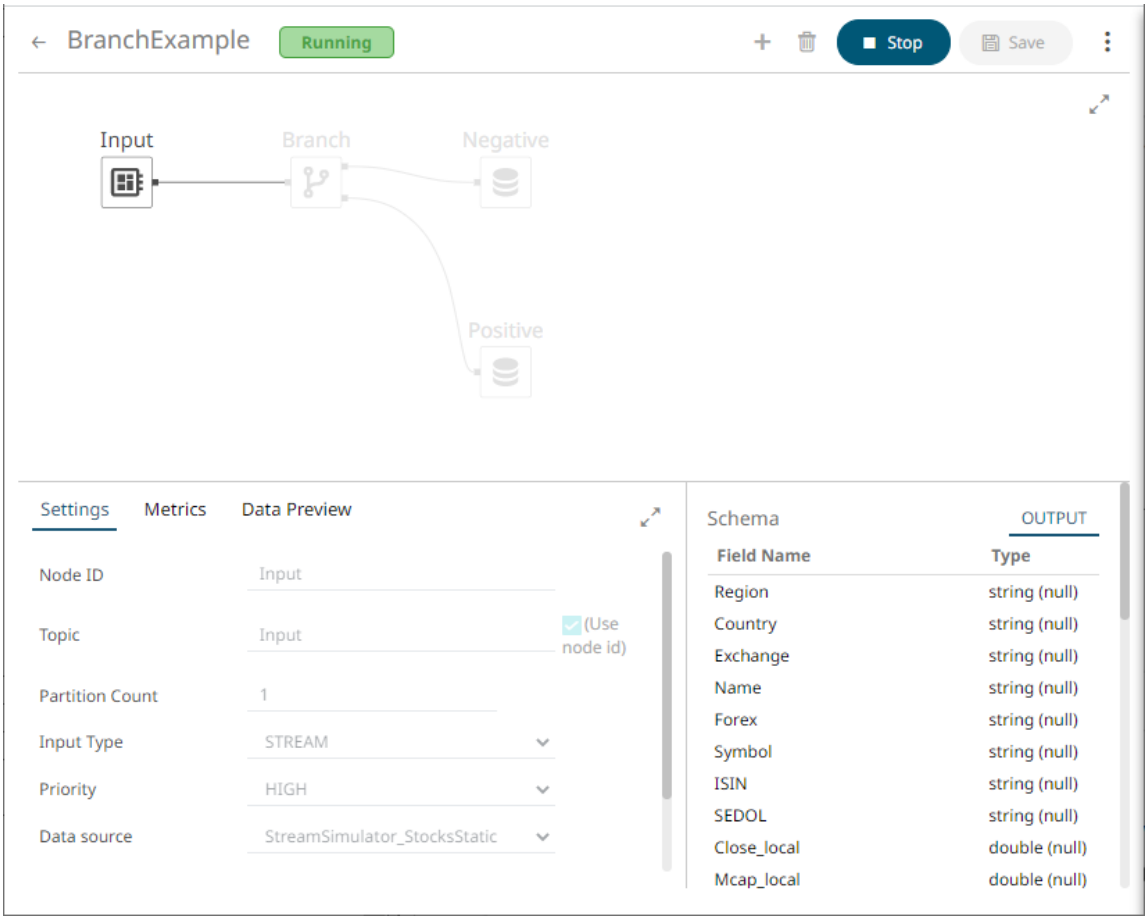
2. Click to run the application.

   The button changes to and the **Running** status displays.

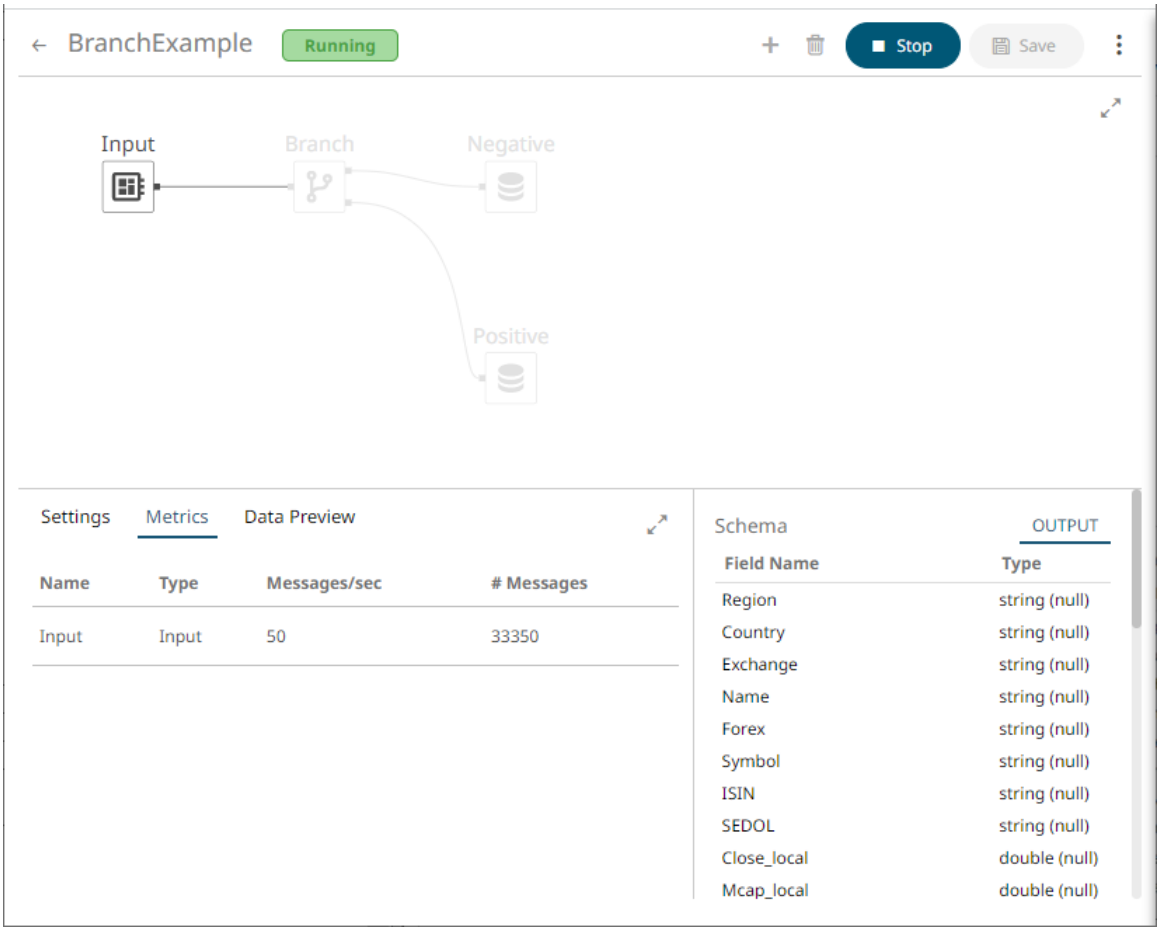
   The stream topics and the data producer are generated.

3. You can also perform the following:

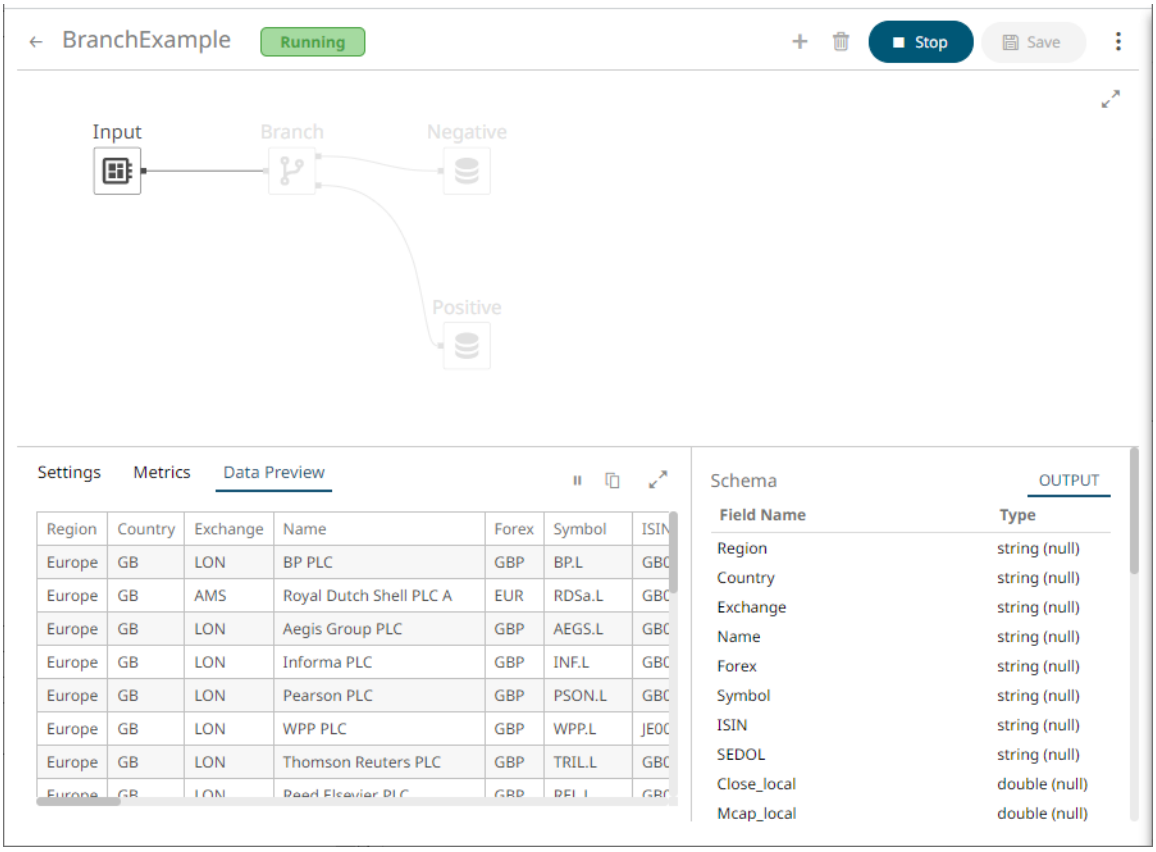
   - click on a node in the *Graph* panel to display its *Operator Settings* as well as the preview of the data (OUTPUT) in the *Schema* panel



- click the Metrics tab to display the node's *Metrics* as well as the preview of the data (OUTPUT) in the *Schema* panel


BranchExample
Running
Stop
Save
Input
Branch
Negative
Positive
Settings
Metrics
Data Preview
Name
Type
Messages/sec
Messages
Input
Input
50
33350
Schema
OUTPUT
Field Name
Type
Region
string (null)
Country
string (null)
Exchange
string (null)
Name
string (null)
Forex
string (null)
Symbol
string (null)
ISIN
string (null)
SEDOL
string (null)
Close_local
double (null)
Mcap_local
double (null)

- select an input or output node and click the **Data Preview** tab to display the data preview.



You can also click:

- to copy the data to a clipboard.
- to pause the update. To start the update, click .

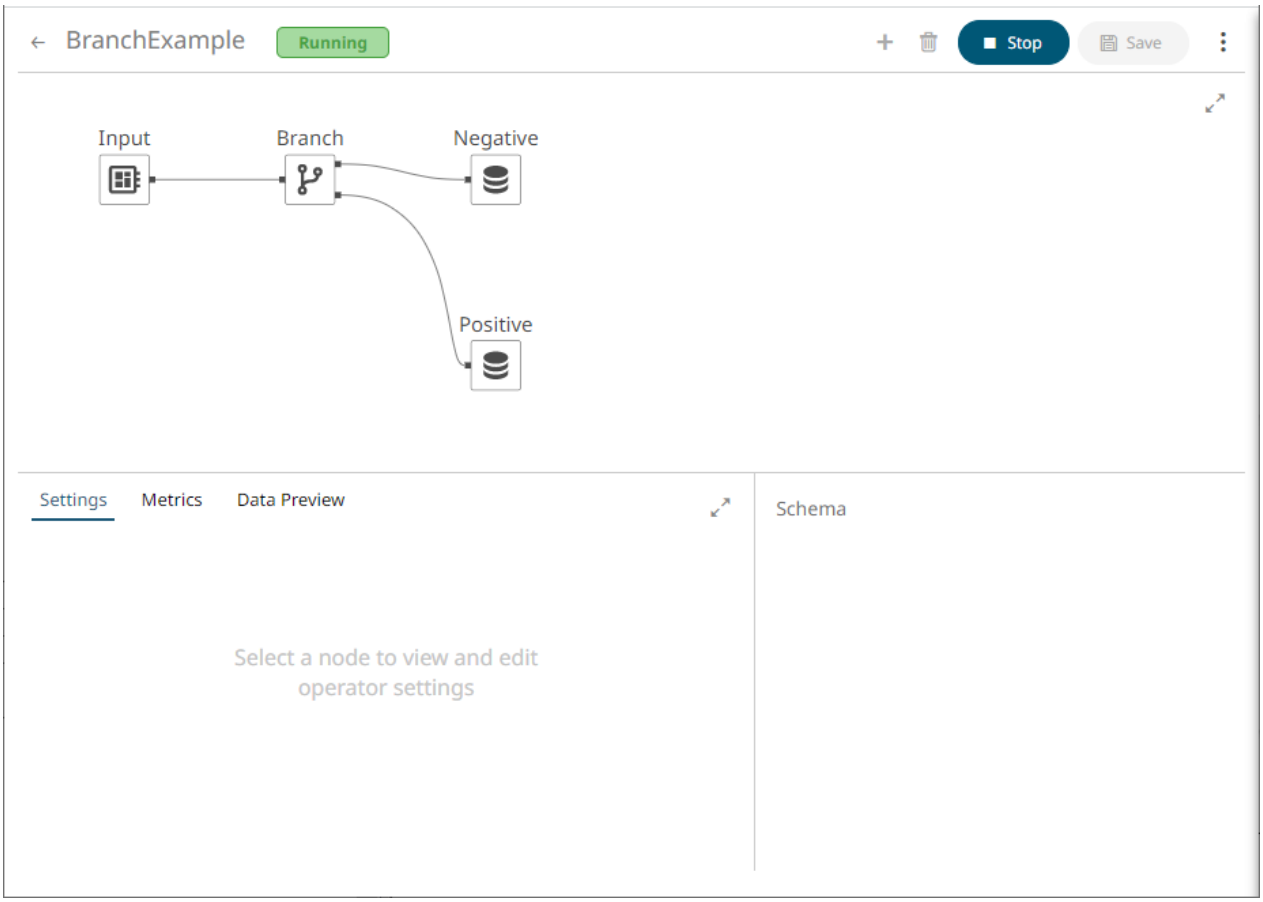
STOPPING AN APPLICATION

Stopping the execution of an application can either be done on the Applications tab or the *Application* page.

Stopping an Application on the Applications Tab

Click to stop the execution of the application. The icon is changed back to .

Stopping an Application on the Application Page

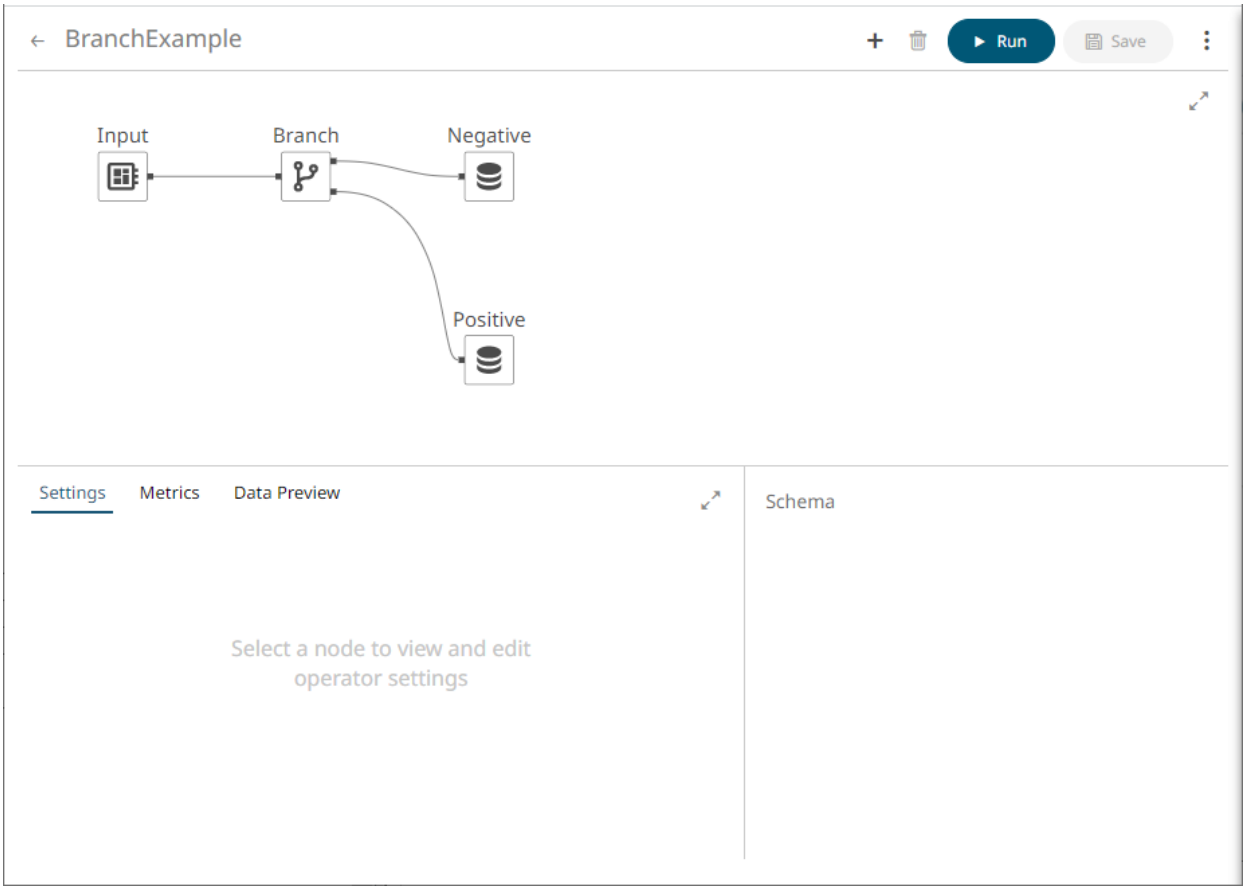


Click

to stop the execution of the application. The

changes to

.


BranchExample
Run
Save
Input
Branch
Negative
Positive
Settings
Metrics
Data Preview
Schema
Select a node to view and edit operator settings

[9] MANAGING DATA SOURCES

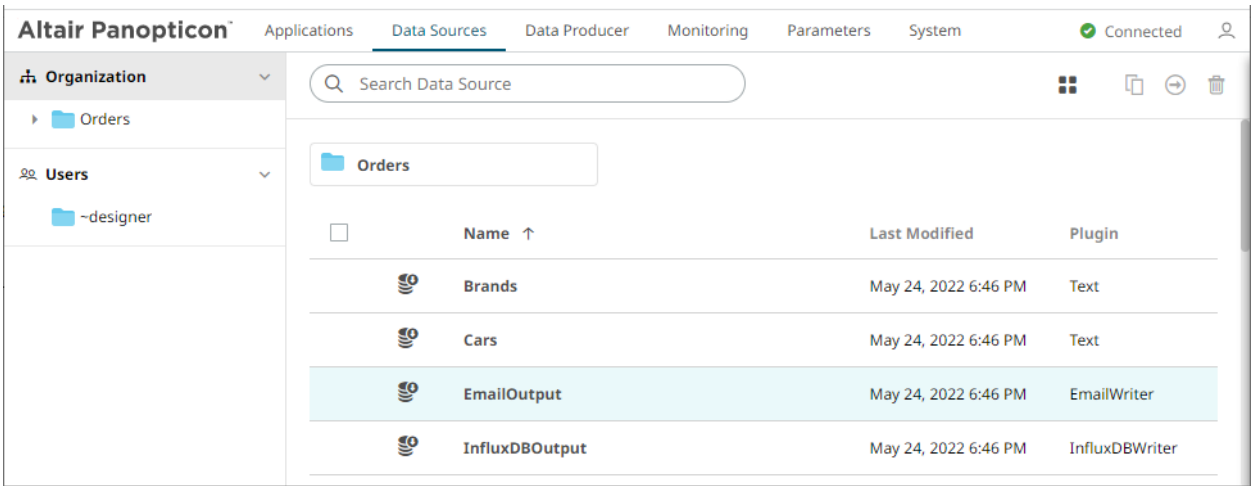


Figure 9-1. Data Sources page for the Administrator user role

On the **Data Sources** page, users with Administrator or Designer role can:

- rename data sources
- View application usages
- move or copy data sources to folders or subfolders to which the user has permission
- download a copy
- remove a data source

To create or upload a data source, a user must have a Designer role.

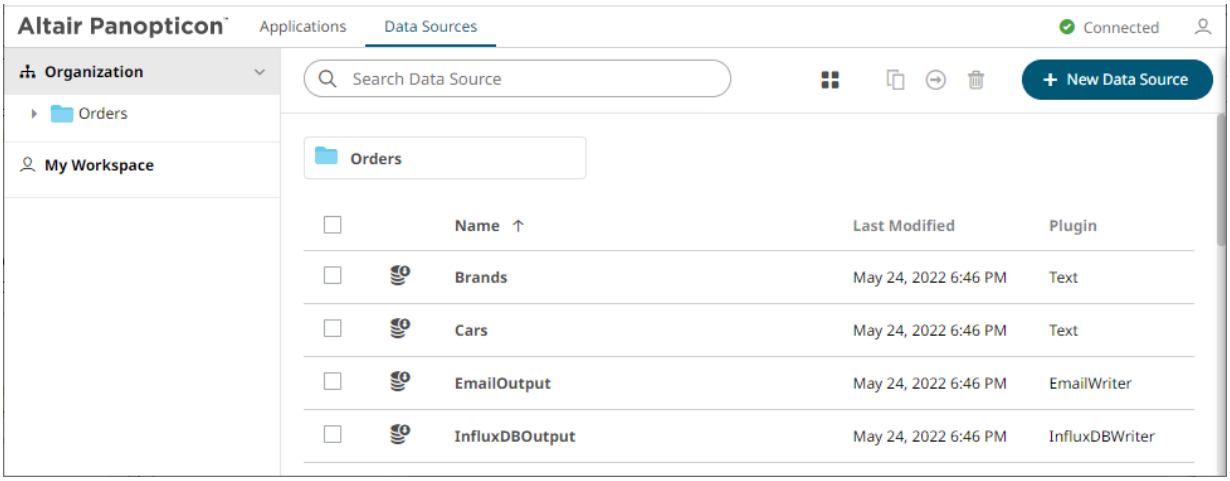


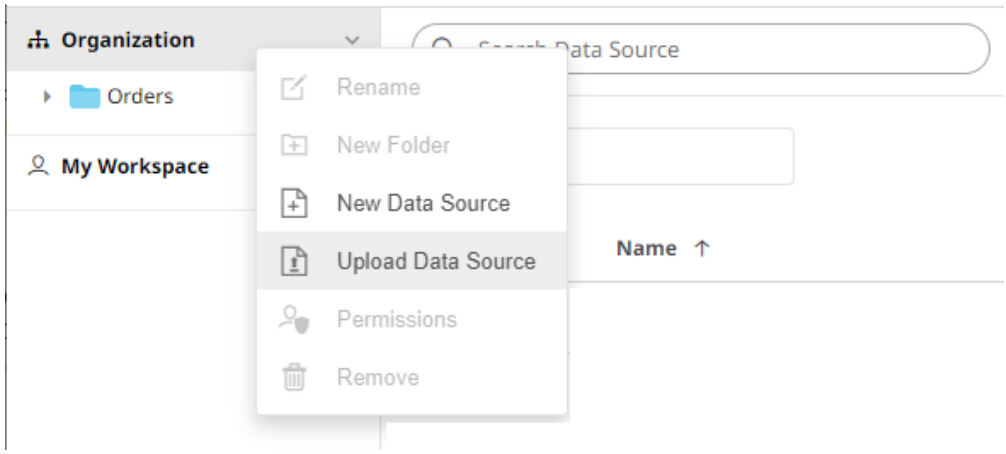
Figure 9-2. Data Sources page for the Designer user role

UPLOADING DATA SOURCES

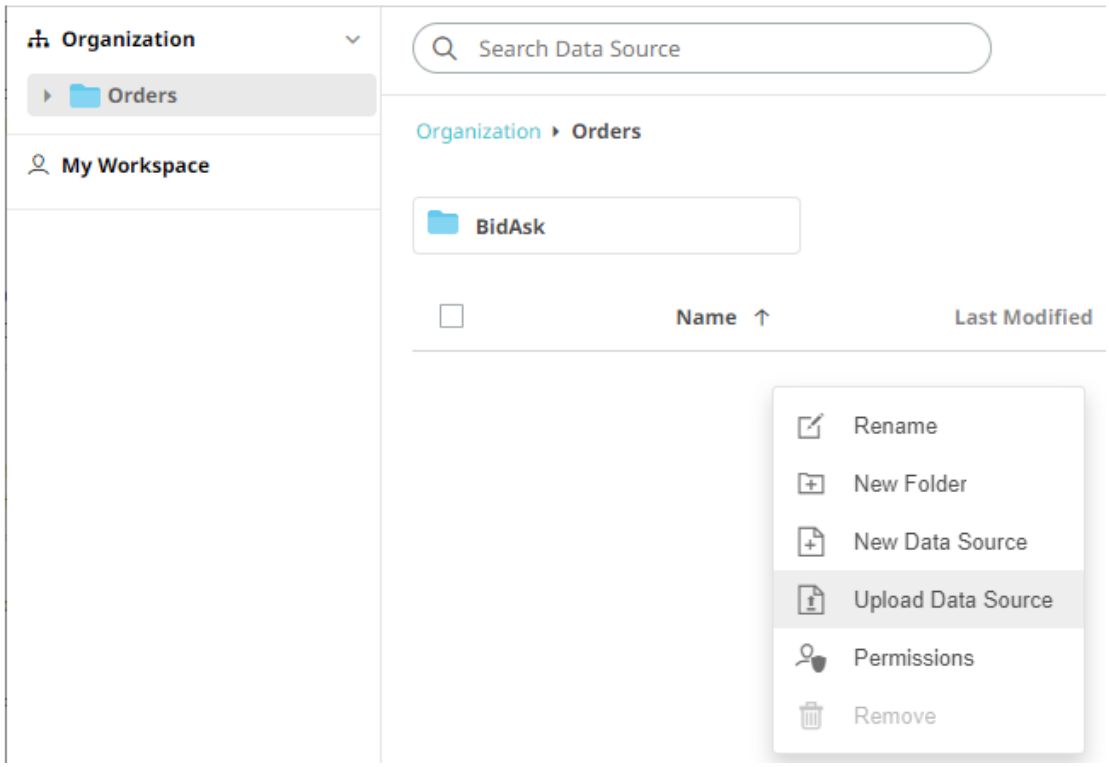
Users with a Designer role can upload data sources to folder or subfolders to which they have permission.

Steps:

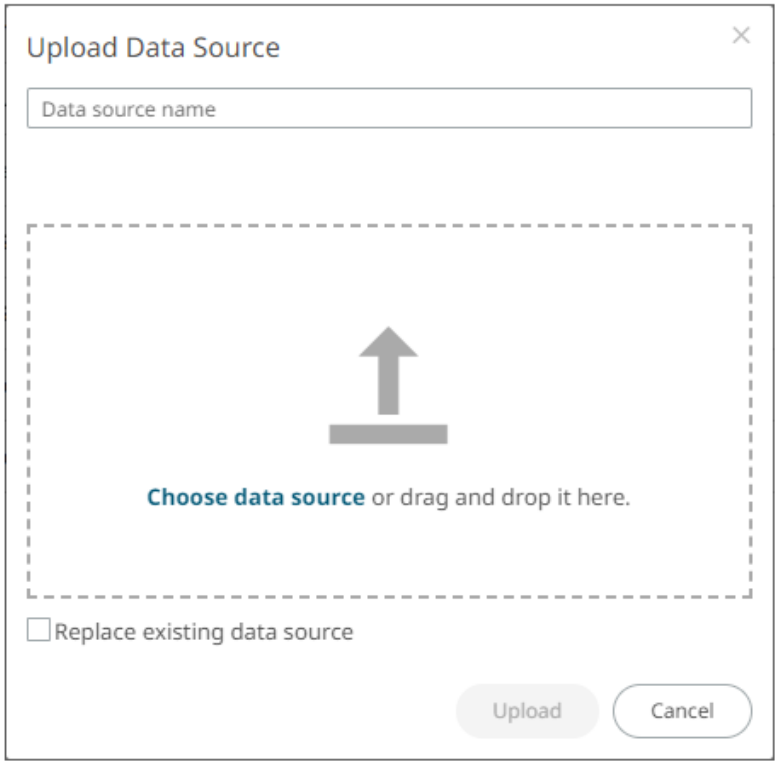
1. To upload data sources, you can either right-click a folder or subfolder then select **Upload Data Sources**:
   - on the expanded *Folder* hierarchy list



   - or on the *Data Sources* pane

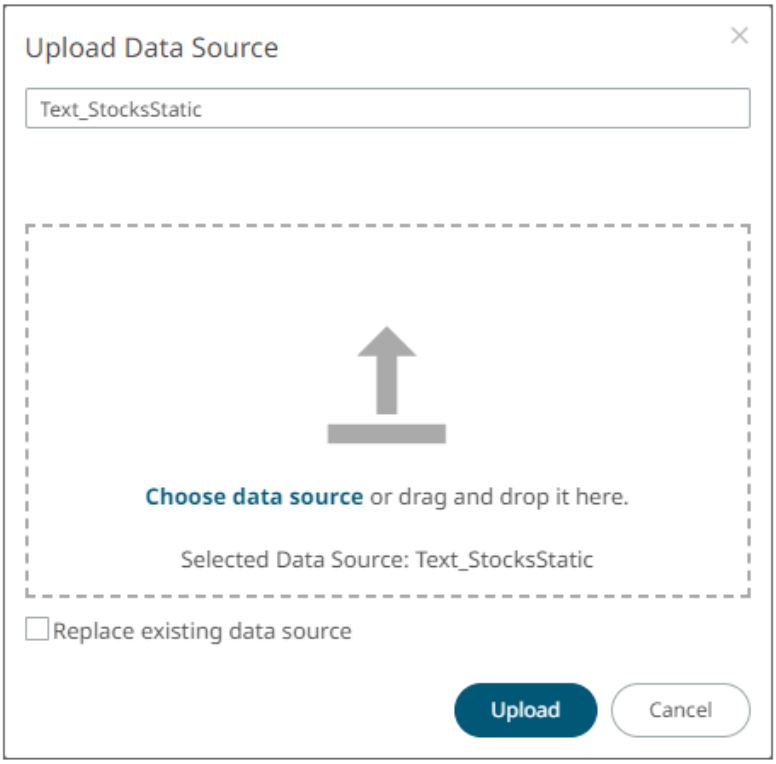


The *Upload Data Source* dialog displays.



2. To upload a data source, you can either:
   - drag it from your desktop and drop in the dialog, or
   - click **Choose Data Source** and select one in the *Open* dialog that displays.

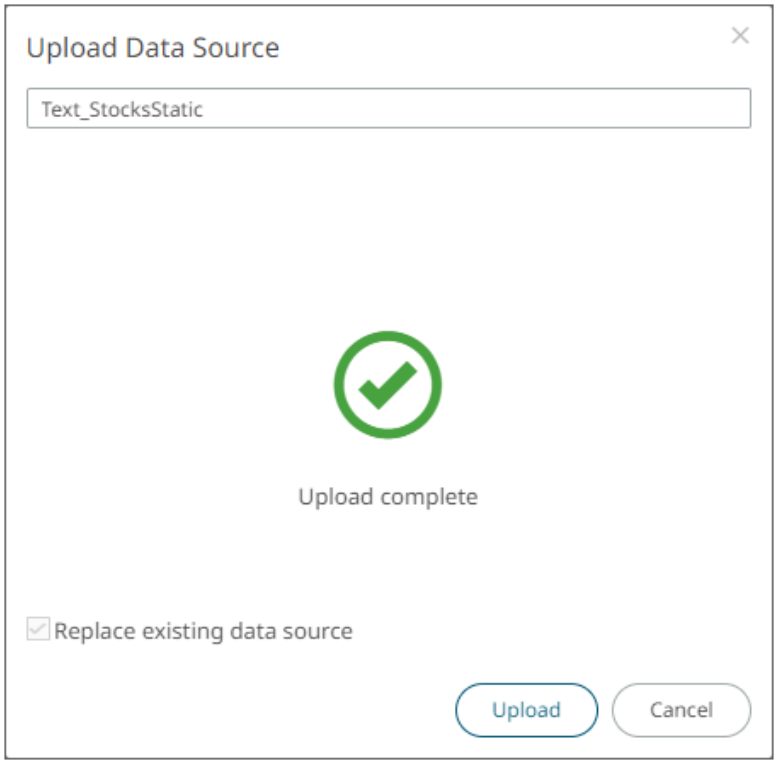
     The name of the data source is displayed on the uploaded data source area and in the *Name* box.



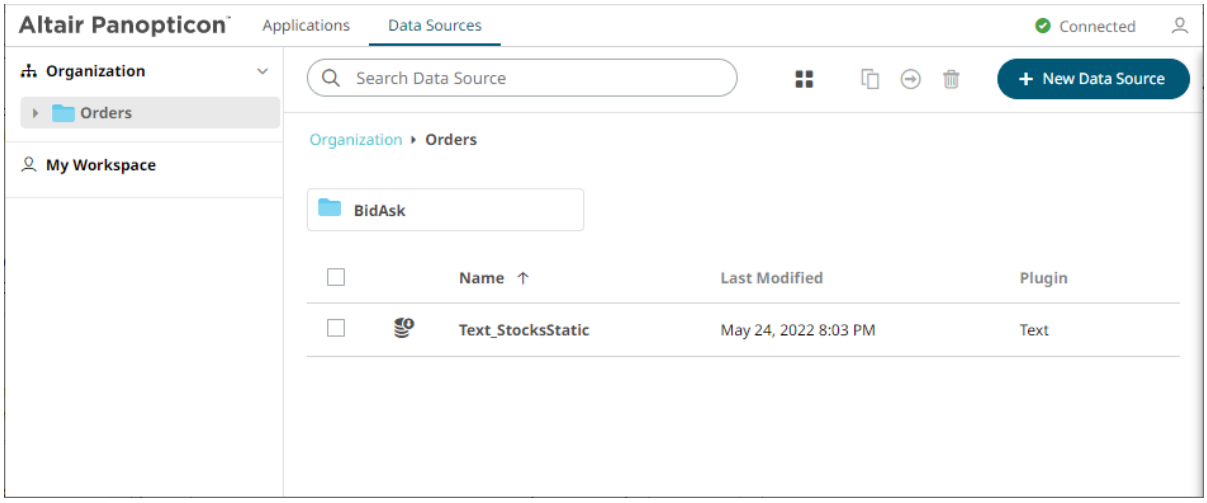
3. You can opt to rename the data source.
4. To replace an existing data source, check the *Replace existing data source* box.
5. Click

.

   You will be notified when the data source has been uploaded.

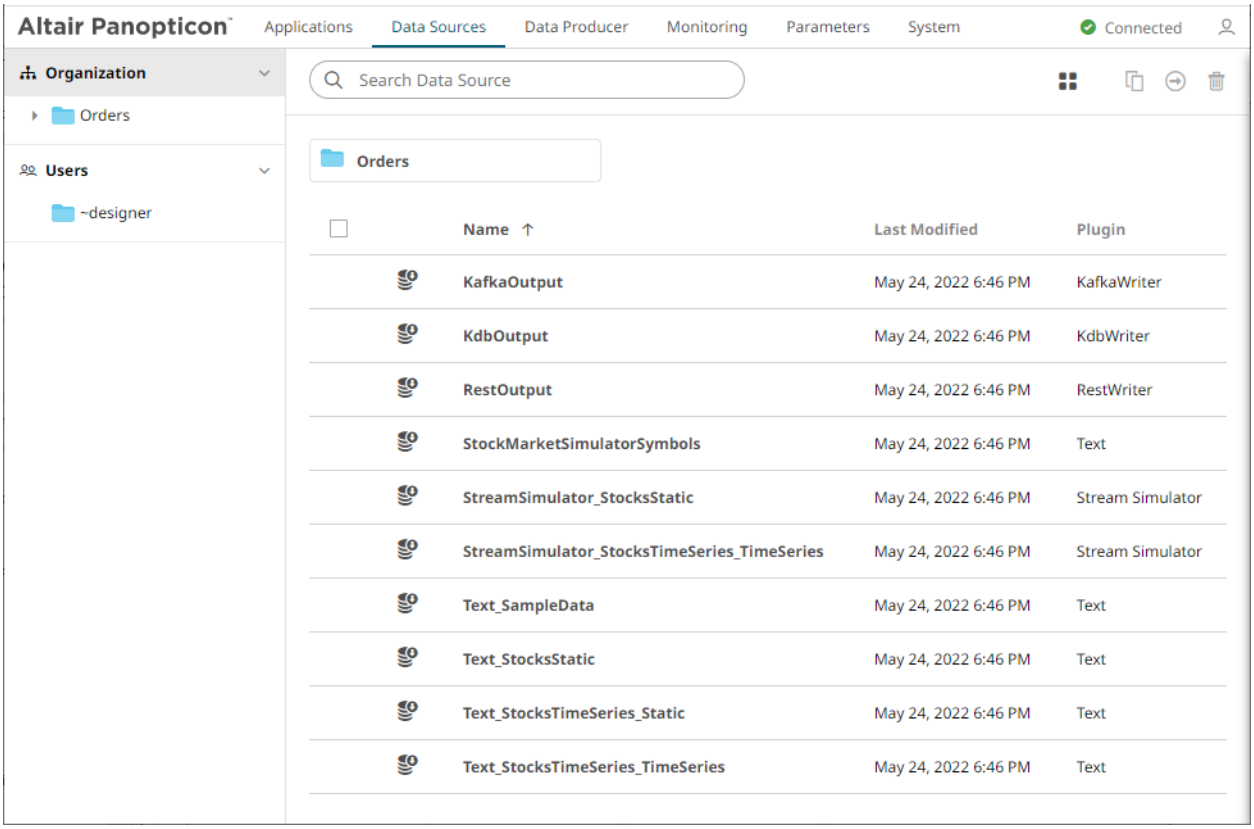


The data source is added and displayed on the **Data Sources** tab.



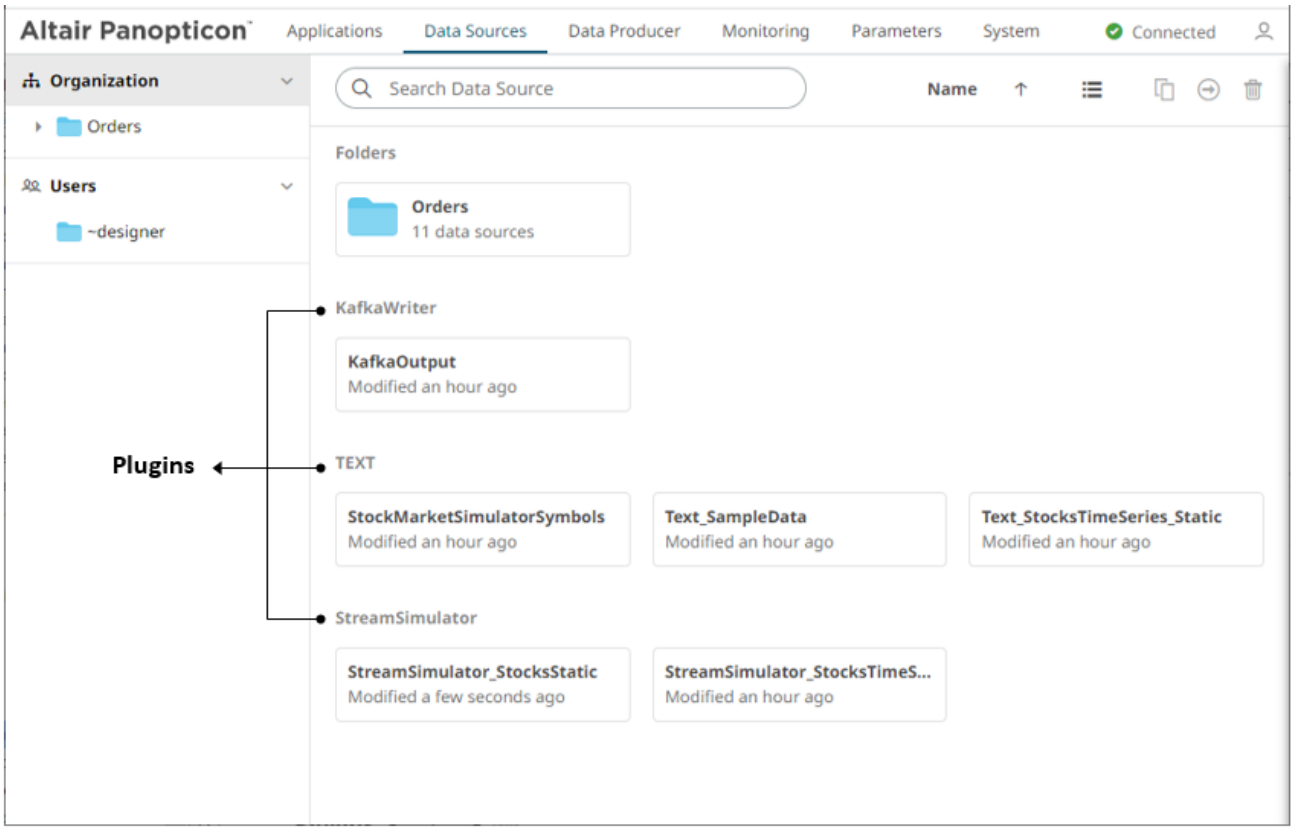
Folders and Data Sources Display View

Folders and data sources can be displayed either on a *List* or *Grid View*. By default, the folders and data sources are displayed in the *List View*.



On the toolbar, click **Grid View** . The folders and data sources are displayed as thumbnails.

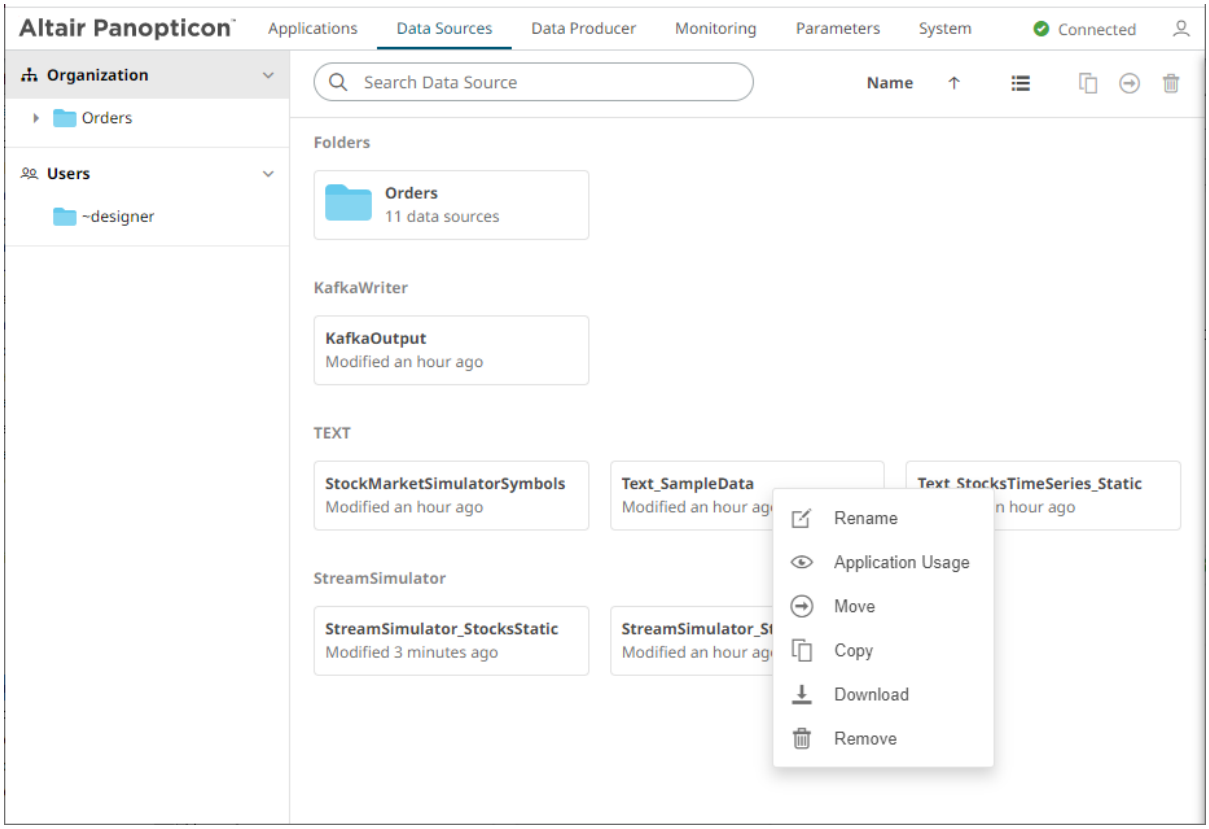
NOTE **Data Sources are placed under their corresponding plugin.**



Click **List View** to return to the standard listing.

On either display view style:

- clicking on a data source title or thumbnail displays the data source
- right-clicking on a data source displays the context menu



SORTING THE LIST OF DATA SOURCES

Sorting data sources can be done by *Name*, *Last Modified*, or *Plugin*.

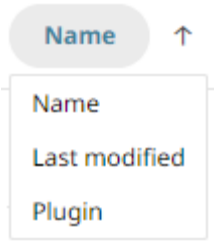
Sort By	Default Sort Order
Name	Ascending
Last Modifed	Descending
Plugin	Ascending

Steps:

On the *Folders and Data Sources Summary* layout, either:

- click the **Sort By** option on the *Toolbar* of the *Grid View*

  By default, the sorting is by **Name**.

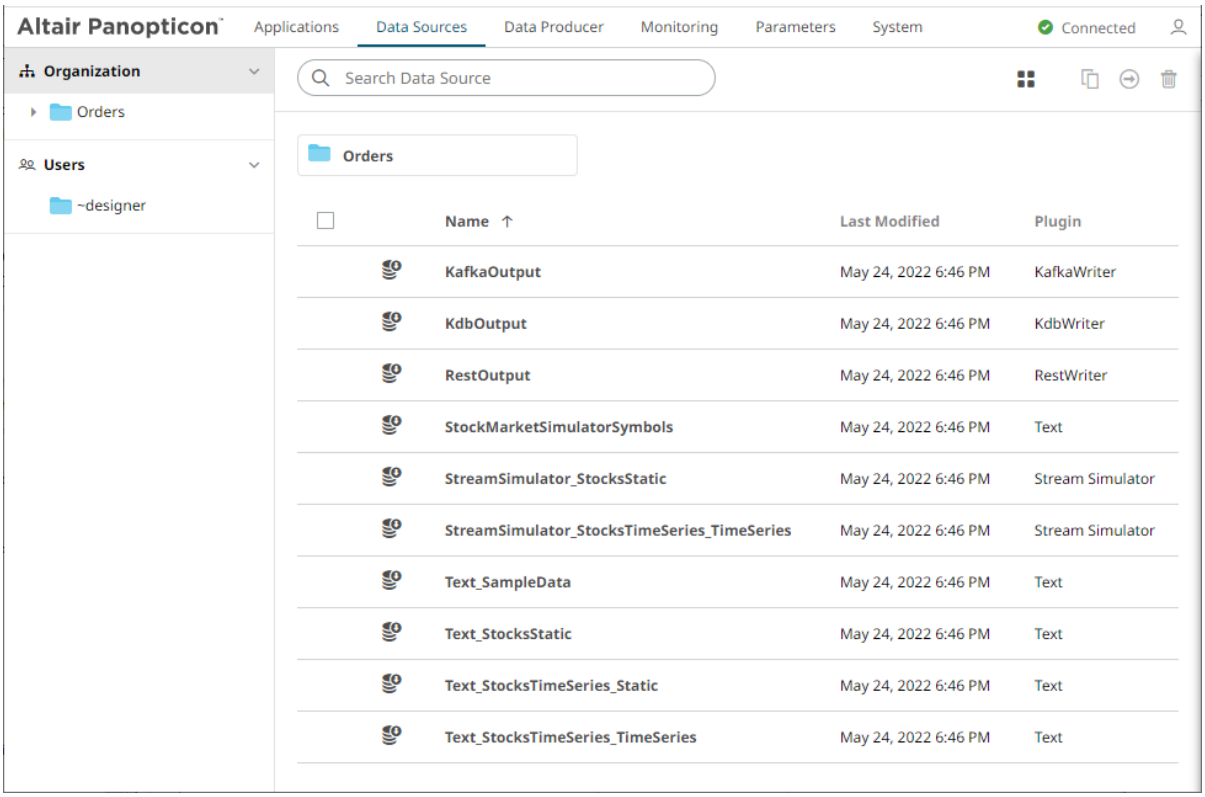


- Name
- Last Modified
- Plugin

Then click the *Sort Order*:

- ↑ Ascending
- ↓ Descending

- click on the **Name**, **Last Modified**, or **Plugin** column header of the *List View*

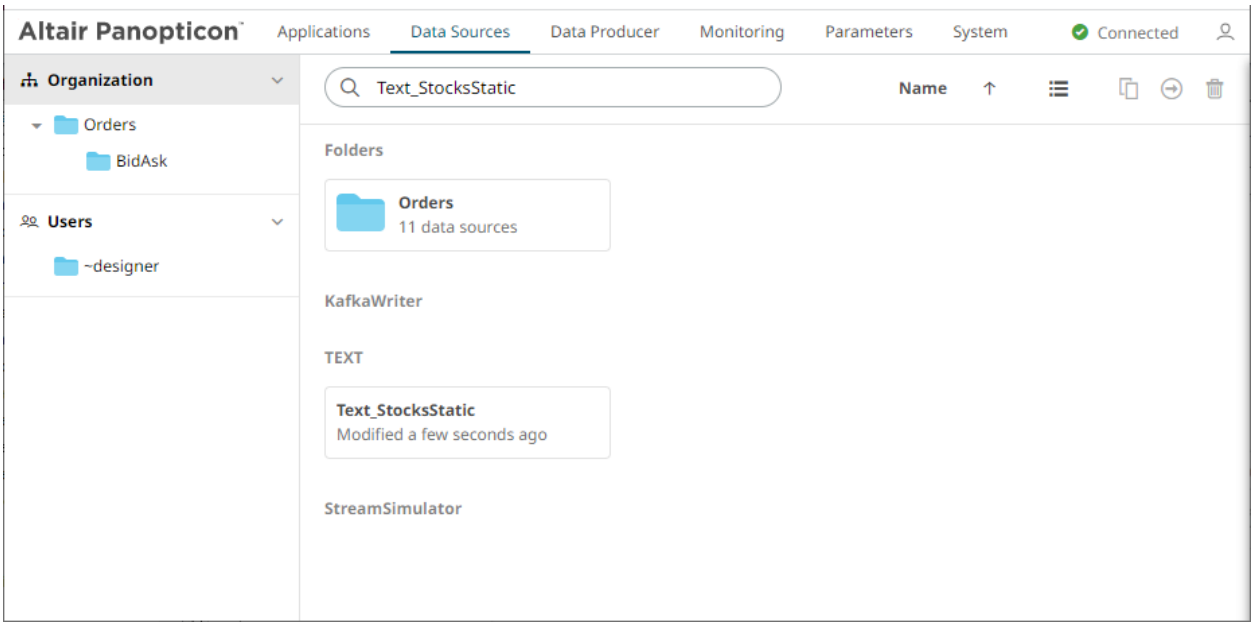


Then click the *Sort Order*:

- ↑ Ascending
- ↓ Descending

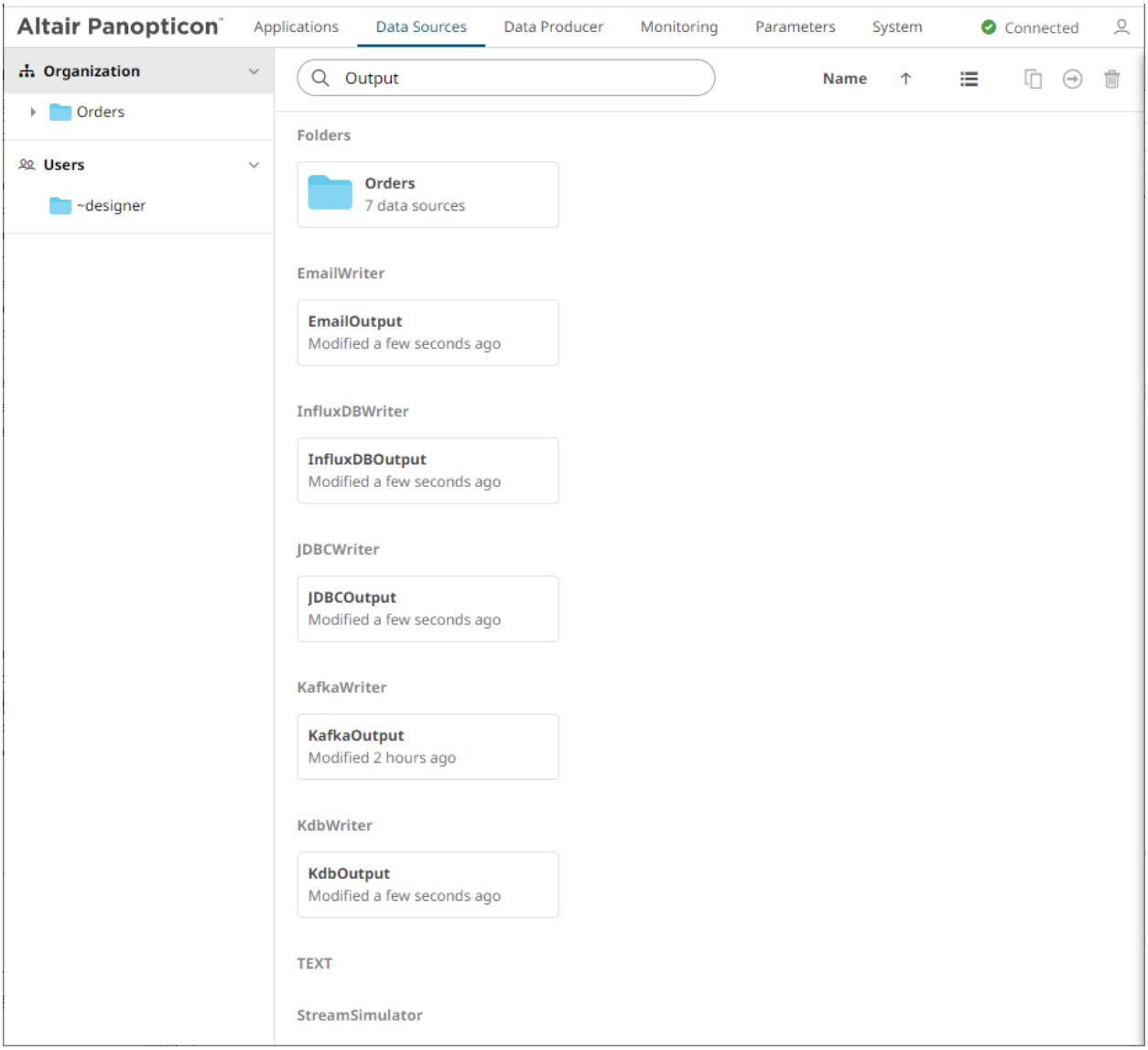
SEARCHING FOR DATA SOURCES

To search for a particular data source, enter it in the *Search Data Source* box.



Searching data sources in the Grid View

You can also enter one of more characters into the *Search Data Source* box and the suggested list of data sources that matched the entries will be displayed.

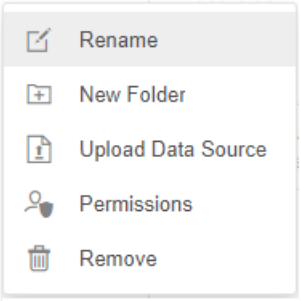


RENAMING DATA SOURCES OR FOLDERS

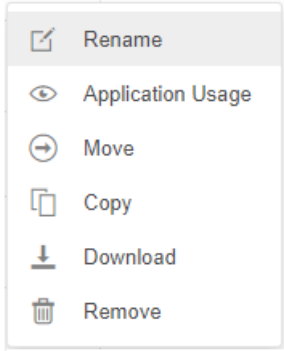
A user with an Administrator or Designer role can rename data sources and folders.

Steps:

1. Right-click on a data source or folder then select **Rename** on the context menu.

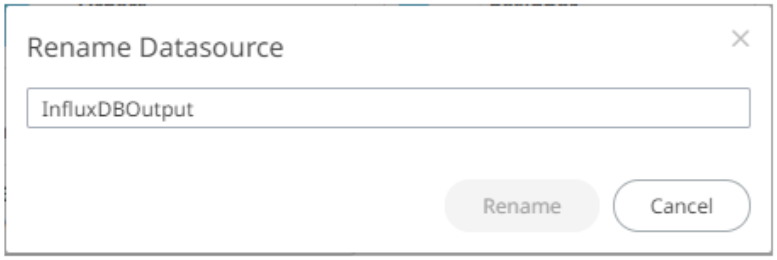


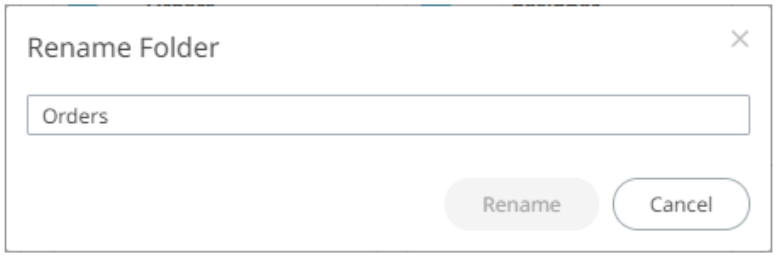
Folder or Subfolder Context Menu



Data Source Context Menu

The *Rename Data Source* or *Rename Folder* dialog displays.





2. Enter a new name then click

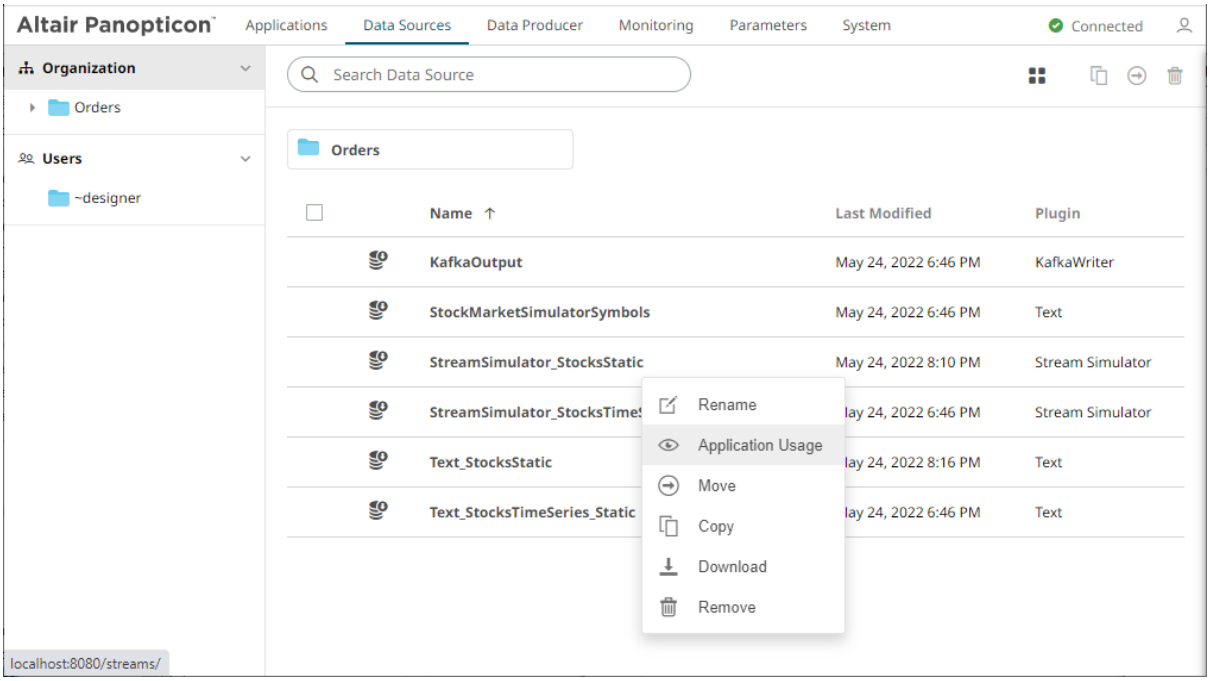
.

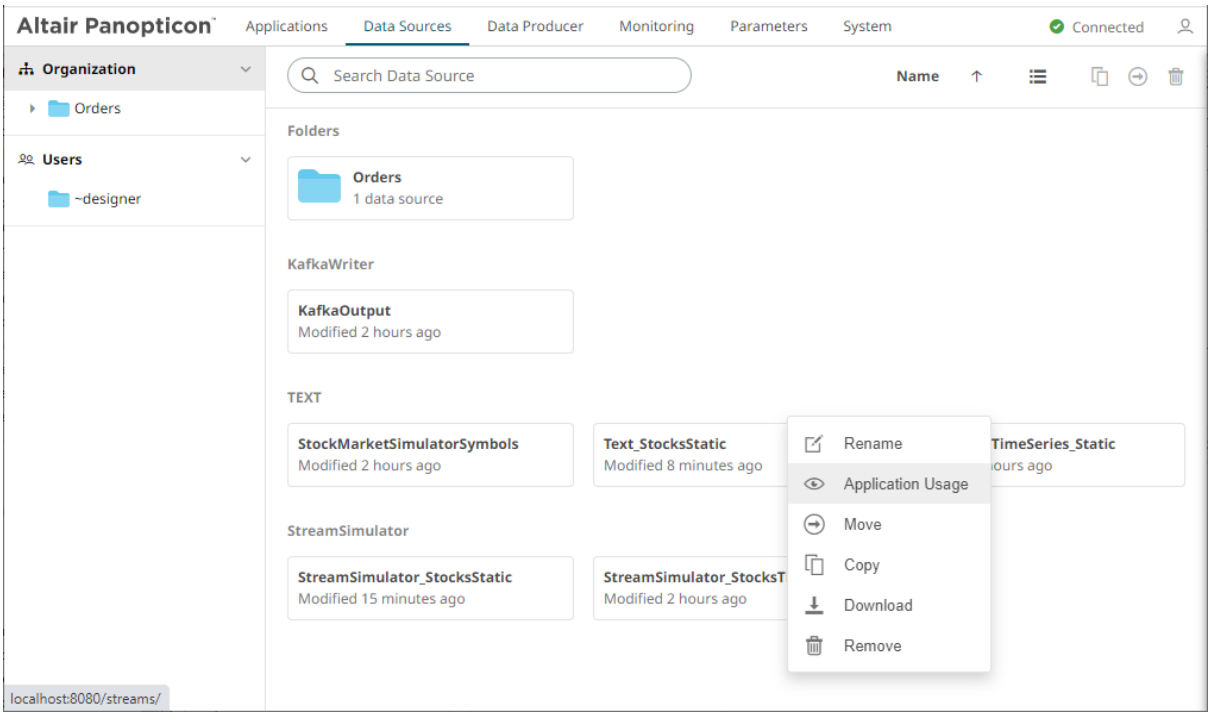
VIEWING APPLICATION USAGES

On the **Data Sources** tab, you can view the applications that currently use a data source.

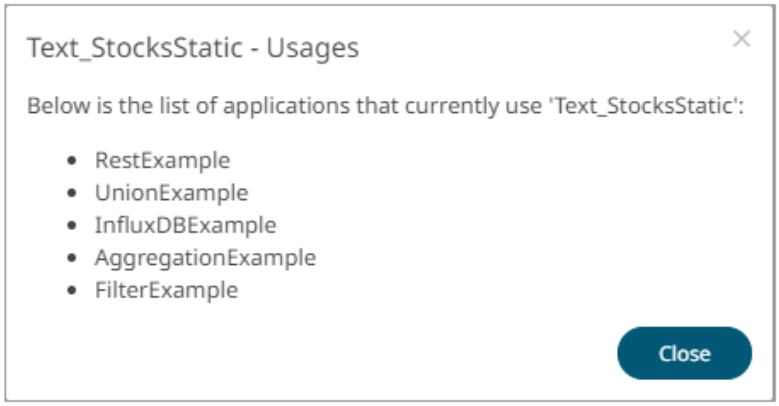
Steps:

1. On the *List* view or *Grid* view, right-click on a data source and select **Application Usage**.





The list of applications that currently use the data source displays.



3. Click

.

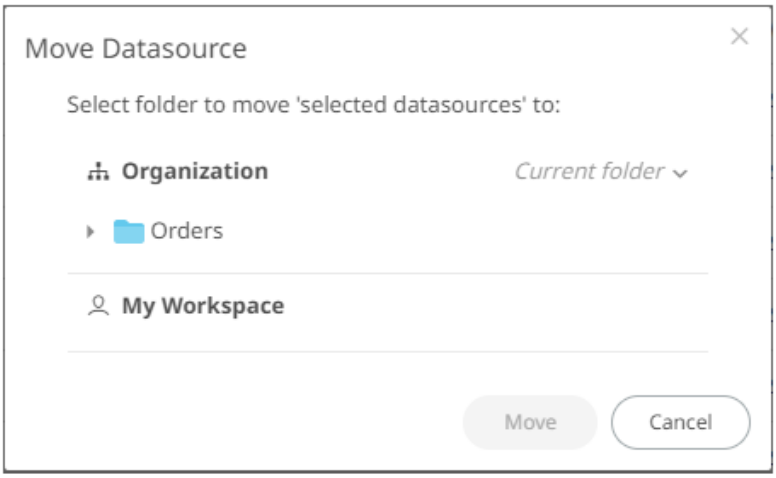
MOVING DATA SOURCES

Users with a Designer role can move data sources to folders or subfolders to which they have permission.

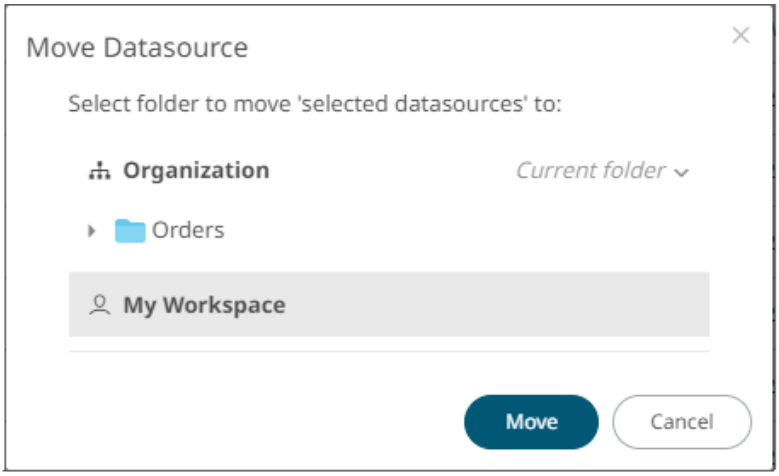
Steps:

1. Check the box of multiple data sources either on the *Grid View* or the *List View*.
2. Then select either:
   - the **Move** icon on the toolbar
   - **Move** on the context menu

   The *Move Data Source* dialog displays with the folder or subfolders to which the user has permission to move the data sources.



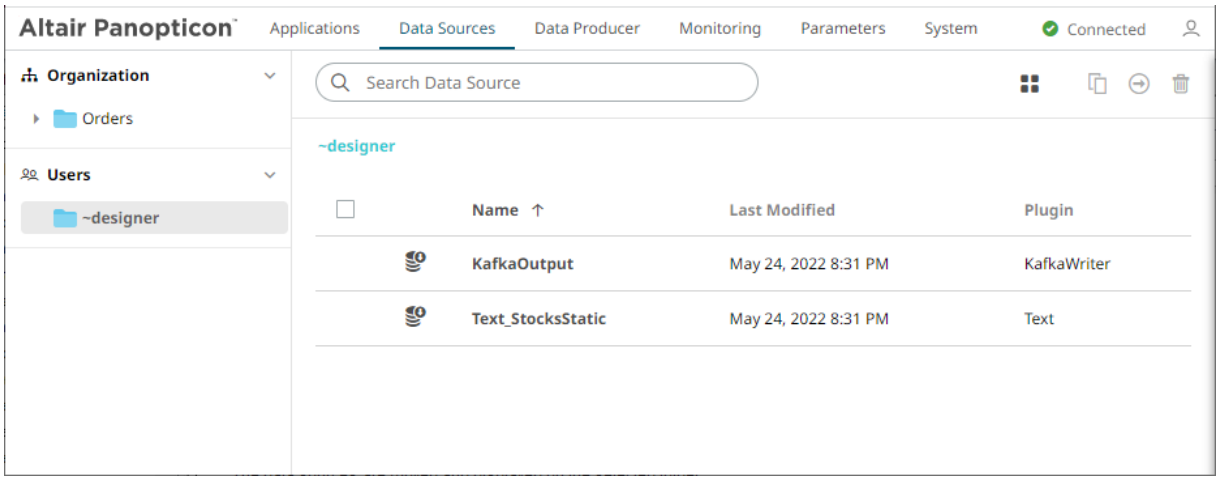
3. Select the folder or subfolder.



4. Click

.

   The data sources are moved and displayed on the selected folder.



NOTE **If data sources with the same name are already in the selected folder, a copy of them will be moved.**

COPYING DATA SOURCES

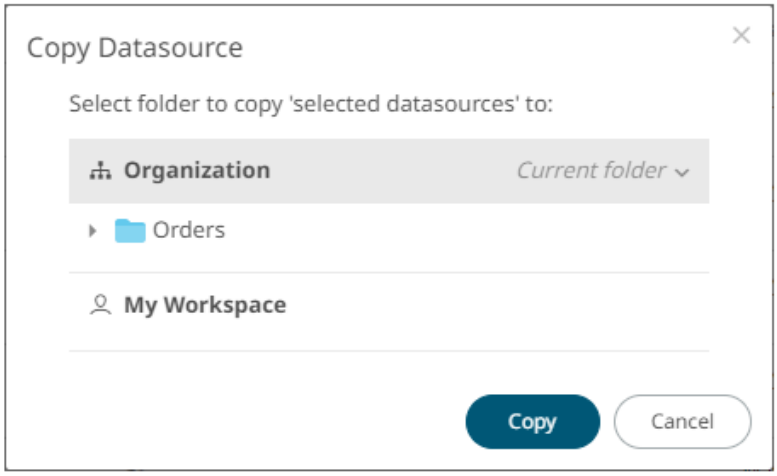
Users with a Designer role can copy data sources to folders or subfolders to which they have permission.

Steps:

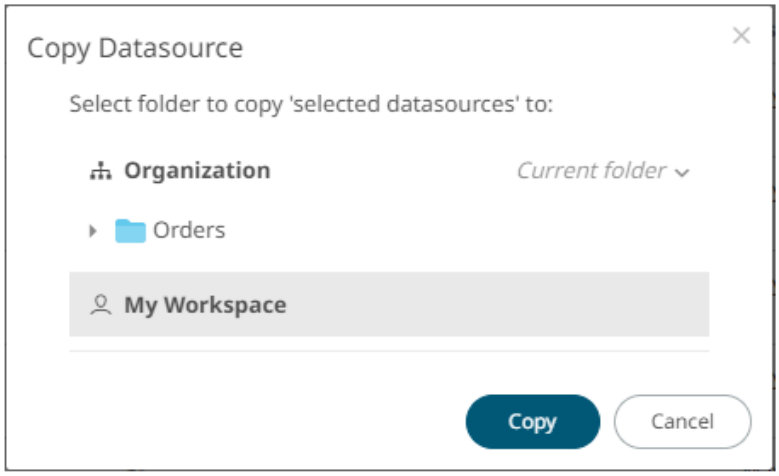
1. Check the box of one or multiple data sources either on the *Grid View* or *List View*.
2. Then select either:

- the **Copy** icon on the toolbar
- **Copy** on the context menu

The *Copy Data Source* dialog displays with the folder or subfolders to which the user has permission to copy the data sources.



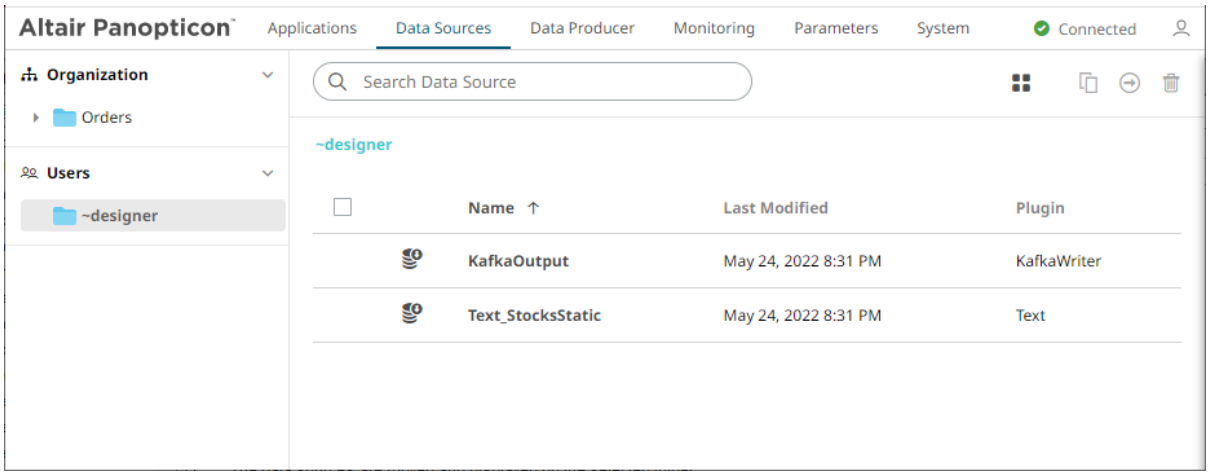
3. Select the folder or subfolder.



4. Click

.

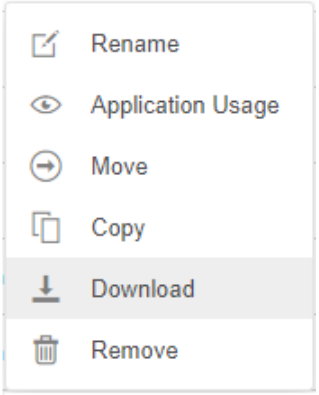
   The data sources are copied and displayed on the selected folder.



NOTE **If data sources with the same name are already in the selected folder, it will be added as copy.**

DOWNLOADING A DATA SOURCE

Users with Administrator or Designer role are allowed to download a copy of a data source by right-clicking on a data source and selecting **Download** on the context menu.



The data source is downloaded.

DELETING DATA SOURCES

Users with an Administrator or Designer role can delete data sources using the toolbar or context menu.

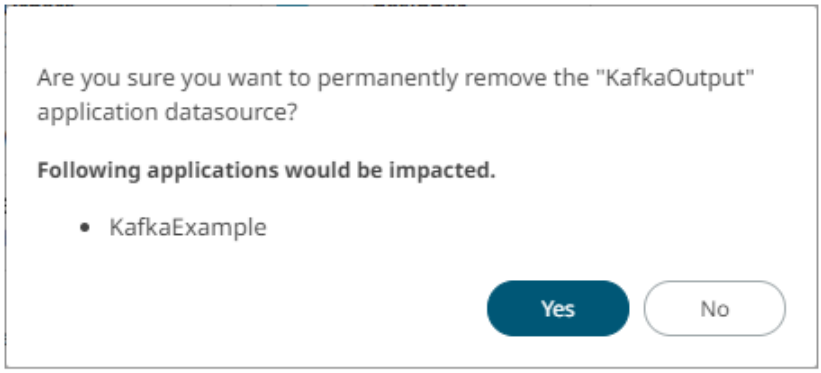
Steps:

1. Check the box of one or several data sources either on the *Grid View* or *List View*.
2. Then click either:

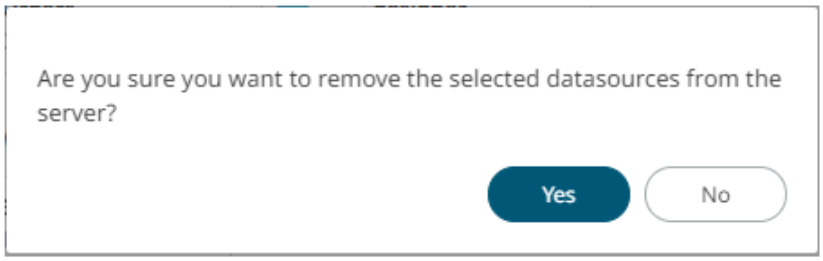
- the **Remove** icon on the toolbar
- **Remove** on the context menu

A notification message displays.

For a data source, the corresponding applications that will be impacted is listed:



For several data sources:



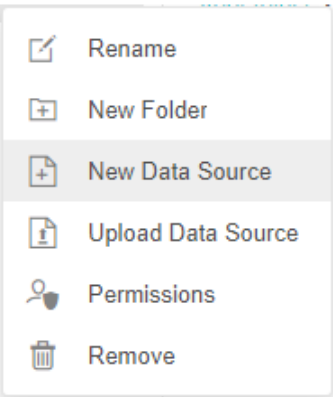
3. Click Yes to remove.

CREATING A DATA SOURCE

Panopticon Streams supports creation of data sources that can be used as inputs or outputs in the application model.

Steps:

1. On the **Data Sources** tab:
   - click + New Data Source on the toolbar, or
   - right-click on a folder or subfolder and select **New Data Source.**

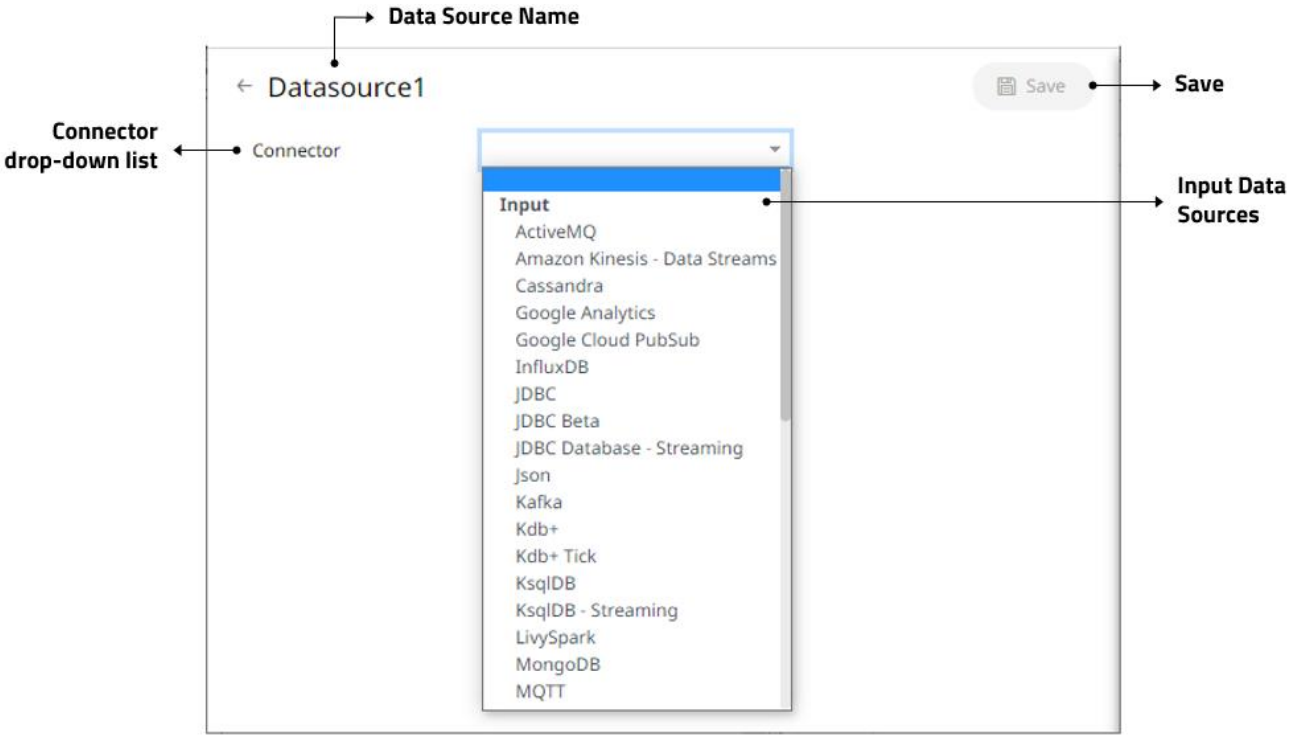


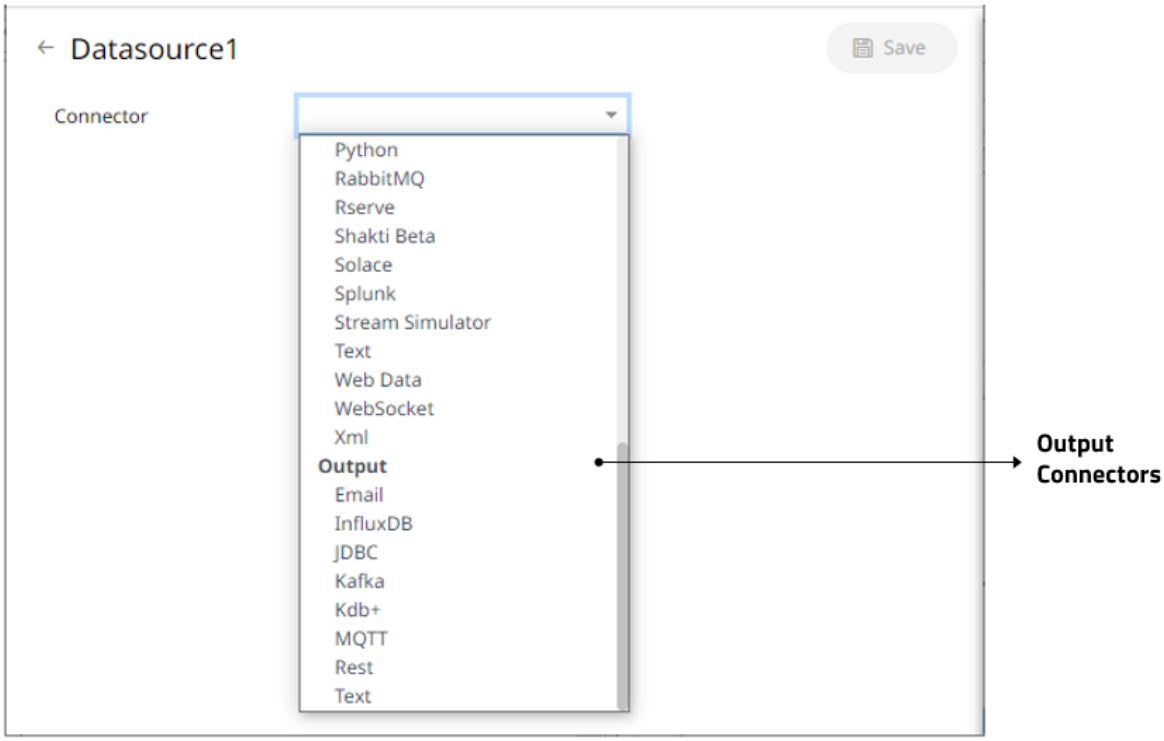
The *New Data Source* dialog displays.

2. Enter the *Name* of the data source and click

.

   The **Data Source** tab displays with the following sections:





Section/Pane	Description
Data Source Name	Name of the data source. Click the ← button to go back to the *Data Sources* listing page.
Connector drop-down list	Includes the input data sources and output connectors.
Save	Saves the changes made on the **Data Sources** tab.

NOTE **Inactive connectors are not displayed in the *Connector* drop-down.**

3. Enter the *Name* of the data source. This should be unique and should only contain letters (a to Z), numbers (0 to 9), and underscores.

4. Click ✓ or press **Enter** to apply the name.

5. Select any of the following:

   - output connectors
     - Email
     - InfluxDB
     - JDBC Database
     - Apache Kafka
     - Kx kdb+

    - MQTT
    - REST
    - Text
- Input data sources
    - ActiveMQ
    - Amazon Kinesis – Data Streams
    - Cassandra
    - Elasticsearch 6.x
    - Elasticsearch 7.x
    - Google Analytics
    - Google Cloud Pub/Sub
    - InfluxDB
    - JDBC Databases
    - JDBC (Beta)
    - JDBC Database - Streaming
    - JSON
    - Apache Kafka
    - Kx kdb+
    - Kx kdb+ Tick
    - ksqlDB
    - ksqlDB - Streaming
    - Livy Spark
    - MongoDB
    - MQTT
    - MS Excel (legacy)
    - MS Excel (xlsx)
    - OneTick
    - OneTick CEP
    - OneTick Cloud
    - Python
    - RabbitMQ
    - Rserve
    - Shakti (Beta)
    - Solace
    - Splunk

- Stream Simulator
- StreamBase 7.1
- StreamBase LiveView
- Text
- WebSocket
- Web Data
- XML

The tab page changes depending on the selected connector. Refer to the sections below for more information.

Common Data Source Settings

Some of the data sources share the following settings or parts:

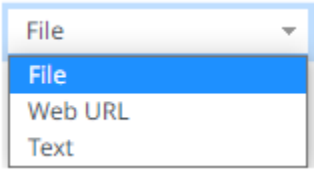
- Data Connector File Source
- Message Type selection and definition
- Saving and loading of column definitions
- Data Souce Toolbar
- Defining Real-time Settings

Selecting and Defining the Data Connector File Source

Several connectors including JSON, MS Excel (legacy), Text, XML, and Stream Simulator, allow selection from a File, Web URL, or Text source.

Steps:

Select the connector file source:

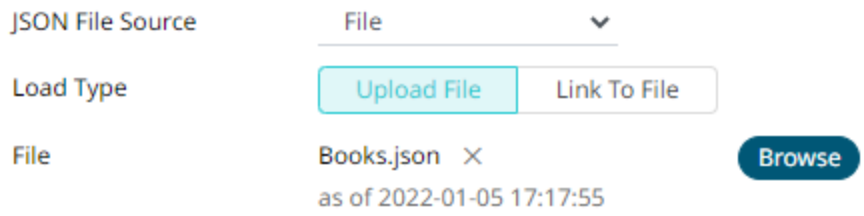


- File

  You can either:

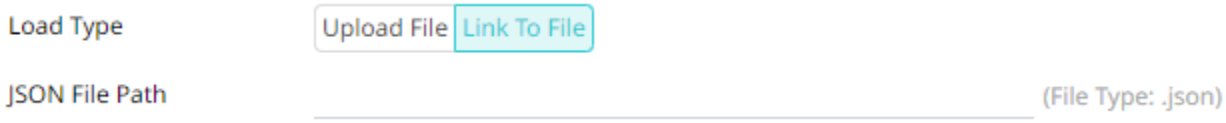
  - Upload a data source snapshot by clicking **Upload File** then **Browse** to browse to the file source.

    After selecting the file, it is displayed with the timestamp of the snapshot.



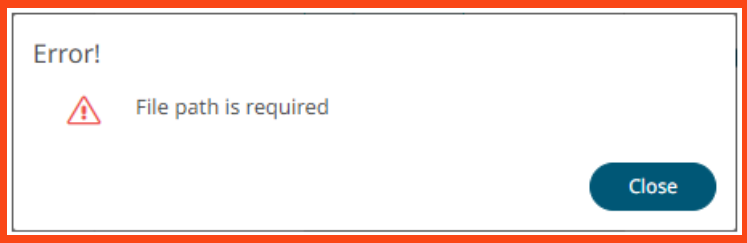
To change the data source, click ✕ then **Browse** Browse to browse to a new file.

- Link to a data source file by clicking **Link to File** Link To File and entering a *File Path*.



Ensure that in a cluster, you need to use a a shared path, or put it on every node and use a path that resolves on every node. You can update its contents whenever you want.

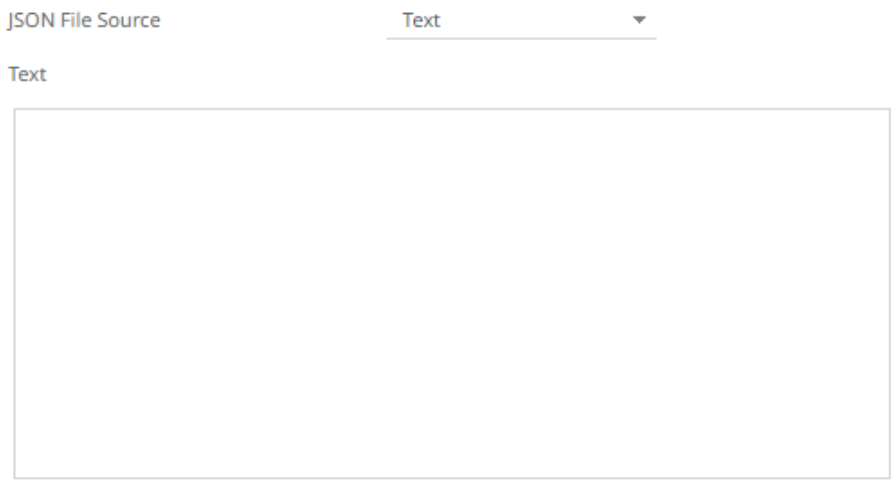
NOTE **An error notification displays upon fetching sheets/column generation when the *File Path* is blank.**



Click Close and enter the file path.

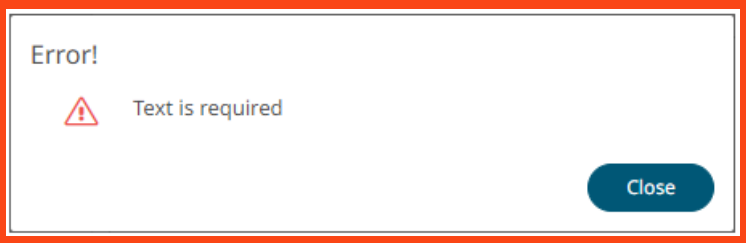
- Text

  Then enter the text block to be parsed.



NOTE

- **The Text file source is not available for the MS Excel connectors.**
- **An error notification displays upon column generation when the *Text* box is blank.**



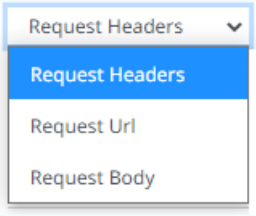
Click Close and enter the text block.

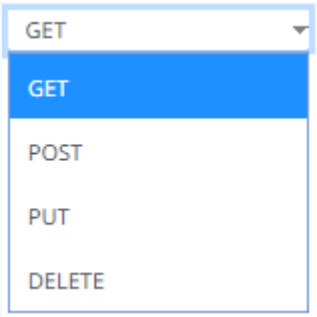
- Web URL

  The dialog changes to allow specification of the following:

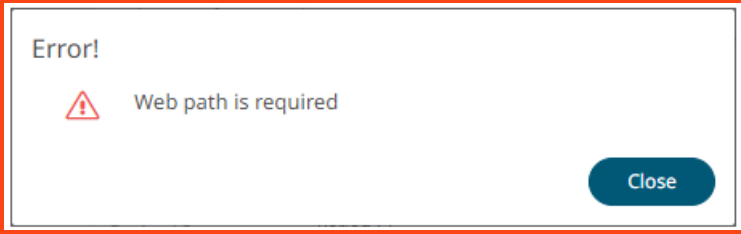
JSON File Source	Web URL
Authentication Type	Basic
Path	
Proxy Server URI	
Headers	
Content Encoding	None
User Id	
Password	Show characters
Http Method	GET
Timeout	10
Request Body	
Content Type	application/x-www-form-urlencoded

Property	Description
Authentication Type	• **Basic** The basic authentication. • **OAuth** Authentication Type: OAuth Token Url Token Request Body Add Access Token To: Request Headers Url Then enter the following settings: ○ **Token URL** – The URL to retrieve the access token from. ○ **Token Request Body** – The request body used for access token requests.

	○ **Add Access Token To** - The Access token retrieved from the *Token URL* can be added to headers, URL or request body, depending on how the endpoint needs the token.  ▪ Request Header - A header is automatically added to the REST API request. ▪ Request URL - The URL needs to be manually parameterised with a {access_token} parameter, before calling the REST API, the parameter is replaced with the actual token. ▪ Request Body - The Request Body needs to be manually parameterised with a {access_token} parameter, before calling the REST API, the parameter is replaced with the actual token. **NOTES:** • The given request body is posted to the Token URL as **application/x-www-form-urlencoded**, so the request body must be formatted like **field1=value1&field2=value2**, e.g., **client_id=xxxx&client_secret=xxxx&grant_type=client_credentials**. • Not available in the Stream Simulator connector.
Path	The absolute path including the http where the file is located.
Proxy Server URI	The HTTP Proxy setting that will allow the connector to reach the endpoint.
Headers	• Headers are separated by a comma • Each Header is entered as **Name = Value**, where *Name* and *Value* can be enclosed in double quotes to allow inclusion of any character except for double quotes • *Name* and *Value* can also be left unquoted, in which case they may not include comma or equals characters
Content Encoding	Select the *Content Encoding* with the HTTP Header: **None, GZip, Deflate,** or **GZip and Deflate**
User Id	The user Id that will be used to connect to the connector's service.
Password	The password to connect to the connector's service. Check the **Show Characters** box to display the entered characters.
HTTP Method	Select the appropriate HTTP method for the request from the following options:

	 • GET – retrieve data • POST – add new data • PUT – replace existing data • DELETE – remove existing data
Timeout	The length of time to wait for the server response (10 to 300). Default is **10**.
Request Body	The Request Body for the HTTP POST.
Content Type	The required Content Type. Default is **application/x-www-form-urlencoded**
Record Path	The record path that will be queried by the connector's path (e.g., **myroot.items.item**).

NOTE **An error notification displays upon column generation when the *Path* is blank.**



Click Close and enter the web path.

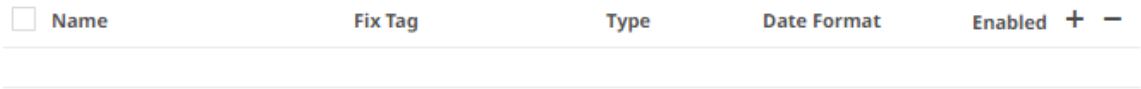
Defining the Message Type in Data Sources

Message types specify the format of the data within the message.

Steps:

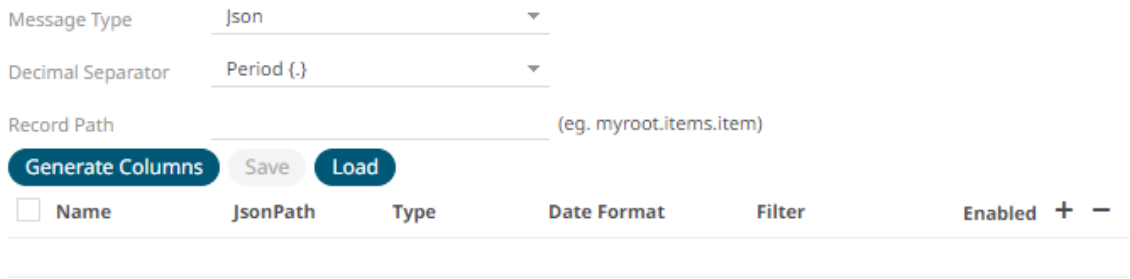
1. Select the *Message Type*:

- FIX



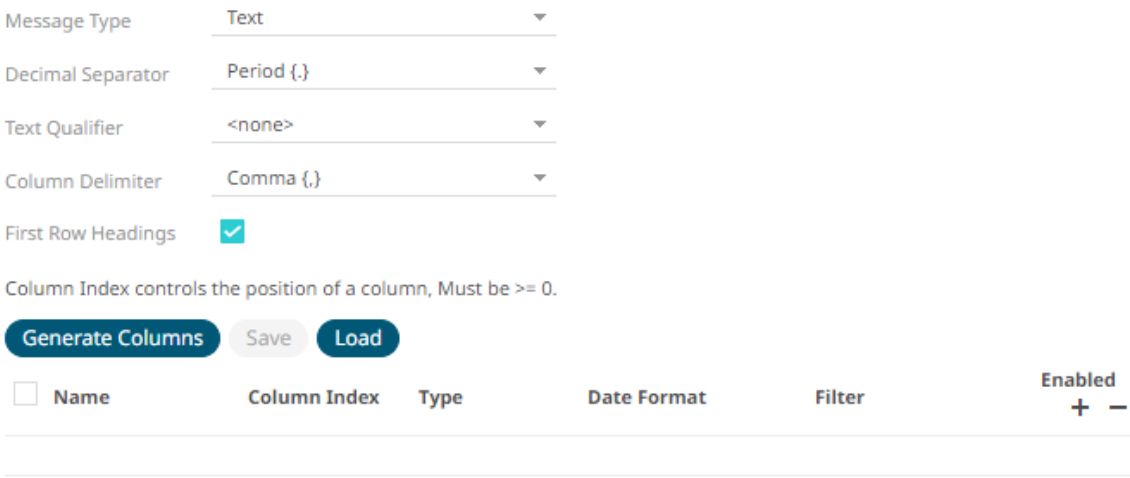
- JSON

  If **JSON** is selected, enter the *Record Path* which allows the identification of multiple records within the JSON document (e.g., **myroot.items.item**).



- Text

  If **Text** has been selected, confirm the **Decimal Separator, Text Qualifier**, **Column Delimiter**, and if the first row of the message includes column headings.



- XML

Name	XPath	Type	Date Format	Filter	Enabled + −

2. Define or set the columns that represent the sections of the message.

Property	Description
Name	The column name of the source schema.
Fix Tag/JsonPath/Text Column Index/XPath	The Fix Tag/JsonPath/Text Column Index/XPath of the source schema.
Type	The data type of the column. Can be a **Text, Numeric,** or **Time**
Date Format	The format when the data type is **Time**.
Enabled	Determines whether the message field should be processed.

NOTE **To parse and format times with higher than millisecond precision, the format string needs to end with a period followed by sequence of upper case S. There can be no additional characters following them.**

For example: yyyy-MM-dd HH:mm:ss.SSSSSS

To delete a column, check its ☐ or all the column entries, check the topmost ☐, then click −.

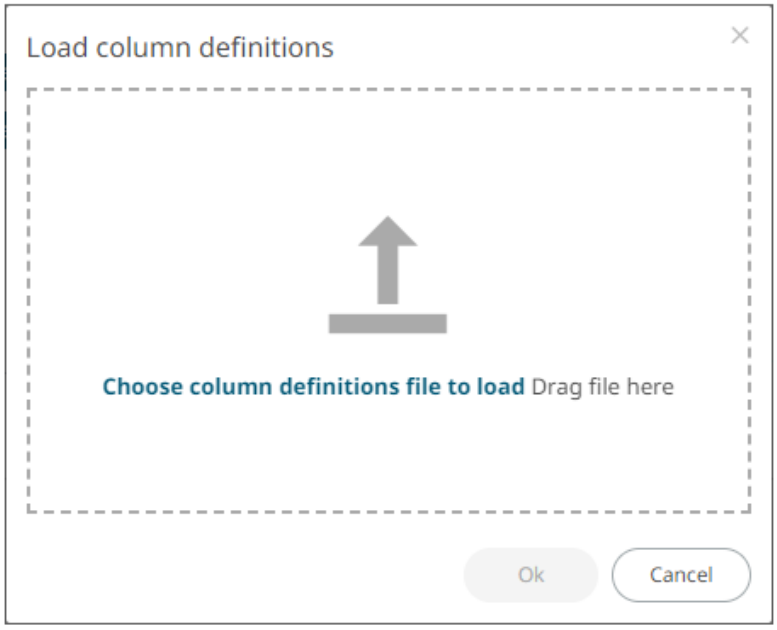
Saving or Loading Column Definitions in the Data Sources

Save or load column definitions in the data sources.

Steps:

1. Click Save to save a copy of a column definitions file (**.exs**).
2. Instead of generating columns done in step 8, click Load to load a column definitions (.exs) file.

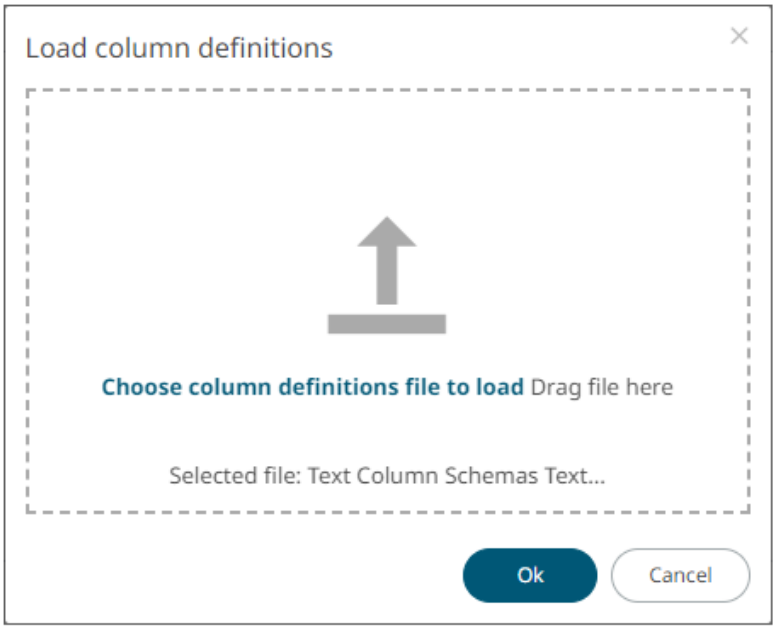
   The *Load Column Definitions* dialog displays.



2.1. To load column definitions, you can either:

- drag it from your desktop and drop in the dialog, or
- click **Choose Column Definitions File to Load** and select one in the *Open* dialog that displays.

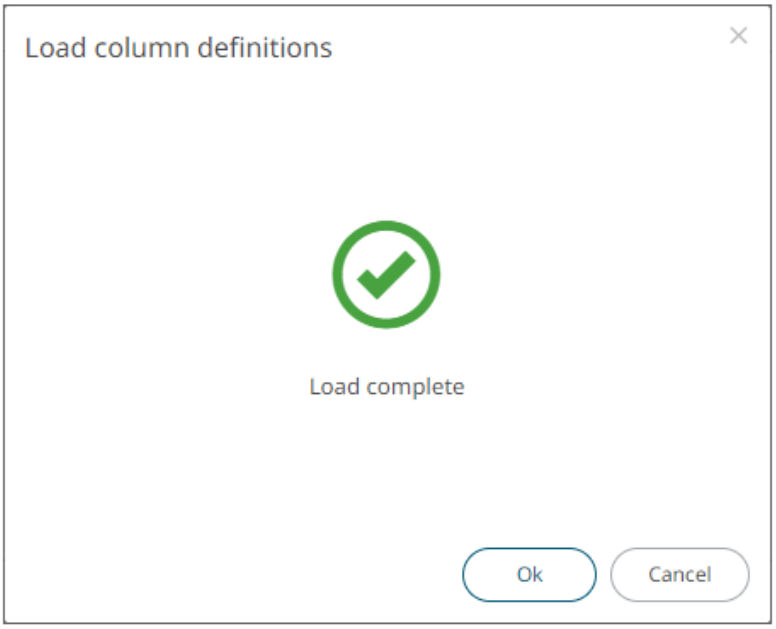
  The name of the column definitions is displayed on the loaded column definitions area.



2.2. Click

.

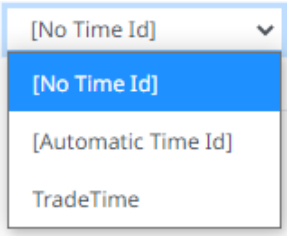
A notification displays when the file is loaded.



This populates the list of columns from the .exs file.

Defining Real-time Settings

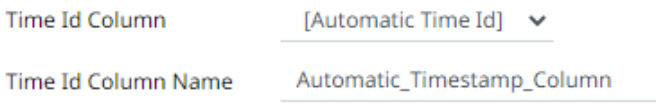
Streaming connectors have a common section to specify the *Time Id Column* to generate the streaming time series window. The *Time Id Column* can be from the soure dataset, or alternatively, automatically generated.



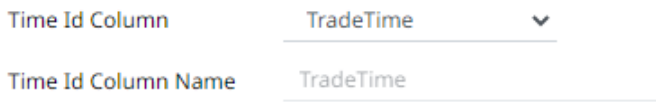
As new data arrives from the subscription, new time slices will automatically be added, and old ones will be deleted.

Steps:

1. Select either:
   - Automatic Time Id

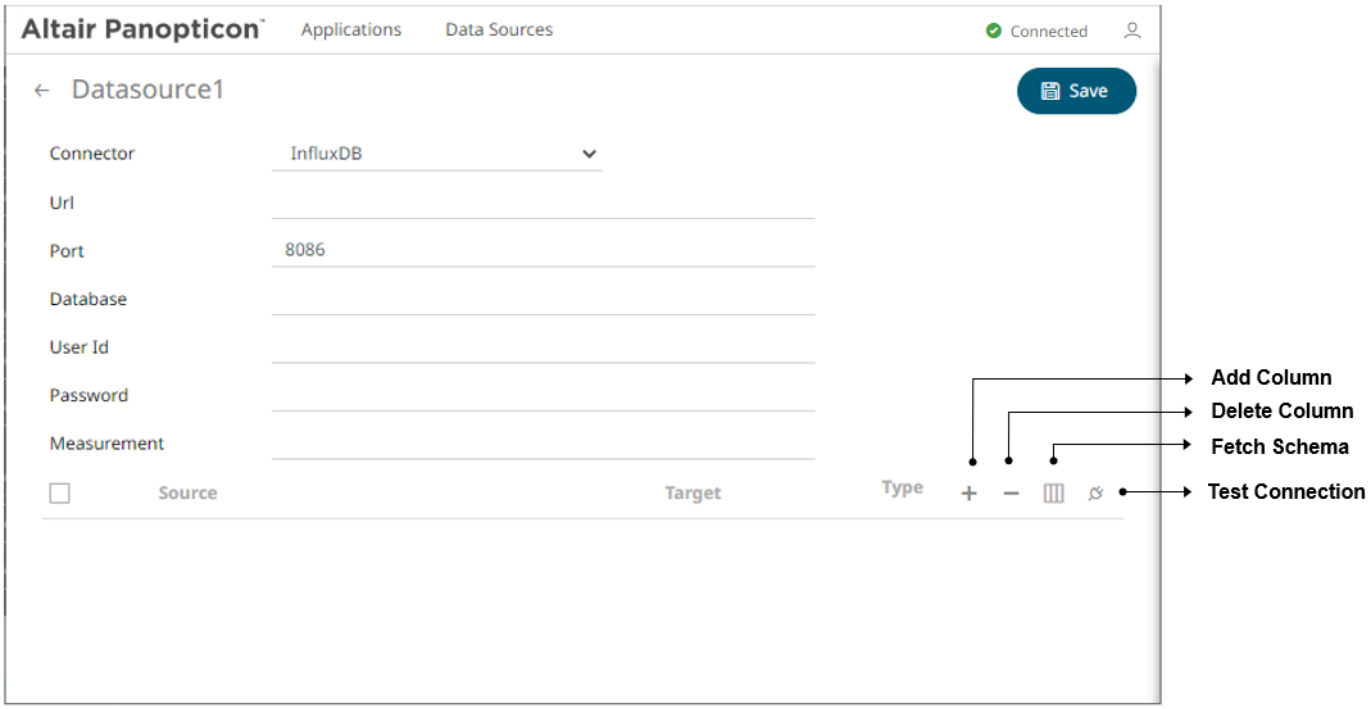
     Then define the *Time Id Column Name*.
   - Date/Time Id column either from the source data or automatically generated

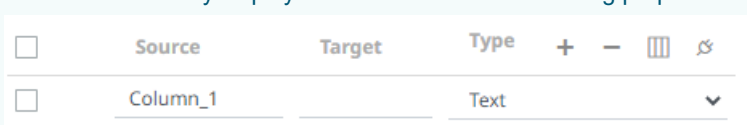
2. Check the **Reset Data on Reconnect** box to flush out the stale data and reload data after reconnection.

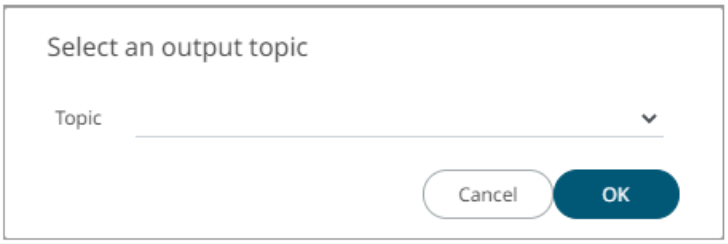
Using the Data Source Toolbar

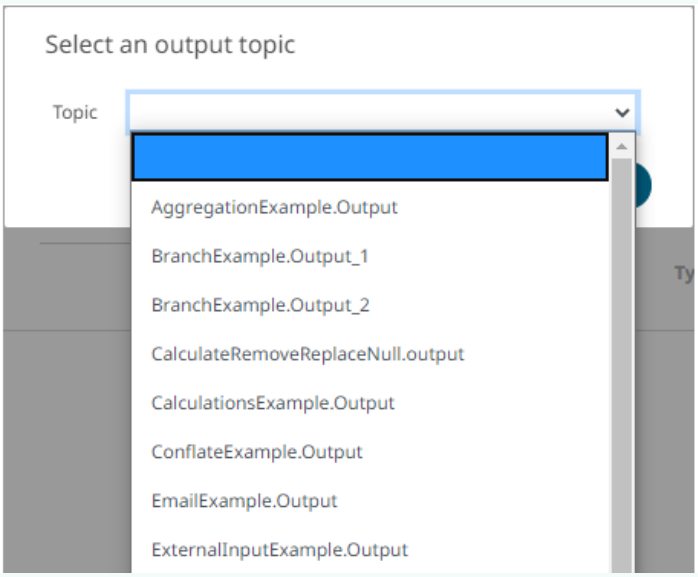
Several data sources have a toolbar:



Click on any of the following icons:

Icon	Description
+	A new column entry displays. Enter or select the following properties:  • Source The column name of the source schema. • Target The column name of the target schema. • Type The data type of the column. Can be a **Text**, **Numeric**, or **Time.**
☐	Check the topmost ☐ to select all data source column entries.
−	Check the ☐ of a data source entry or check the topmost ☐ to select all column entries and click − to delete.
▥	Allows you to select an output topic in the drop-down list.





Click **OK**. The schema of the selected output topic is displayed.

Tests if the connection to the output connector is successful.
If successful, a confirmation message displays.

Click **OK**.
Otherwise, an error prompt displays.

Click **Close** and fix the connection error.

Date/Time Key Elements

The key elements of the Date/Time format include:

Component	Format
Year	yyyy
Month	MM
Month as an abbreviation	MMM
Day	dd
Hour (24-hour clock)	HH
Minute	mm
Second	ss
Hour (12-hour clock; a.m./p.m.)	tt
Millisecond	SSS
Microsecond	SSSSSS
Nanosecond	SSSSSSSSS
Space/separator (required if time is specified)	'T'
Zulu (Greenwich Mean Time)	'Z'
Time zone (ISO 8601 time zone)	X
UNIX Epoch time	POSIX
Milliseconds since UNIX Epoch time	POSIXMILLIS
Seconds since midnight	Seconds
Milliseconds since midnight	Millis
Microseconds since midnight	Micros
Nanoseconds since midnight	Nanos

NOTE

- **To parse and format times with higher than millisecond precision, the format string needs to end with a period followed by sequence of upper case S. There can be no additional characters following them.**

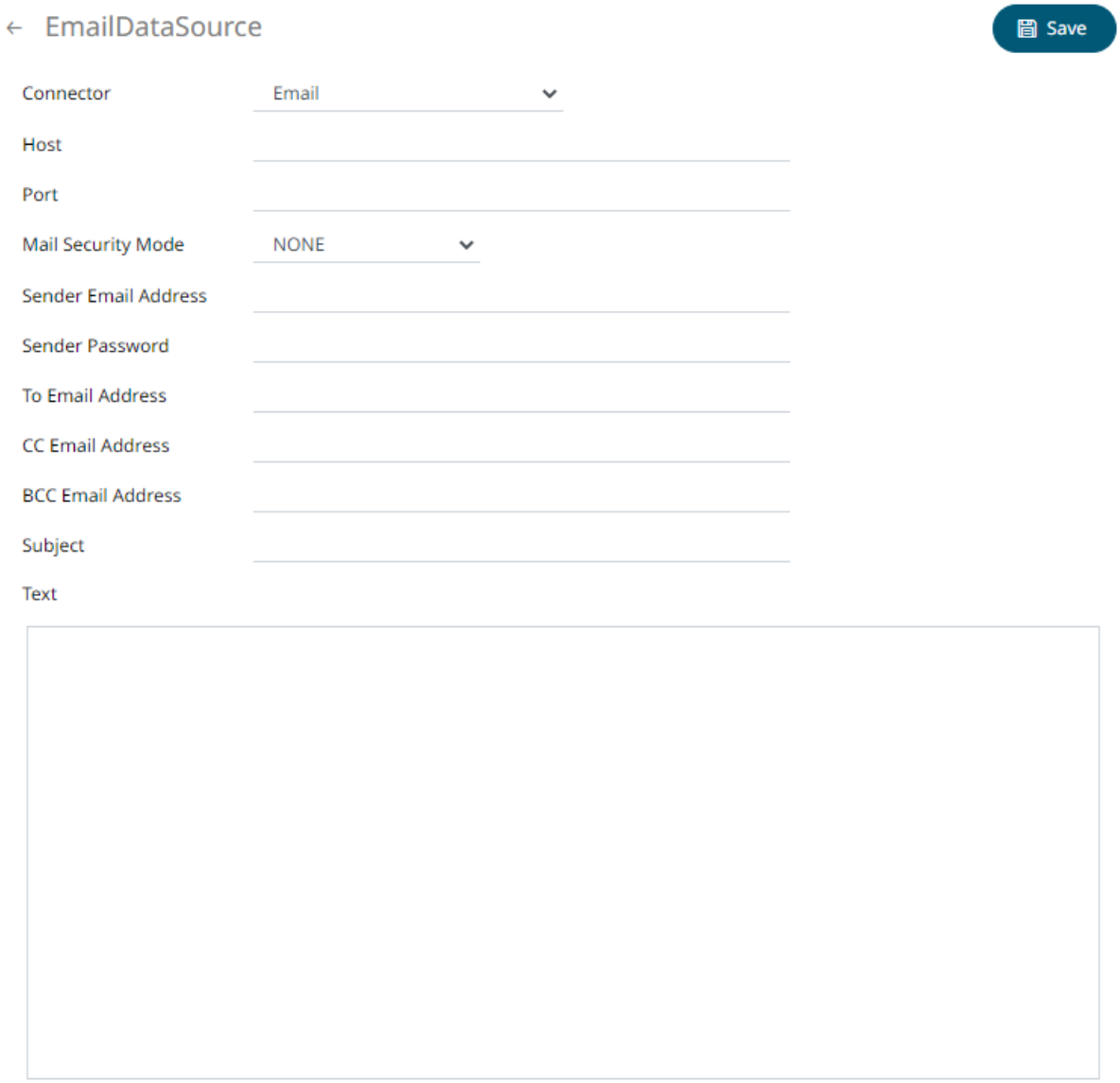
  For example: yyyy-MM-dd HH:mm:ss.SSSSSS

- **The “Seconds”, “Millis”, “Micros”, and “Nanos” formats are used for parsing of the data in the data connectors and not for the display of the Date/Time columns.**

Creating Email Output Connector

Steps:

1. On the **Data Source** tab, select **Output > Email** in the *Connector* drop-down list.



2. Define or select the following properties:

Property	Description
Host	Email host address.
Port	Email host port.
Mail Security Mode	Select the email security mode: **NONE**, **SSL**, or **TLS**
Sender Email Address	Email address of the sender.
Sender Password	Password of the sender.

To Email Address	Email address of the recipient.
CC Email Address	Email address of the CC recipient.
BCC Email Address	Email address of the BCC recipient.
Subject	Subject of the email.
Text	Content of the email.

3. Click

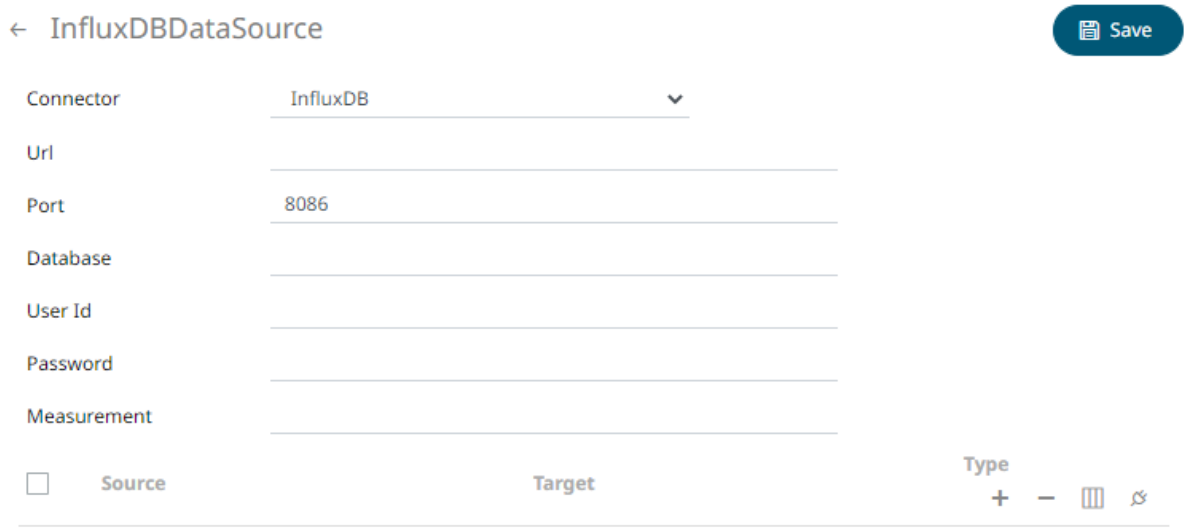
. The new data source is added in the *Data Sources* list.

Creating InfluxDB Output Connector

Allows periodical dumping of data from a Kafka topic into a time series database such as InfluxDB.

Steps:

1. On the **Data Source** tab, select **Output > InfluxDB** in the *Connector* drop-down list.



2. Define or select the following properties:

Property	Description
URL	URL of the InfluxDB.
Port	The port running the InfluxDB HTTP service. Default is **8086**.
Database	The name of the database that will be communicate over the HTTP(S).
User Id	The user Id that will be used to connect to InfluxDB.
Password	The password that will be used to connect to InfluxDB.
Measurement	The table name that can be used as measurement.

3. You may opt to use the toolbar to complete the data source definition.

4. Click

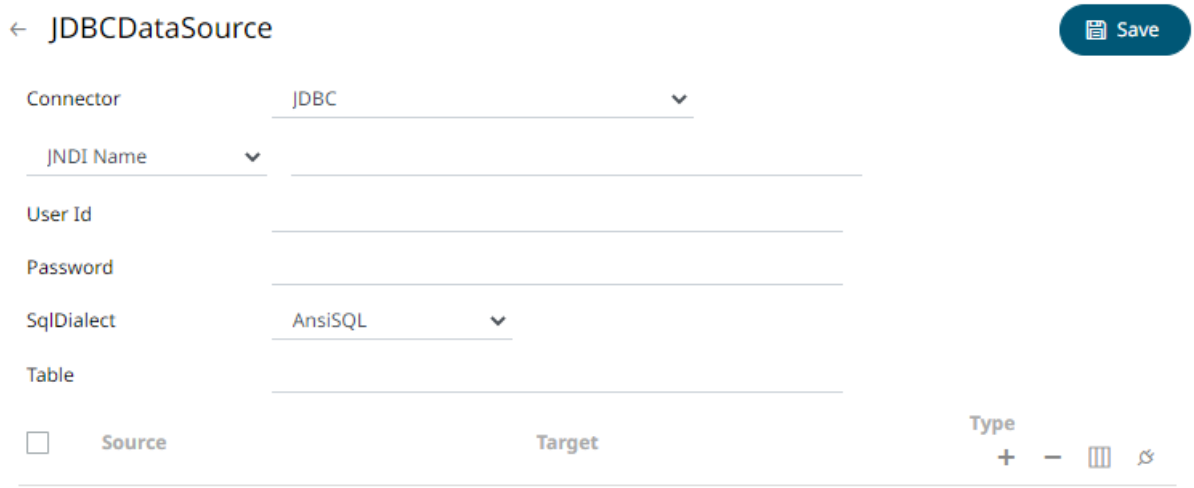
. The new data source is added in the *Data Sources* list.

Creating JDBC Database Output Connector

Allows periodical dumping of records from a Kafka topic into a JDBC database.

Steps:

1. On the **Data Source** tab, select **Output > JDBC** in the *Connector* drop-down list.



2. You can either select:
   - JNDI Name

     Enter the *JNDI resource name* to be used, then the *User Id* and *Password*.

     NOTE **The JNDI resource name needs to be on the form:**

     `java:/comp/env/jdbc/[resourcename]`

   - URL

     Enter the *URL* specific to the database's JDBC driver, the *Driver Class Name* specific to the driver, and the *User Id* and *Password*.

3. Select the appropriate *SQL Dialect* in the drop-down list to be able to generate the correct SQL for the required data repository.

   You can select any of the following *SQL dialects*: AnsiSQL, Access/Excel, MySQL, Oracle, SQL Server, Sybase IQ/ASA, Sybase ASE, Netezza, Vertica, SQLite, HadoopHive, KxQ, DB2, PostgreSQL, Impala, Redshift, Informix, Teradata, dBase, SparkSQL.

4. Enter the source *Table* (can be parameterized).

5. You may opt to use the toolbar to complete the data source definition.

6. Click

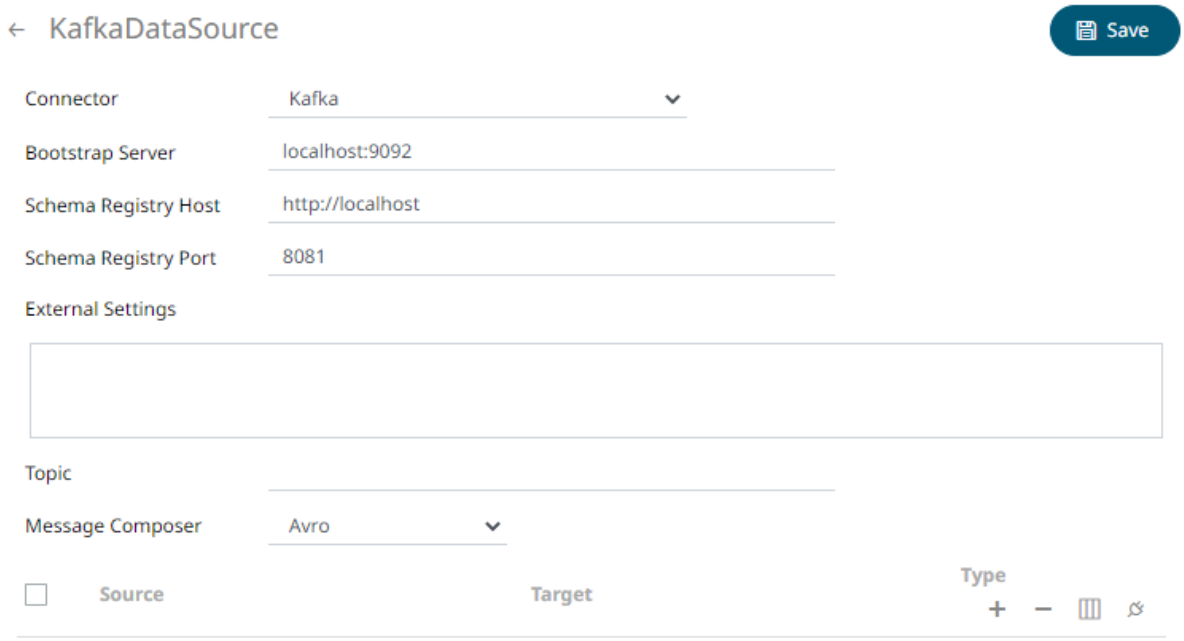
. The new data source is added in the *Data Sources* list.

Creating Apache Kafka Output Connector

Allows publishing of events to an external Kafka JSON or Avro topic. For Avro, ensure to point towards the schema registry used by the external Kafka cluster.

Steps:

1. On the **Data Source** tab, select **Output > Kafka** in the *Connector* drop-down list.



2. Enter or select the following properties:

Property	Description
Bootstrap Server	List of host/port pairs of Kafka servers used to bootstrap connections to a Kafka cluster. By default, the value is `localhost:9092`. However, this can be overridden by specifying another bootstrap server in the *External Settings* text box (as specified in step 3).
Schema Registry Host	Where the Schema Registry is located. This can be in a different location from the Kafka cluster.
Schema Registry Port	The port number of the schema registry which provides the serving layer for the metadata. Default is **8081**.

3. Enter the *External Settings* to support authentication (i.e., username and password). Note that if the bootstrap server is not secure, then there is no need to authenticate and you may leave this text box blank.

   Below is an example of system settings for an SASL authentication:

```
bootstrap.servers=localhost:9093
sasl.jaas.config=\
  org.apache.kafka.common.security.plain.PlainLoginModule required \
```

4. Enter the *Topic* name.
5. Select the Message Composer: **Avro** or **JSON.**
6. You may opt to use the toolbar to complete the data source definition.
7. Click

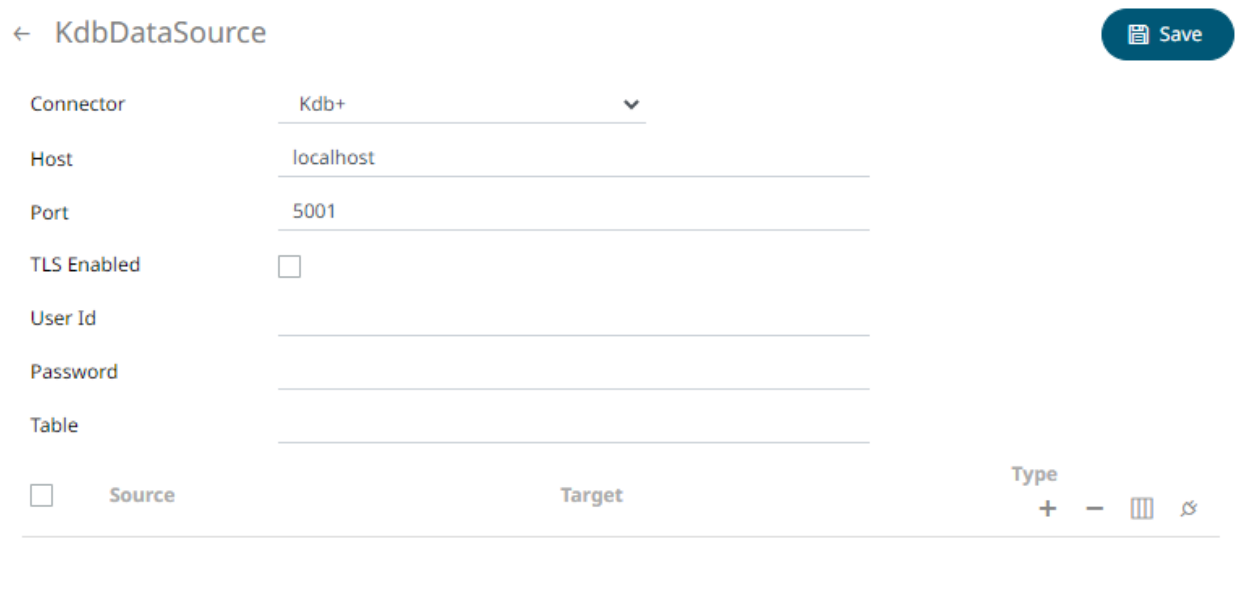
. The new data source is added in the *Data Sources* list.

Creating Kx kdb+ Output Connector

Allows periodical dumping of records from a Kafka topic into a Kx kdb+ connector.

Steps:

1. On the **Data Source** tab, select **Output > Kdb+** in the *Connector* drop-down list.



2. Define or select the following properties:

Property	Description
Host	Kx kdb+ host address.
Port	Kx kdb+ host port. Default is **5001**.
TLS Enabled	Ensure to check if you have started q with TLS only.
User Id	The user Id that will be used to connect to Kx kdb+.
Password	The password that will be used to connect to Kx kdb+.
Table	The source table.

NOTE **These properties can be parameterized.**

3. You may opt to use the toolbar to complete the data source definition.
4. Click

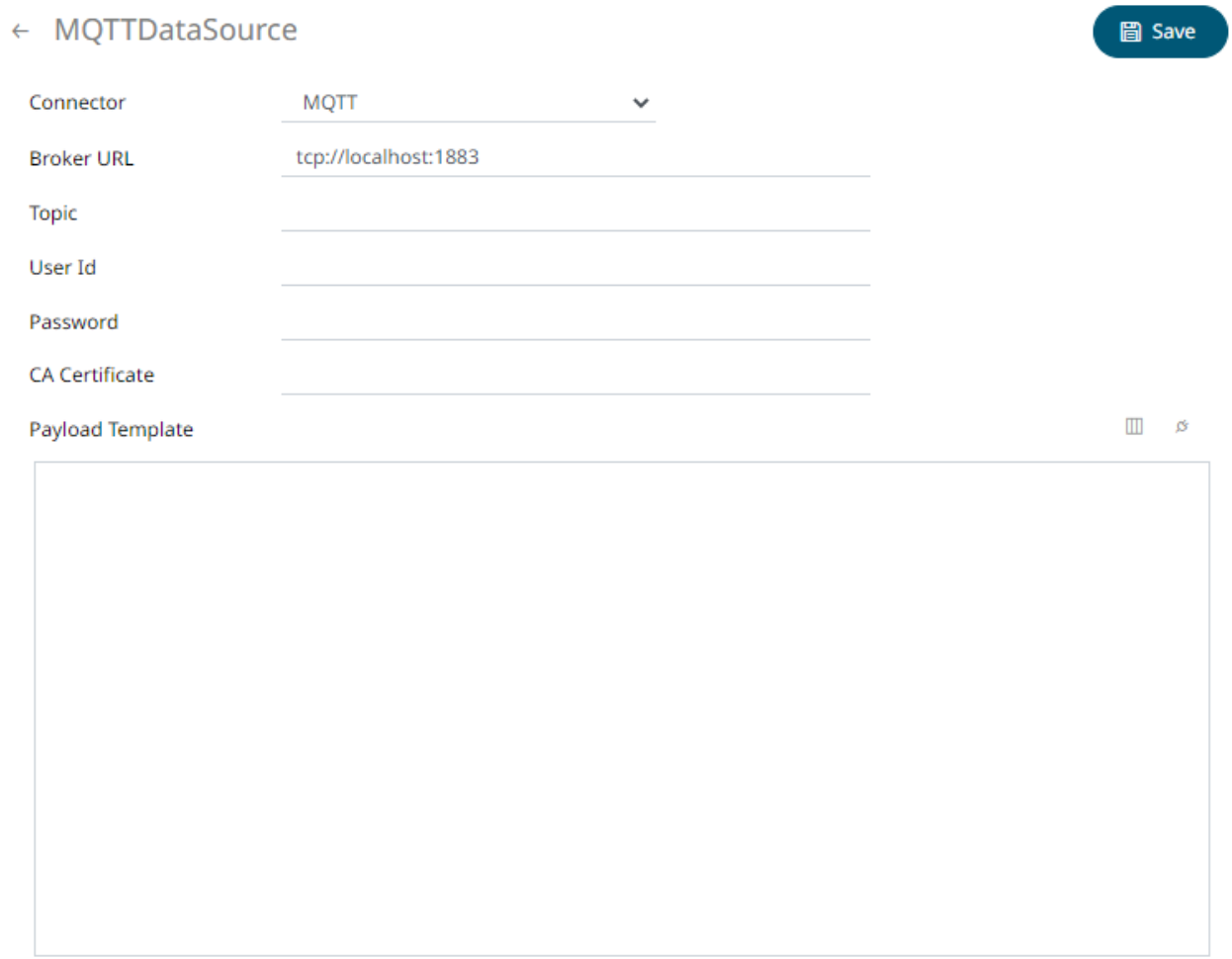
. The new data source is added in the *Data Sources* list.

Creating a MQTT Output Connector

Allows publishing of data to external MQTT topic.

Steps:

1. On the **Data Source** tab, select **Output > MQTT** in the *Connector* drop-down list.



2. Define the following properties:

Property	Description

Broker URL	The location of the message broker. Default is `tcp://localhost:1883`
Topic	The topic or the queue physical name. Can be parameterized. Example: level1/level2/level3/level4 etc. **NOTES:** You can also opt to use a wild card in the topic name specification. • The plus sign symbol (+) can be used as a wild card for any value at one specific level. Example: **level1/level2/+/level4** • The hash sign symbol (#) can be used as a wild card for any values across more than one level. Example: **level1/#/level4**
User Id	The user Id that will be used to connect to MQTT.
Password	The password that will be used to connect to MQTT.
Payload Template	The template that will be rendered to generate the payload. Can be parameterized with output schema columns.

3. To allow the encrypted connections, enter the path to the *CA Certificate* file.
4. You may opt to use the toolbar to complete the data source definition.
5. Click

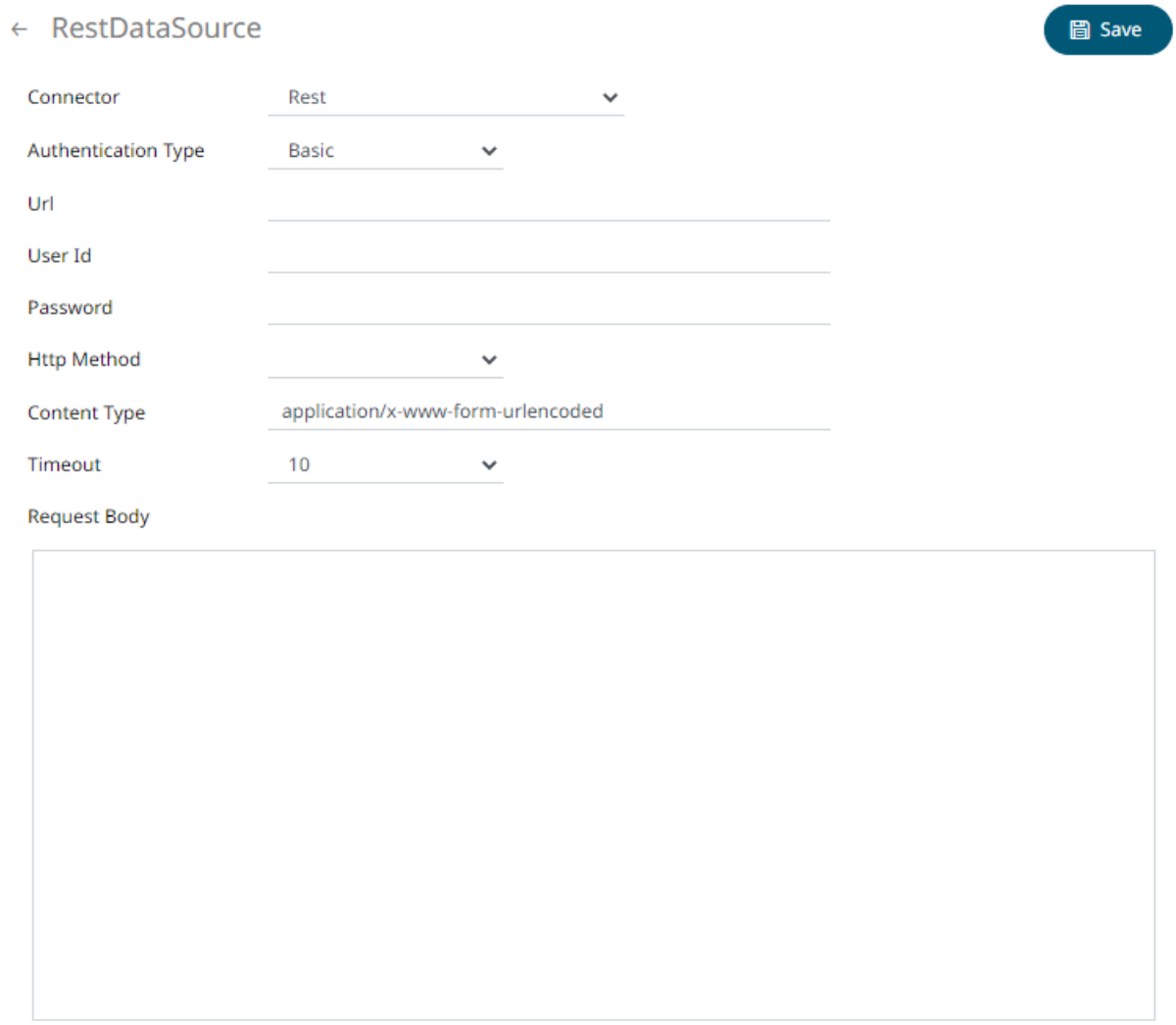
. The new data source is added in the *Data Sources* list.

Creating a REST Output Connector

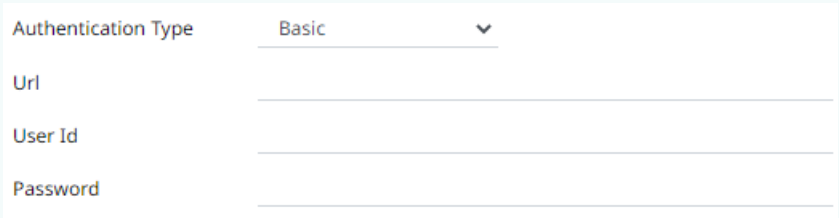
Outputs an event to a REST API. This output connector can also be used as an alerting system.

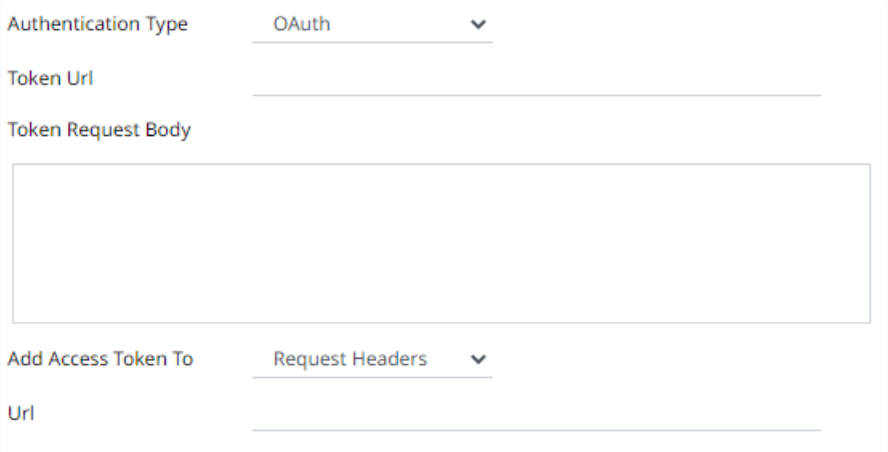
Steps:

1. On the **Data Source** tab, select **Output > Rest** in the *Connector* drop-down list.



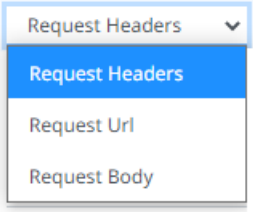
2. Define or select the following properties:

Property	Description
Authentication Type	• **Basic**  Enter the *URL* of the REST API. Then enter the *User Id* and the *Password* that will be used to the connect to the REST API. • **OAuth**



Then enter the following settings:

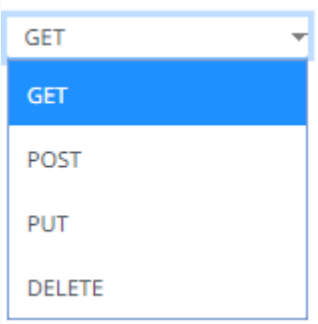
- **Token URL** – The URL to retrieve the access token from.
- **Token Request Body** – The request body used for access token requests.
- **Add Access Token To** - The Access token retrieved from the *Token URL* can be added to headers, URL or request body, depending on how the REST endpoint needs the token.



  - Request Header - A header is automatically added to the REST API request.
  - Request URL - The URL needs to be manually parameterised with a {access_token} parameter, before calling the REST API, the parameter is replaced with the actual token.
  - Request Body - The Request Body needs to be manually parameterised with a {access_token} parameter, before calling the REST API, the parameter is replaced with the actual token.
- URL – The URL of the REST API.

HTTP Method

Select the appropriate HTTP method for the request from the following options:



- GET – retrieve data
- POST – add new data

	• PUT – replace existing data • DELETE – remove existing data
Content Type	The required Content Type. Default is **application/x-www-form-urlencoded**
Timeout	The length of time to wait for the server response (10 to 300). Default is **10**.
Request Body	The Request Body for the HTTP POST.

3. Click

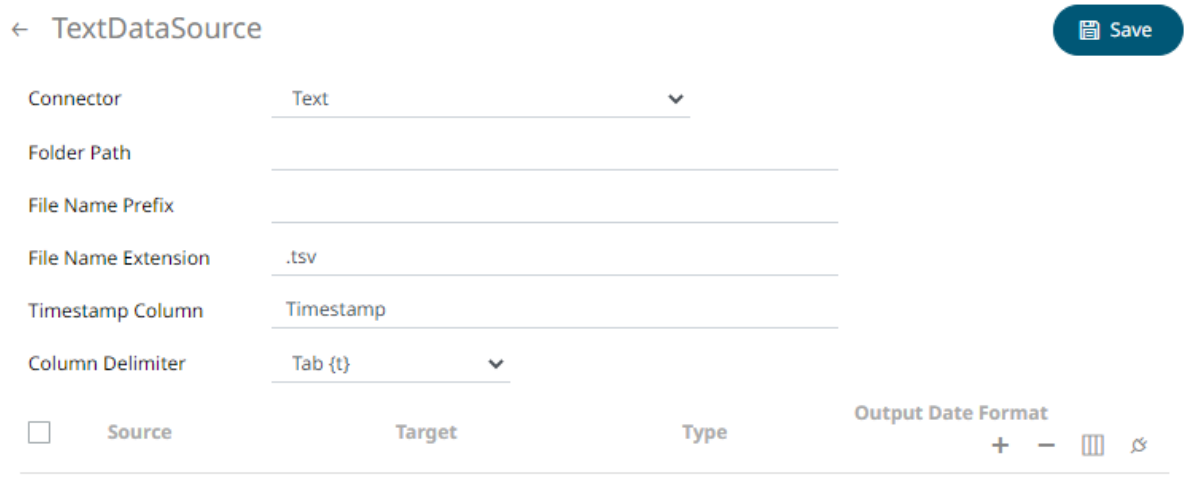
. The new data source is added in the *Data Sources* list.

Creating Text Output Connector

Allows retrieval and processing of delimited Text files (such as CSV, TSV, etc.). The files produced can be consumed by the Text connector.

Steps:

1 On the **Data Source** tab, select **Output > Text** in the *Connector* drop-down list.

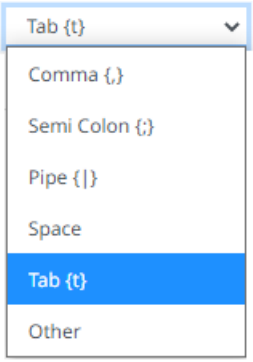


2 Define or select the following properties:

Property	Description
Folder Path	The path where the Text output will be placed.
File Name Prefix	The prefix for the file name. This can be parameterized with field names. Consequently, each event can generate a new file in the given folder. For example, if the Text output connector is attached as the consumer to StockStatic, you can use “{Region}” inside the *File Name Prefix*, causing it to create three files for Asia Pacific, Europe, and North America. Note that partitioning file names with current date in “yyyyMMdd” format is still done automatically and can’t be controlled, at the moment. For the StockStatic example, if it was executed today, it would have created three files like Asia Pacific_20181219.tsv.

File Name Extension	File name extension of the text output. Possible values are **.tsv** and **.csv**.
Timestamp Column	The name of the new column that will include the timestamp. Default is **Timestamp**.

3. Select the *Column Delimiter* from the drop-down list to be used when parsing the text file.



4. You may opt to use the toolbar to complete the data source definition.

5. Click

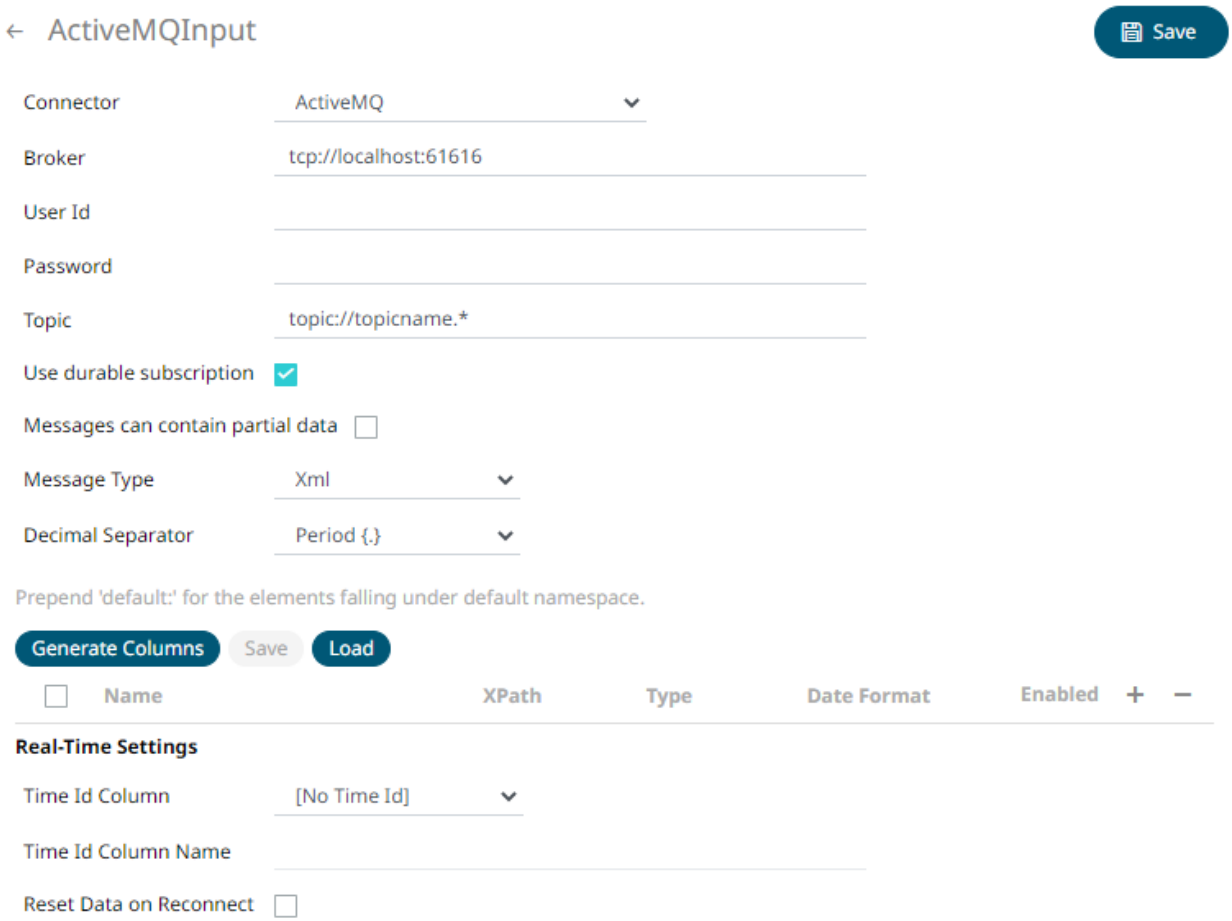
. The new data source is added in the *Data Sources* list.

Creating ActiveMQ Input Data Source

Allows connection to Apache's ActiveMQ message bus on a real-time streaming basis. Specifically, the connector allows Panopticon Streams to subscribe to XML, JSON or FIX based messages that are published on topics. The data format itself is arbitrary, and consequently, the connection includes the message definition.

Steps:

1. In the *New Data Source* page, select **Input > ActiveMQ** in the *Connector* drop-down list.



2. Enter the following information:

Property	Description
Broker	The location of the message broker. Default is **tcp://localhost:61616**.
User Id	The user Id that will be used to connect to the ActiveMQ service.
Password	The password to connect to the ActiveMQ service.
Topic	Accepts topic in `topic://topicname.*` format and also `topicname.*`. Therefore, `topic://pano.>` and `pano.>` both will work as topic value. Default is **topic://topicname.***

3. Check/uncheck the **Use durable subscription** box.

> **NOTE** **When connecting to a message bus, it is recommended to** disable **durable messaging. When it is enabled, this puts a heavier load to the server, and slows down the start and stop of subscriptions.**

4. Check/uncheck **Messages can contain partial data** box.
5. Select the *Message Type*.

6. Select either the period (.) or comma (,) as the *Decimal Separator*.

> **NOTE** **Prepend 'default:' for the elements falling under default namespace.**

7. Click Generate Columns to the fetch the schema based on the connection details. Consequently, the list of columns with the data type found from inspecting the first 'n' rows of the input data source is populated and the **Save** button is enabled.

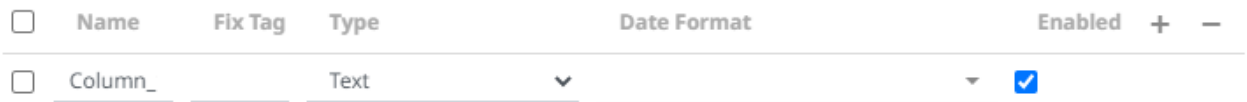
8. You can also opt to load or save a copy of the column definition.

9. You can also opt to click + to add columns to the MQ connection that represent sections of the message. Then enter or select:

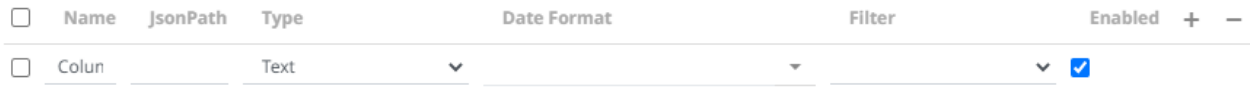
Property	Description
Name	The column name of the source schema.
Fix Tag/JsonPath/Text Column Index/XPath	The Fix Tag/JsonPath/Text Column Index/XPath of the source schema.
Type	The data type of the column. Can be a **Text**, **Numeric**, or **Time**
Date Format	The format when the data type is **Time**.
Filter	Defined parameters that can be used as filter.
Enabled	Determines whether the message field should be processed.

> **NOTE** **To parse and format times with higher than millisecond precision, the format string needs to end with a period followed by sequence of upper case S. There can be no additional characters following them.**
>
> **For example: yyyy-MM-dd HH:mm:ss.SSSSSS**

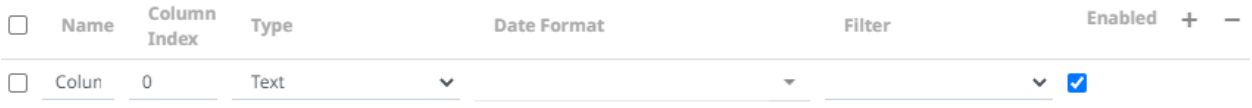
If *Message Type* is set to **Fix**, the *Add Column* will display as:



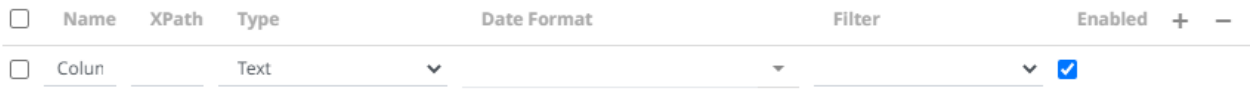
If *Message Type* is set to **JSON**, the *Add Column* will display as:



If *Message Type* is set to **Text**, the *Add column* will display as:



If *Message Type* is set to **XML**, the *Add column* will display as:



To delete a column, check its ☐ or all the column entries, check the topmost ☐, then click −.

10. Define the Real-time Settings.

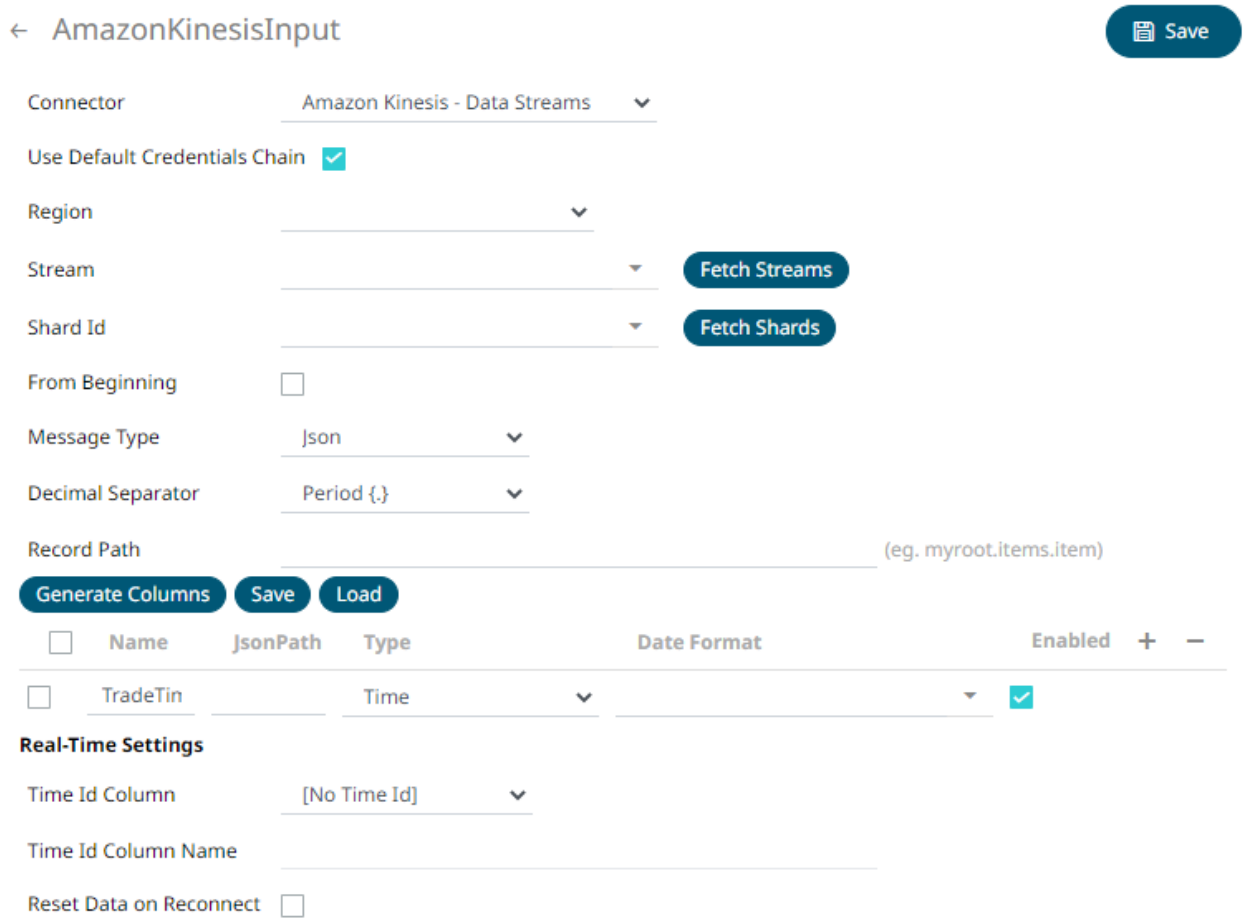
11. Click Save. The new data source is added in the *Data Sources* list.

Creating Amazon Kinesis – Data Streams Data Source

The Amazon Kinesis – Data Streams connector reads records from the given data stream and Shard ID.

Steps:

1. In the *New Data Source* page, select **Input > Amazon Kinesis – Data Streams** in the *Connector* drop-down list.

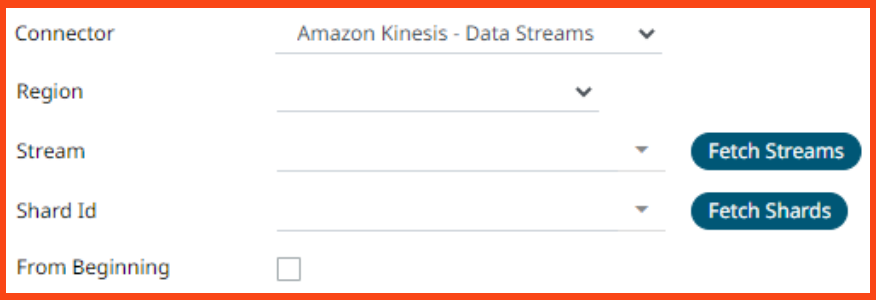


2. You can either:
   - check the **Use Default Credentials Chain** box to use the default *Access Key ID* and *Secret Key Access*, or
   - uncheck the **Use Default Credentials Chain** box and enter the Access Key ID and Secret Key Access

NOTE **The *Access Key ID* and *Secret Key Access* from the AWS account can be configured in three places:**

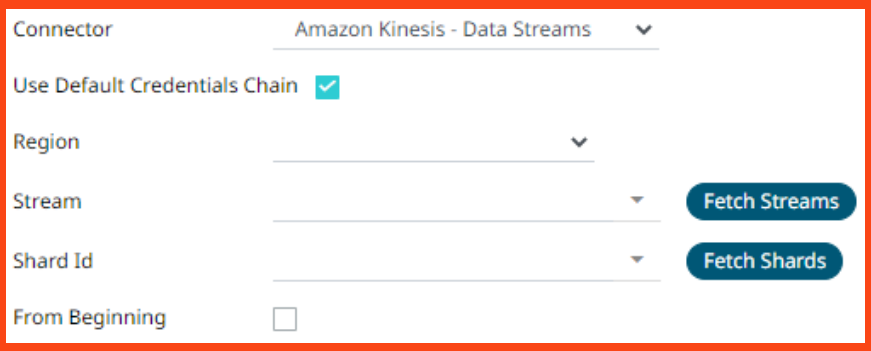
- **Two properties at the `Streams.properties` file which is available in the `AppData` folder of Panopticon Streams Server**
  - **`connector.kinesis.datastreams.accesskeyid`**
  - **`connector.kinesis.datastreams.secretaccesskey`**

  If this is the used configuration, the Use Default Credentials Chain box is not displayed in the connector UI.



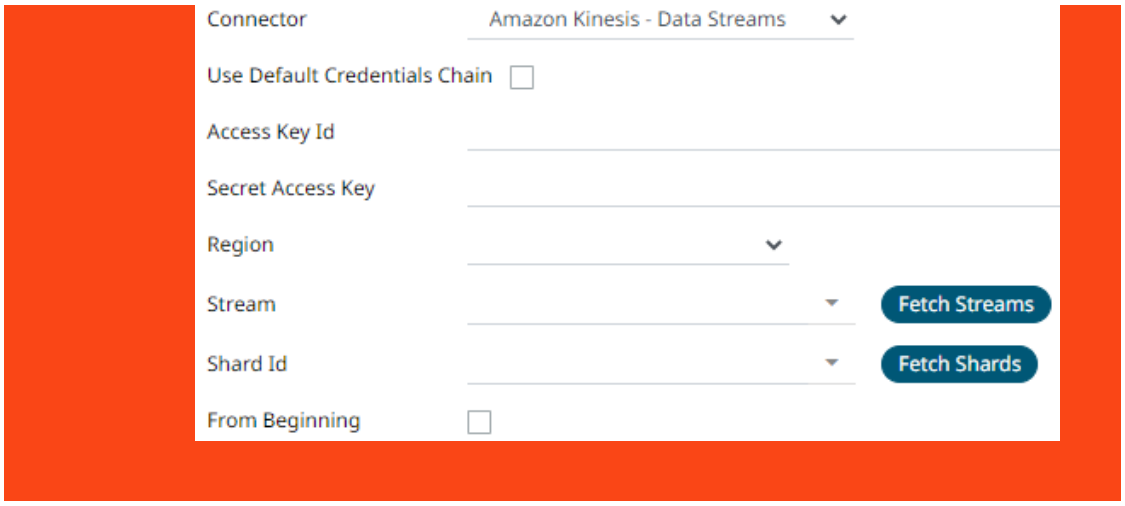
  This is the recommended way to provide the credentials.

- **AWS credentials provider chain**
  - **Environment Variables - `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`**
  - **Credential profiles file at the default location - `~/.aws/credentials` on Linux, macOS, or Unix, and `C:\Users\USERNAME\.aws\credentials` on Windows.**



- **Dedicated fields in the connector**

  Not the recommended configuration.



3. Select or define the following properties:

Property	Description
Region	Physical location of the data center. The list is picked up from the Amazon Kinesis Data Streams Endpoints and Quotas page.
Stream	Name of the stream from where you want to pull the data. Click **Fetch Streams** Fetch Streams to load all of the available streams from the AWS account.
Shard Id	Each connector instance or data source is connected to only one shard. Click **Fetch Shards** Fetch Shards to pull all of the shards from the selected stream.
From Beginning	The starting position in the data stream from which to start streaming. Default value is unchecked, which means **LATEST.** When checked, the starting position is set to **TRIM_HORIZON**.

NOTE **All of the connection settings can be parameterized.**

4. Select the *Message Type*.
5. Select either the dot (.) or comma (,) as the *Decimal Separator*.

NOTE **Prepend 'default:' for the elements falling under default namespace.**

6. Click Generate Columns to the fetch the schema based on the connection details. Consequently, the list of columns with the data type found from inspecting the first ‘n’ rows of the input data source is populated and the **Save** button is enabled.

   This also populates the *Id column* with the set of columns, of arbitrary type, that can be concatenated to form a unique row identifier.

7. You can also opt to load or save a copy of the column definition.

10. Click + to add columns to the Amazon Kinesis – Data Streams connection that represent sections of the message. Then enter or select:

Property	Description
Name	The column name of the source schema.
Fix Tag/JsonPath/Text Column Index/XPath	The Fix Tag/JsonPath/Text Column Index/XPath of the source schema.
Type	The data type of the column. Can be a **Text, Numeric,** or **Time**
Date Format	The format when the data type is **Time**.
Enabled	Determines whether the message field should be processed.

NOTE

To parse and format times with higher than millisecond precision, the format string needs to end with a period followed by sequence of upper case S. There can be no additional characters following them.

For example: yyyy-MM-dd HH:mm:ss.SSSSSS

To delete a column, check its ☐ or all the column entries, check the topmost ☐, then click −.

11. Define the Real-time Settings.

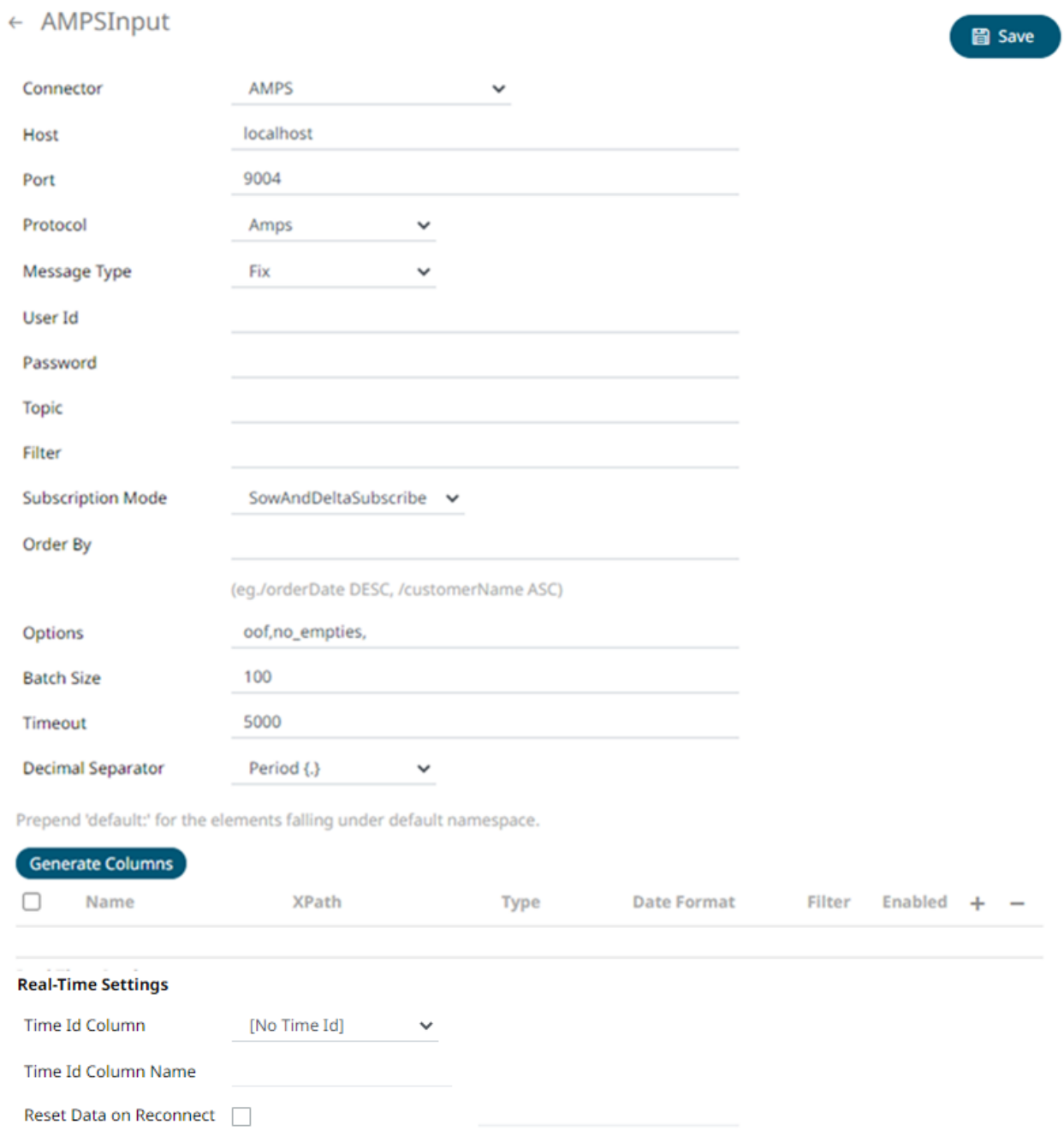
12. Click Save. The new data source is added in the *Data Sources* list.

Creating AMPS Input Data Source

The AMPS connector allows connection to AMPS message bus on a real-time streaming basis. The connector allows Panopticon Streams to subscribe to the Native FIX and XML message support. The data format itself is arbitrary, and in turn the connection includes the message definition.

Steps:

1. In the *New Data Source* page, select **Input > AMPS** in the *Connector* drop-down list.



2. Enter the following information:

Property	Description
Host	AMPS host address.
Port	AMPS host port. Default is **9004**.
User Id	The user Id that will be used to connect to the AMPS service.
Password	The password to connect to the AMPS service.
Topic	The topic or queue physical name.

Filter	The filter expression.

3. Select the *Protocol*. This will specify the format of the headers:
   - Amps (default)
   - Fix
   - NvFix
   - XML
4. Select the *Message Type*. This will specify the format of the data within the message:
5. Select from any of the following *Subscription Modes*:
   - Sow
   - SowAndSubscribe
   - SowAndDeltaSubscribe (default)
   - Subscribe
   - DeltaSubcribe
6. Enter the *Order By Statement* in order to limit the returned data. For example:

   /orderDate DESC

   /custumerName ASC
7. Enter any of the following *Option/s* for the selected *Subscription Mode*:
   - cancel
   - live
   - no_empties
   - null
   - no_sowkey
   - oof
   - pause
   - replace
   - resume
   - send_keys
   - timestamp

   NOTE **Leave the *Options* box blank if you selected the** Subscribe **subscription mode.**

8. Enter the *Batch Size*. This is the number of messages that will be sent at a time as results are returned. Default is **100**.
9. Enter the *Timeout* for the length of time to wait for the Server response. Default is **5000**.

10. Select either the period (.) or comma (,) as the *Decimal Separator*.

NOTE **Prepend 'default:' for the elements falling under default namespace.**

11. Click Generate Columns to the fetch the schema based on the connection details. This populates the list of columns with the data type found from inspecting the first 'n' rows of the input data source.

12. You can also opt to click + . This adds columns to the AMPS connection that will represent sections of the message.

13. Provide the following information:

Property	Description
Name	The column name of the source schema.
Fix Tag/XPath/Json Path	The Fix Tag/XPath/Json Path of the source schema.
Type	The data type of the column. Can be a **Text, Numeric,** or **Time**
Date Format	The format when the data type is **Time.**
Filter	Defined parameters that can be used as filter. Only available for Fix, JSON, and XML message types.
Enabled	Determines whether the message field should be processed.

To delete a column, check its ☐ or all the column entries, check the topmost ☐ , then click − .

14. Define the Real-time Settings.

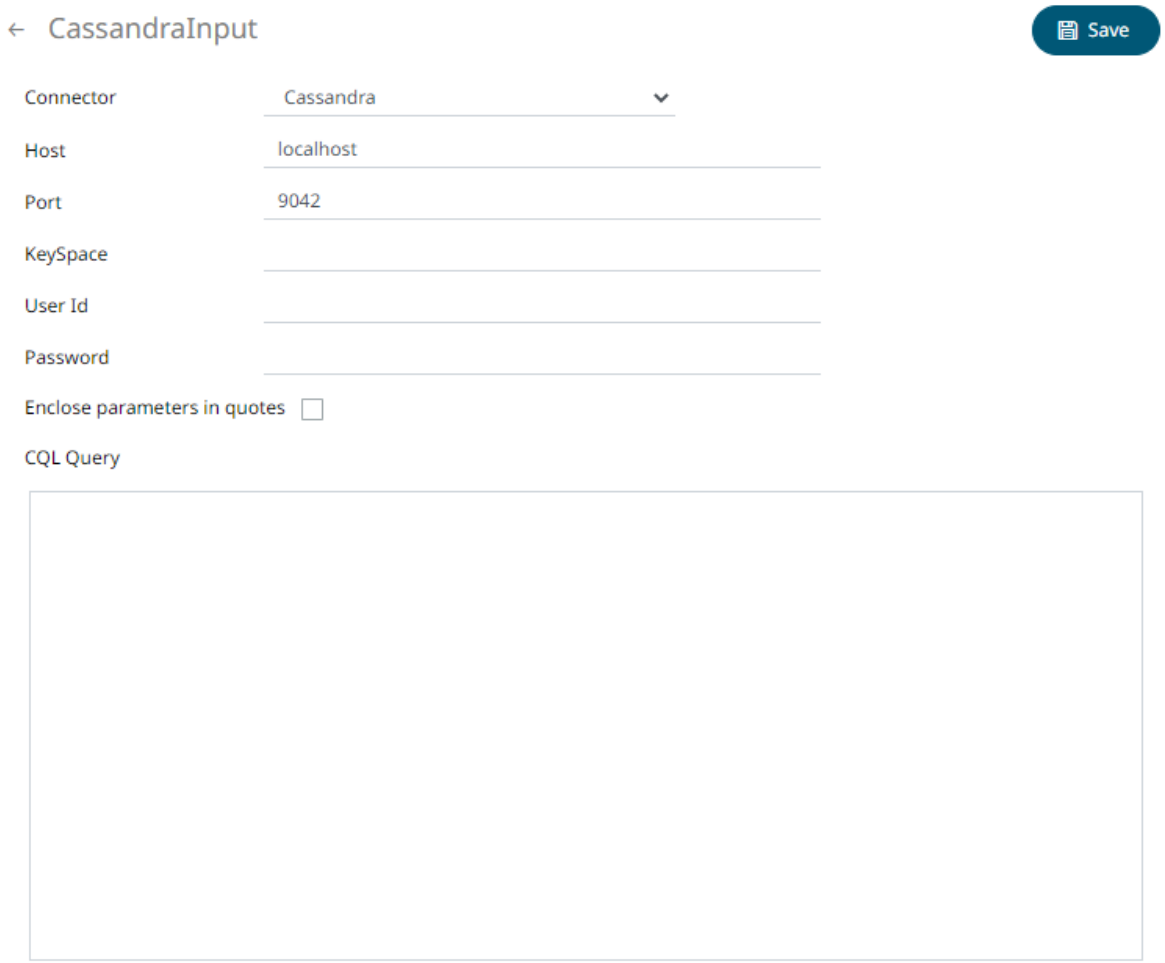
15. Click Save **.** The new data source is added in the *Data Sources* list.

Creating Cassandra Input Data Source

The Apache Cassandra connector allows connection to Apache and Datastax Cassandra instances, by executing a pre-defined CQL query, and retrieving the resulting data.

Steps:

1. On the *New Data Extract* page, select **Cassandra** in the *Connector* drop-down list.



2. Enter the following information:

Property	Description
Host	Apache Cassandra host address.
Port	Apache Cassandra host port. Default is **9042**.
KeySpace	Namespace that defines data replication in nodes.
User Id	The username used to connect to the Apache Cassandra service.
Password	The password used to connect to the Apache Cassandra service.

3. Select whether the parameters should be automatically enclosed in quotes, by checking the **Enclose parameters in quotes** box.
4. Enter the *CQL Query*, which can contain parameters in a similar manner to the database connector.
5. The time zone of input parameters and output data is by default, unchanged. Changing the time zone is supported by using the *Show in Timezone* drop-down list box based on the assumption that data are stored in UTC time and outputs are presented in the selected time zone.
6. Click

. The new data source is added in the *Data Sources* panel.

Creating Elasticsearch 6.x Input Data Source

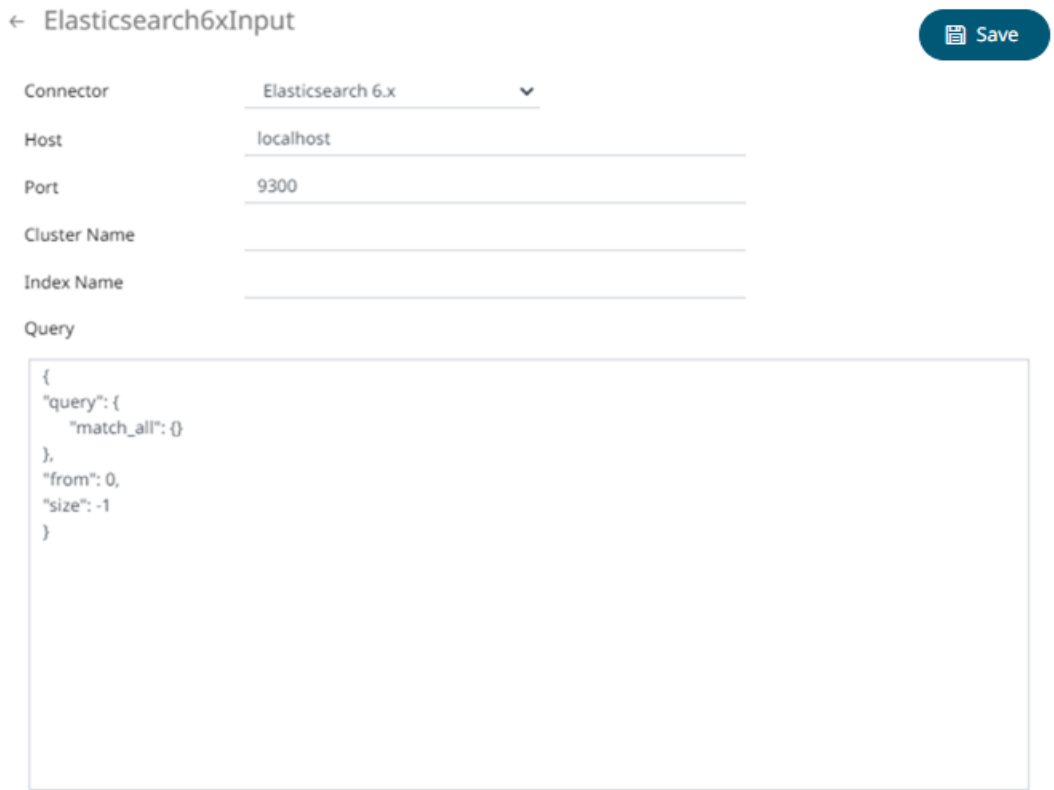
The Elasticsearch 6.x connector allows you to connect and access data from an Elasticsearch cluster using Transport Client.

NOTE

- **To enable the Elasticsearch 6.x connector, refer to Elasticsearch Connectors Dependency Installation for more information on how to copy the provided dependency files to the `Lib` folder.**
- **The Elasticsearch 6.x connector supports Elasticsearch 6.x versions.**
- **Elasticsearch 6.x and Elasticsearch 7.x connectors will not work in a single Panopticon Streams Server instance due to conflicting Elasticsearch API dependencies.**

Steps:

1. In the *New Data Source* page, select **Input > Elasticsearch 6.x** in the *Connector* drop-down list.



2. Enter the following information:

Property	Description
Host	The hostname of any node in your Elasticsearch cluster, or localhost for a node on your local machine.

Port	The port running the Elasticsearch HTTP service (default is **9300**). If the port you wish to use is different from the default port, change the value to the correct one.
Cluster Name	The cluster name that can be used to discover and auto-join nodes.
Index Name	The Index name in Elasticsearch. This is some type of data organization mechanism that allows partition of data in a certain way.

3. Enter an optional JSON-encoded request body in the *Query* box.
4. Click

. The new data source is added in the *Data Sources* panel.

Creating Elasticsearch 7.x Input Data Source

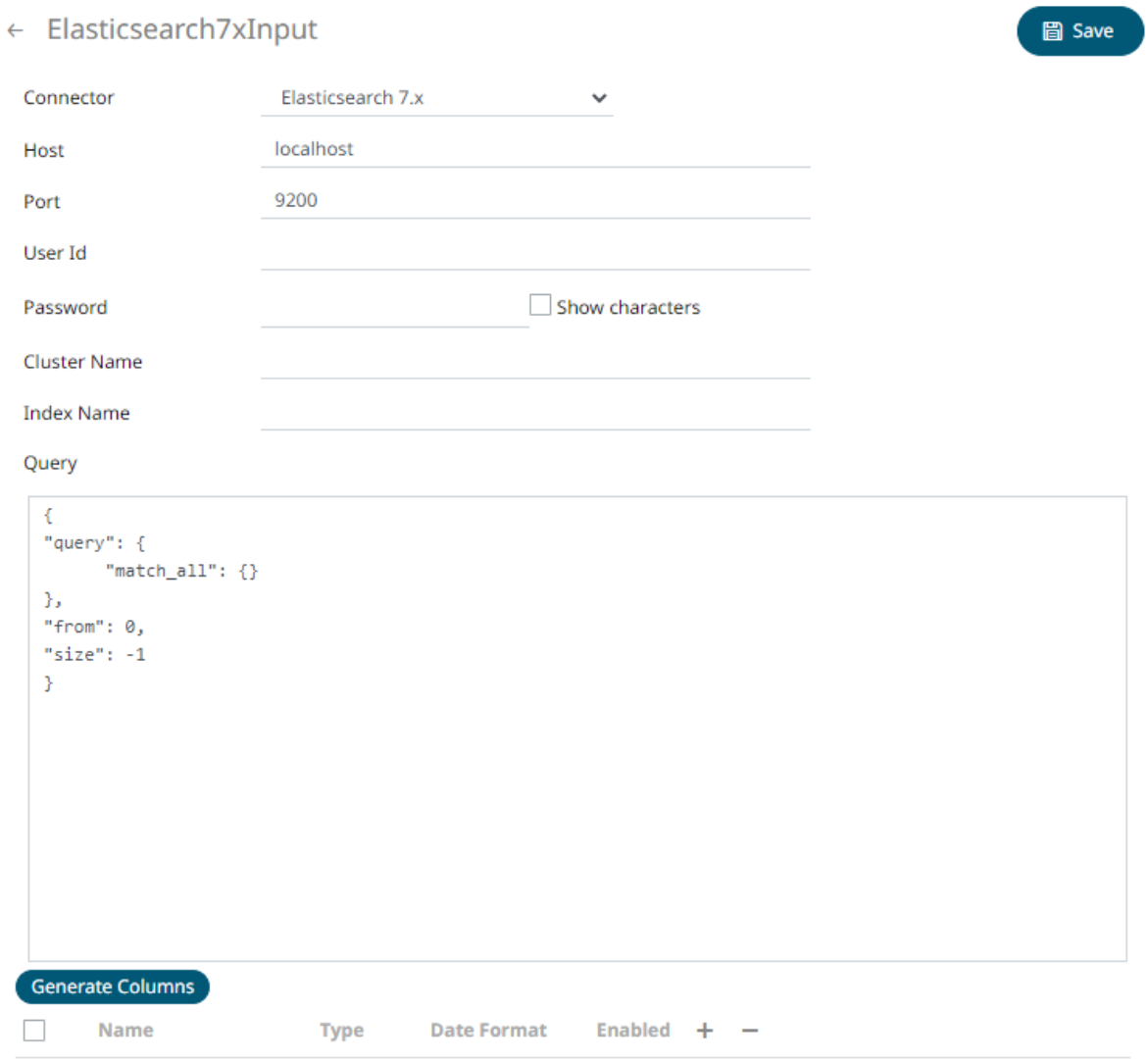
The Elasticsearch 7.x connector allows you to connect and access data from an Elasticsearch cluster using Java High Level REST Client.

NOTE

- **Similar to Elasticsearch 6.x connector but uses Java High Level REST Client.**
- **To enable the Elasticsearch 7.x connector, refer to Elasticsearch Connectors Dependency Installation for more information on how to copy the provided dependency files to the Lib folder.**
- **The Elasticsearch 7.x connector supports Elasticsearch 7.x versions.**
- **Elasticsearch 6.x and Elasticsearch 7.x connectors will not work in a single Panopticon Streams Server instance due to conflicting Elasticsearch API dependencies.**

Steps:

1. In the *New Data Source* page, select **Input > Elasticsearch 7.x** in the *Connector* drop-down list.



2. Enter the following information:

Property	Description
Host	The hostname of any node in your Elasticsearch cluster, or localhost for a node on your local machine.
Port	The port running the Elasticsearch HTTP service (default is **9300**). If the port you wish to use is different from the default port, change the value to the correct one.
Cluster Name	The cluster name that can be used to discover and auto-join nodes.
Index Name	The Index name in Elasticsearch. This is some type of data organization mechanism that allows partition of data in a certain way.

3. Enter an optional JSON-encoded request body in the *Query* box.

4. Click Generate Columns. The columns populate the *Output Column* section.

5. Click + to add columns and specify their properties:

Property	Description
Name	The column name of the source schema.
Type	The data type of the column. Can be a **Text, Numeric**, or **Time**
Date Format	The format when the data type is **Time**.
Enabled	Determines whether the message field should be processed.

To delete a column, check its ☐ or all the column entries, check the topmost ☐, then click −.

6. Click Save. The new data source is added in the *Data Sources* panel.

Elasticsearch Connectors Dependency Installation

Dependencies for each supported Elasticsearch version are included in the Panopticon Streams Server zip as individual zip archive files:

- Elastic_6X_Dependencies.zip
- Elastic_7X_Dependencies.zip.

Steps:

1. Select the target Elasticsearch version and unzip the contents of the appropriate dependency zip into the `tomcat/webapps/streams/WEB-INF/lib` folder to enable connectivity for a specific server instance.
2. Restart Tomcat.

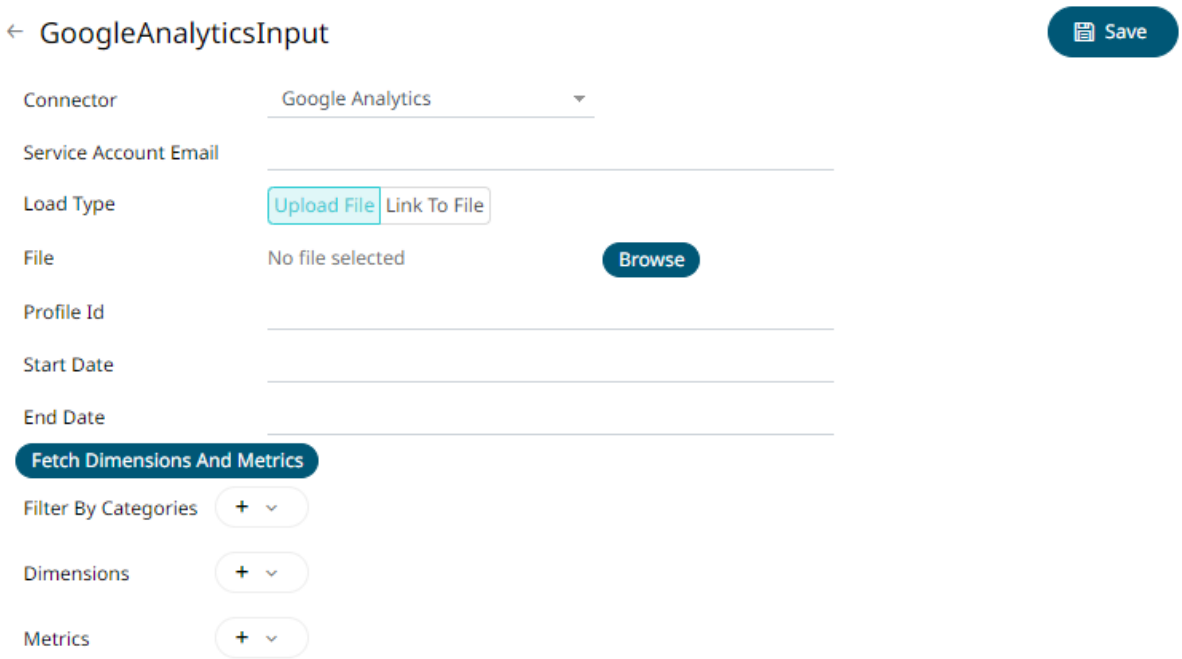
Creating Google Analytics Input Data Source

The Google Analytics connector allows you to track and report website traffic using the Google Analytics service. You will need the following to fetch Google Analytics data:

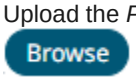
Field	Description
Service Account E-mail	The Service Account ID that is generated while creating credentials for the service account authentication.
Key File	The Key File (**.p12**) furnished by Google Analytics when you created the Service Account.
Profile ID	The Profile ID of the page you want to access in Google Analytics.

Steps:

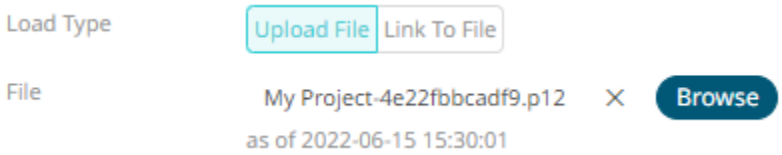
1. In the *New Data Source* page, select **Input > Google Analytics** in the *Connector* drop-down list.



2. Enter the *Service Account Email* that was generated while creating credentials to the service account authentication.
3. Set the *Key File* that will be used to connect to Google Analytics in Panopticon. Do one of the following:

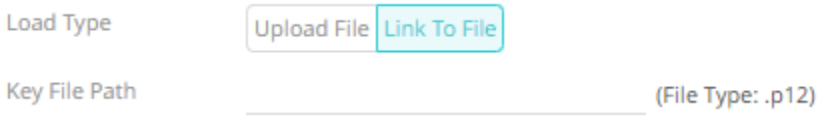
   - Upload the *Personal Information Exchange* file by clicking **Upload File** then **Browse**  to browse to the file.

     After selecting the file, it is displayed with the timestamp.

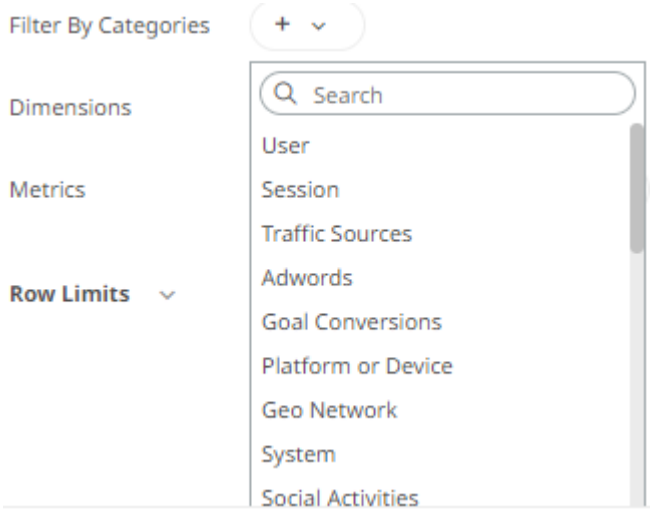
     To change the key file, click ✕ then **Browse** to browse to a new version of the file.

   - Link to a *Personal Information Exchange* file by clicking **Link to File** and entering a *File Path*.

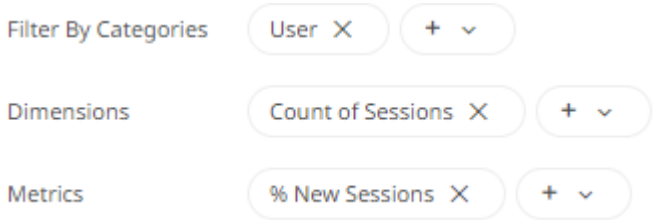
     

4. Enter the *Profile ID* of the page you want to access in Google Analytics.
5. Enter the *Start Date* and *End Date*, if needed.
6. Click . This populates the *Filter By Categories*, *Dimensions*, and *Metrics* list boxes.

7. Click then select any field from these list boxes.



For example:



Click to remove a field.

8. Click **.** The new data source is added in the *Data Sources* list.

Creating Google Cloud Pub/Sub Input Data Source

The Google Cloud Pub/Sub connector allows connection to Google Cloud Pub/Sub's message bus on a real-time streaming basis. Specifically, the connector allows Panopticon Streams to subscribe to XML, JSON, TEXT or FIX based messages that are published on particular topics. The data format itself is arbitrary, and consequently, the connection includes the message definition.

Steps:

1. In the *New Data Source* page, select **Input > Google Cloud PubSub** in the *Connector* drop-down list.

← GoogleCloudPubSubInput

Save

Connector: Google Cloud PubSub

Service Account Credential Json Text

Topic: Fetch

Subscription Name: Fetch

Message Type: Json

Decimal Separator: Period {.}

Record Path: (eg. myroot.items.item)

Generate Columns | Save | Load

Name	JsonPath	Type	Date Format	Enabled
TradeTi		Time		✓

Attribute Columns

Name	Attribute Name	Enabled

Real-Time Settings

Time Id Column: [No Time Id]

Time Id Column Name

Reset Data on Reconnect

2. Enter the *Service Account Credential JSON Text* with the generated JSON key (contains the private key) in the following format:

```
{
  "type": "service_account",
  "project_id": "project-id",
  "private_key_id": "some_number",
  "private_key": "-----BEGIN PRIVATE KEY-----\n....
  =\n-----END PRIVATE KEY-----\n",
  "client_email": "<api-name>api@project-id.iam.gserviceaccount.com",
  "client_id": "...",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://accounts.google.com/o/oauth2/token",
  "auth_provider_x509_cert_url":
"https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/...<api-
name>api%40project-id.iam.gserviceaccount.com"
}
```

NOTE **Ensure that when parameterizing the values in the Credential JSON Text, there is no white space as a single line content.**

3. Click Fetch to populate the *Topic* drop-down list. Initially, the first topic in the list is displayed in the *Topic* drop-down box.

   Select a topic.

4. Click Fetch to populate the *Subscription Name* drop-down list and select a subscription name.

   You can also opt to create a subscription by manually entering the value into the *Subscription Name* list box.

NOTE

- **A subscription name will be automatically generated when it is not entered or selected in the drop-down list.**

  This subscription will be created for connection and will be deleted as soon as its work is done. For example, when starting a presentation mode, a subscription will be created. Upon quitting the presentation mode, the subscription will then be deleted.

- **Pub/Sub can automatically delete inactive subscriptions. This can be done by configuring the minimum required time of inactivity to schedule a subscription for deletion. This time must be longer than the message retention duration.**

5. Select the *Message Type*.
6. elect either the period (.) or comma (,) as the *Decimal Separator*.

NOTE **Prepend 'default:' for the elements falling under default namespace.**

7. Click Generate Columns to the fetch the schema based on the connection details. Consequently, the list of columns with the data type found from inspecting the first 'n' rows of the input data source is populated and the Save button is enabled.

8. You can also opt to load or save a copy of the column definition.

9. Click +. This adds columns to the Google Cloud Pub/Sub connection that will represent sections of the message.

10. Provide the following information:

Property	Description
Name	The column name of the source schema.
Fix Tag/XPath/Json Path	The Fix Tag/XPath/Json Path of the source schema.
Type	The data type of the column. Can be a **Text, Numeric,** or **Time**
Date Format	The format when the data type is **Time**.
Filter	Defined parameters that can be used as filter. Only available for JSON, Text, and XML message types.
Enabled	Determines whether the message field should be processed.

To delete a column, check its ☐ or all the column entries, check the topmost ☐, then click −.

11. Google Cloud Pub/Sub messages can have additional metadata as custom attributes.

Panopticon Google Cloud Pub/Sub connector supports reading these attributes as column values. The generate column logic automatically checks and generates attribute columns if messages received contain attributes.

Additionally, like columns from message data, you can manually add them by clicking +. A new entry displays.

Name can be any unique column name within the data source. The attribute name must match to an attribute name in message otherwise it will be treated as null value. Currently all attribute columns are treated as Text columns, we can't change column type.

Check the *Enabled* box to enable an attribute column.

To delete an attribute column, check its ☐ or all the column entries, check the topmost ☐, then click −.

12. Define the Real-time Settings.

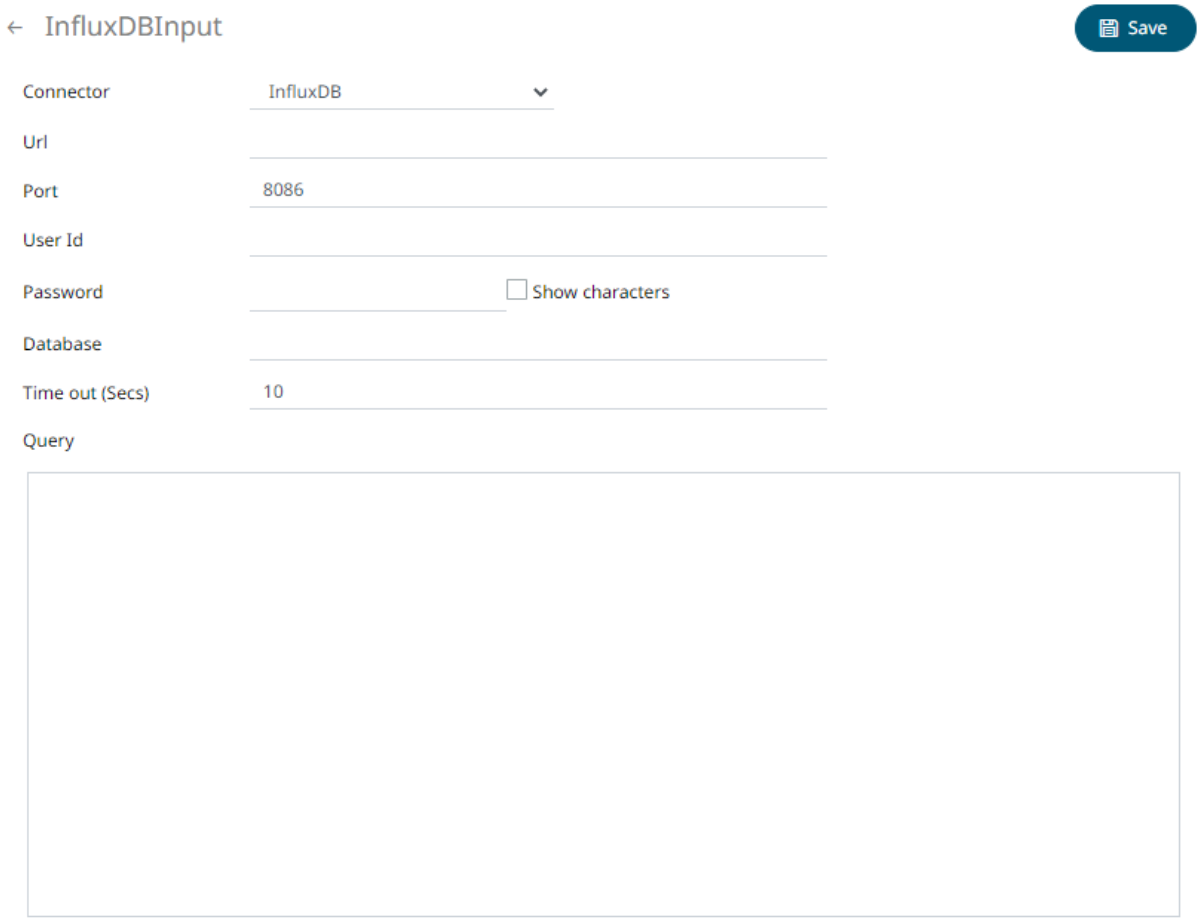
13. Click Save. The new data source is added in the *Data Sources* list.

Creating an InfluxDB Input Data Source

The InfluxDB connector allows for the retrieval of a JSON data set from the InfluxDB. The database communicates over HTTP(S) where you can define a query in the URL to return the desired data.

Steps:

1. In the New Data Source page, select **Input > InfluxDB** in the *Connector* drop-down list.



2. Enter the following information:

Property	Description
URL	InfluxDB host address.
Port	InfluxDB host port. Default is **8086**.
User Id	The user Id that will be used to connect to the InfluxDB service.
Password	The password to connect to the InfluxDB service. Check the **Show Characters** box to display the entered characters.
Database	The name of the database that will communicate over the HTTP(S).
Time out (Secs)	The time out period applied to both the TCP socket and for individual read IO operations. Default is **10**.

3. Enter an SQL-like query language into the *Query* box.

4. Click

. The new data source is added in the *Data Sources* list.

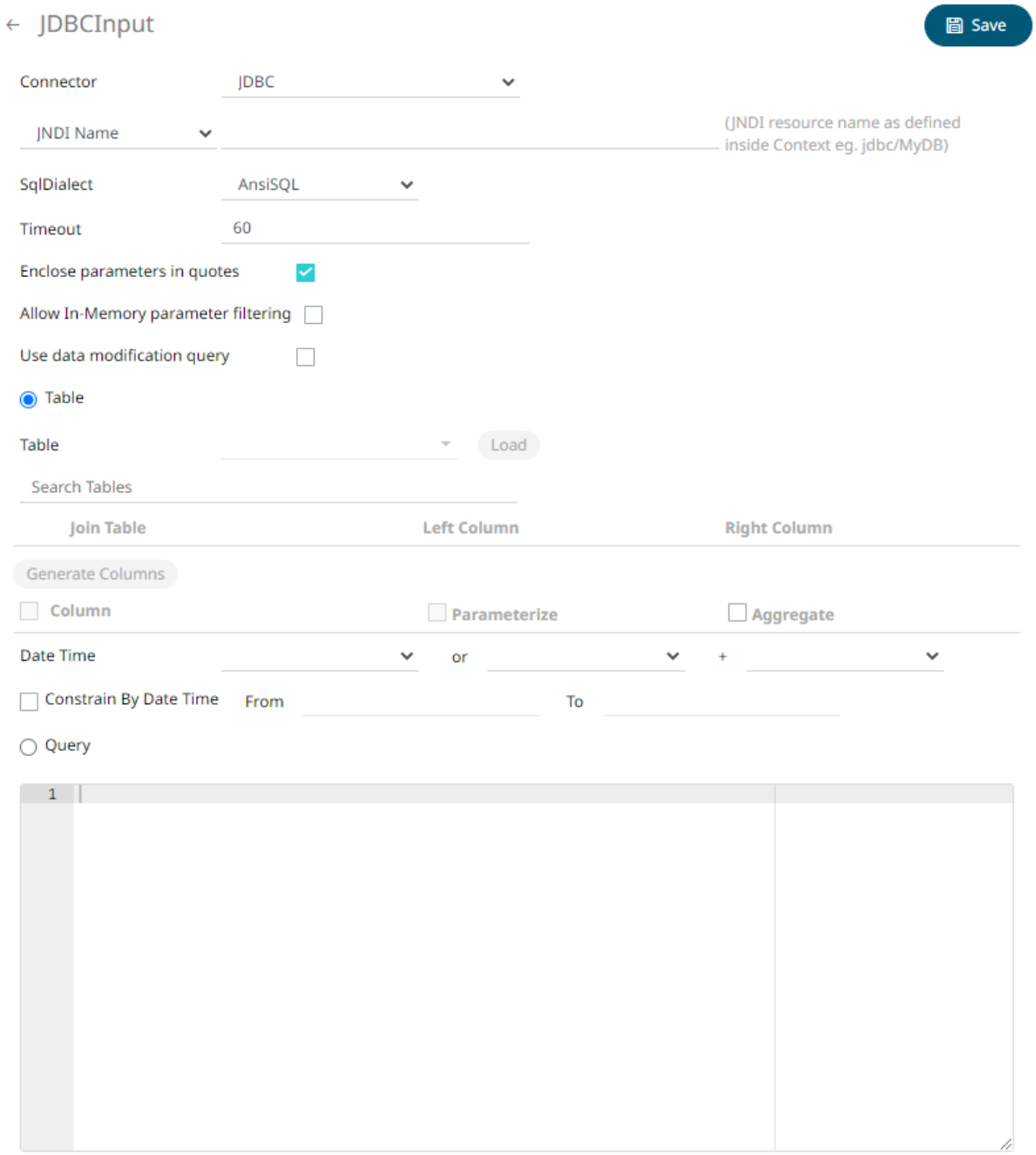
Creating JDBC Database Input Data Source

The JDBC connector allows the retrieval and processing of data from virtually any database, that has a JDBC driver.

IMPORTANT For DolphinDB, the query builder is not supported, only the query mode.

Steps:

1. In the *New Data Source* page, select **Input > JDBC** in the *Connector* drop-down list.



2. You can either select:

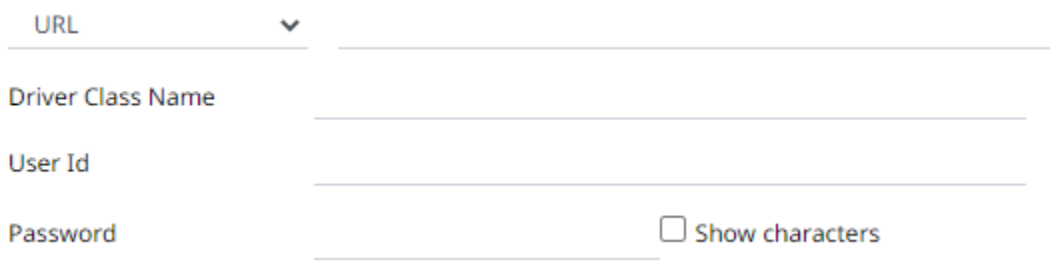
- JNDI Name

Enter the *JNDI resource name* to be used.

NOTE **The JNDI resource name needs to be on the form:**

`jdbc/[resourcename]`

- URL



  Enter the *URL* specific to the database's JDBC driver, the *Driver Class Name* specific to the driver, and the *User Id* and *Password*.

  Check the **Show Characters** box to display the entered characters.

3. Select the appropriate *SQL Dialect* in the drop-down list to be able to generate the correct SQL for the required data repository.

   You can select any of the following *SQL dialects*: AnsiSQL, Access/Excel, MySQL, Oracle, SQL Server, Sybase IQ/ASA, Sybase ASE, Netezza, Vertica, SQLite, HadoopHive, KxQ, DB2, PostgreSQL, Impala, Redshift, Informix, Teradata, dBase, SparkSQL.

   Default is **AnsiSQL.**

4. Enter the *Timeout*. This is the length of time to wait for the server response. Default is **60**.

5. Check any of the following options when building the query:

   - Enclose parameters in quotes

     By default, this option is checked, as the common use case for parameters is a filter `WHERE` clause.

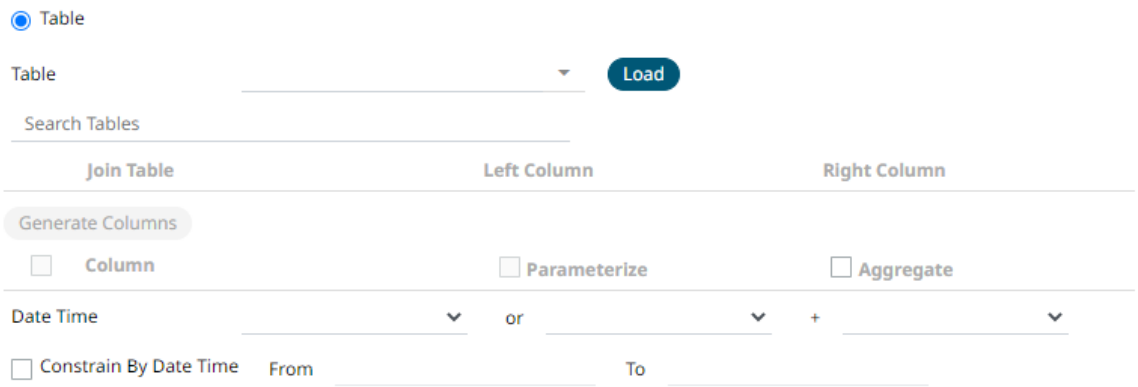
   - Allow in-memory parameter filtering

     Allows the whole dataset to be returned, and then filtered in memory. This process is much less efficient than adding the parameter as a `WHERE` clause of the SQL query; however, it may be efficient in cases where small sets of records are returned on a very frequent basis.

   - Use data modification query

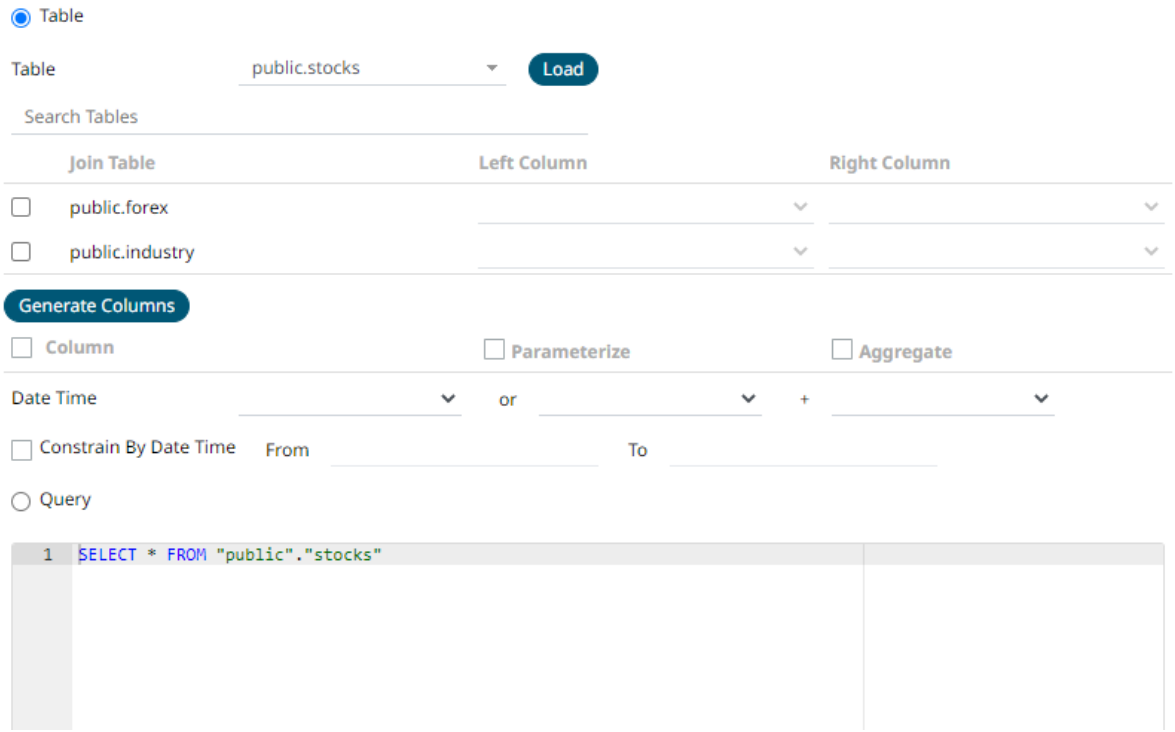
     Signals that the table is created for writing data. This property is also used for filtering out target data tables for further data update action configuration

6. When **Table** is selected, the section below is enabled:



7. On the *Table* field, click to populate the drop-down list with tables. Select a table.

   The list of tables that you can join is displayed. Also, the SQL query is generated and displayed in the *Query* text box.



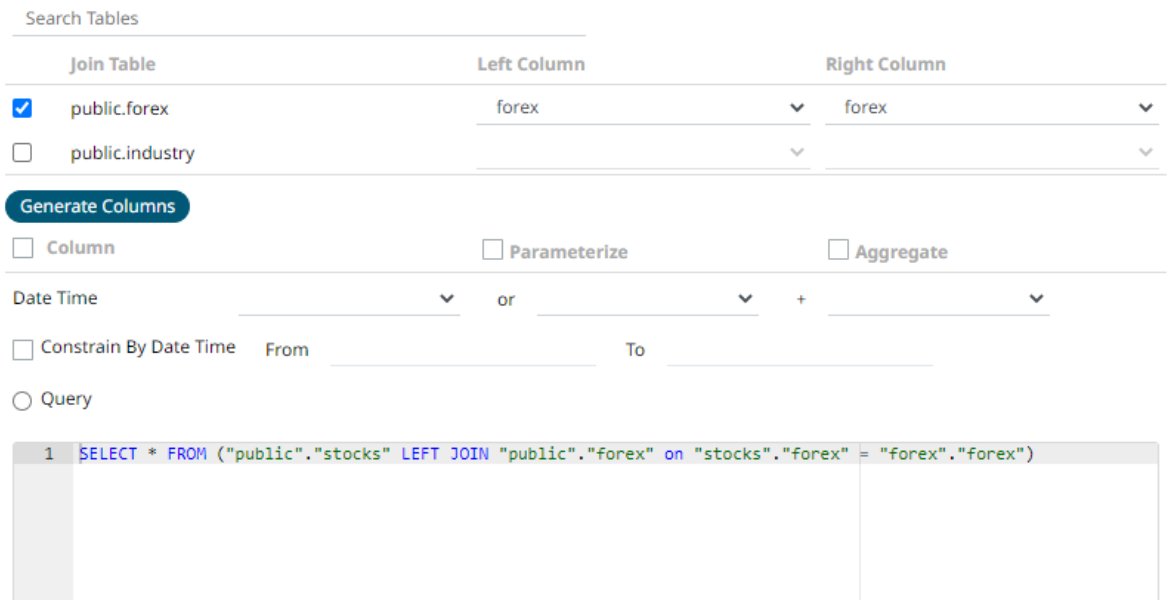
   Use *Search Tables* to filter the list.

8. Perform a join by checking one or more tables in the list.

   The *Left Column* and *Right Column* fields are automatically filled out with the common fields.

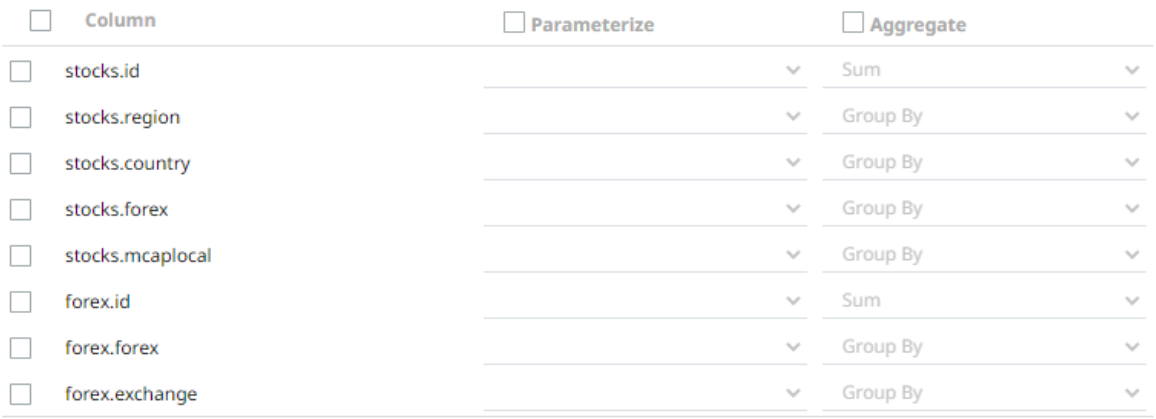
You can also opt to select other common fields.

The SQL query is generated and displayed in the *Query* text box.



9. Click

. The columns populate the *Output Column* section.



10. Individual columns can be added by checking the corresponding *Column* box in the *Output Column* listing. To select all of the columns, check the topmost box.

    The SQL query is generated and displayed in the *Query* text box.

11. If the data returned is to be aggregated, then the **Aggregate** box should be checked. For each selected column, the possible aggregation methods are listed including:
    - Text Columns: Last, First, Count, Group By
    - Date Columns: Count, Min, Max, Group By
    - Numeric Columns: Last, First, Sum, Count, Min, Max, Mean, Group By

    The SQL query is generated and displayed on the *Query* text box.
12. Check the **Parameterize** box and match the parameter to the appropriate column. By default, they will be matched by name.

    The appropriate SQL Query is updated in the *Query* text box. This shows the default parameter value for the preview, and at run time the SQL will be updated to whatever the parameter value is.
13. If the data is to be filtered or aggregated on Date/Times, then a valid *Date Time* field needs to be selected from either a single Date/Time field, or a compound column created from a selected *Date* and a selected *Time* column.

14. Check the **Constrain by Date Time** box and enter *From* and *To* Date/Time constraints.
15. Click the **Query** radio button to enable the text box and modify the SQL-like query language.
16. Click

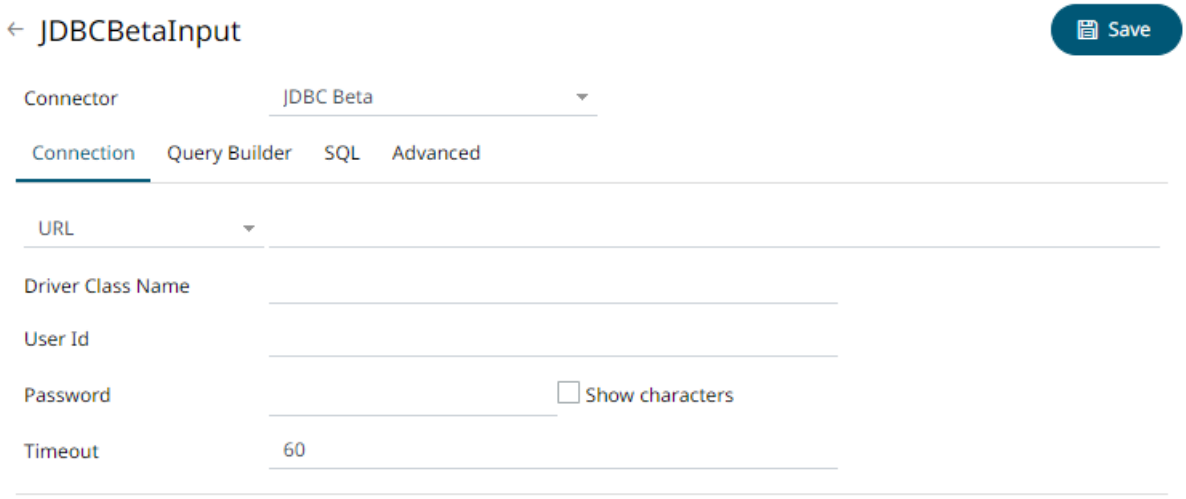
    . The new data source is added in the *Data Sources* list.

Creating JDBC (Beta) Input Data Source

JDBC Beta connector is the new version of JDBC connector and is the recommended connector for any new JDBC connectivity for better performance and configuration UI. Just like JDBC connector, it also allows the retrieval and processing of data from virtually any database, that has a JDBC driver.

Steps:

1. In the *New Data Source* page, select **Input > JDBC Beta** in the *Connector* drop-down list.



2. On the **Connection** tab, set either of the following connection settings:
    - URL

Enter the *URL* specific to the database's JDBC driver, the *Driver Class Name* specific to the driver, and the *User Id* and *Password*.

Check the **Show Characters** box to display the entered characters.

- JNDI Name

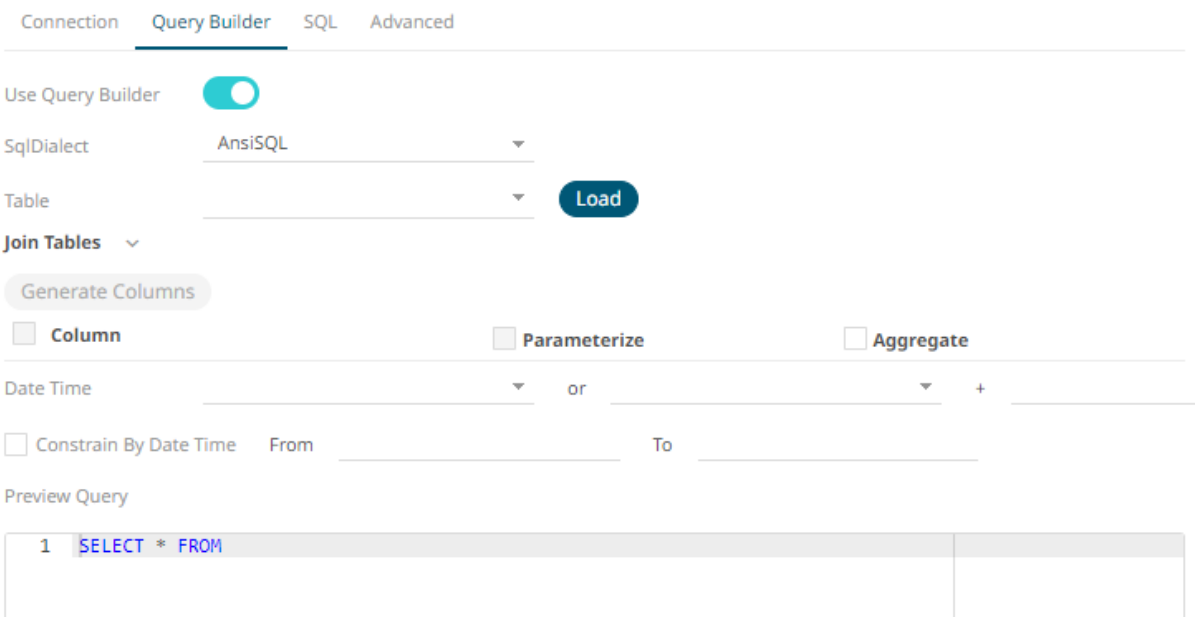
Enter the *JNDI resource name* to be used.

NOTE **The JNDI resource name needs to be on the form:**

`jdbc/[resourcename]`

3. Adjust the *Timeout,* if needed. Default is **60**.
4. Query definition and execution can be done, using either the query builder or freeform SQL. To use the query builder, select the **Query Builder** tab. Otherwise, proceed to step 5.

The **Use Query Builder** option is turned on by default.

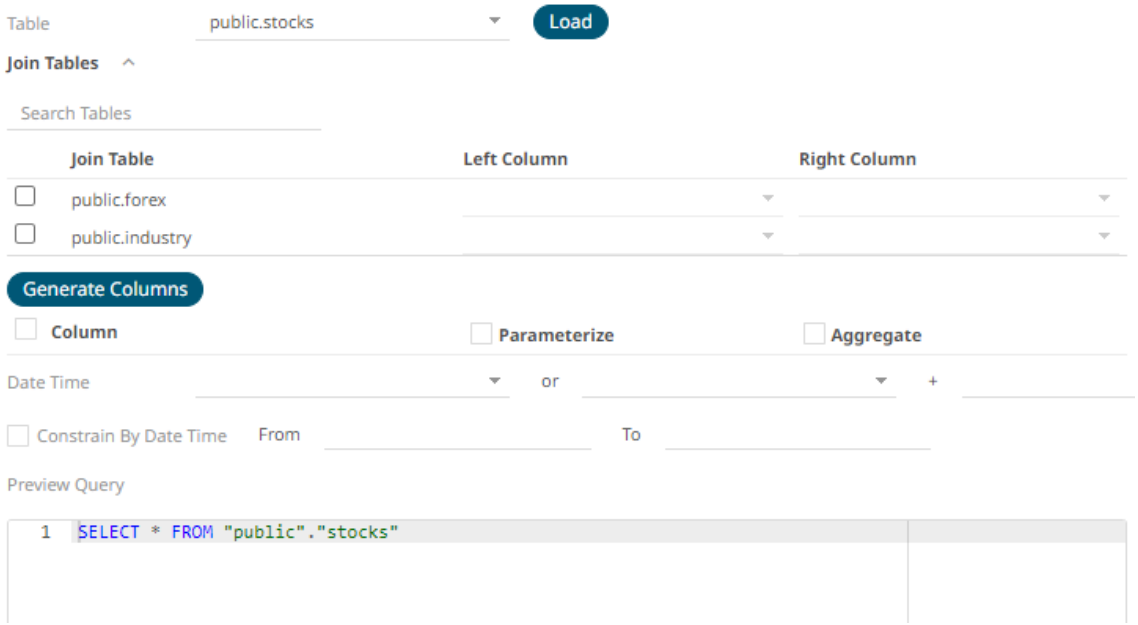


4.1. Select the appropriate *SQL Dialect* in the drop-down list to be able to generate the correct SQL for the required data repository.

4.2. In the *Table* field, click Load to populate the drop-down list with tables. Select a table.

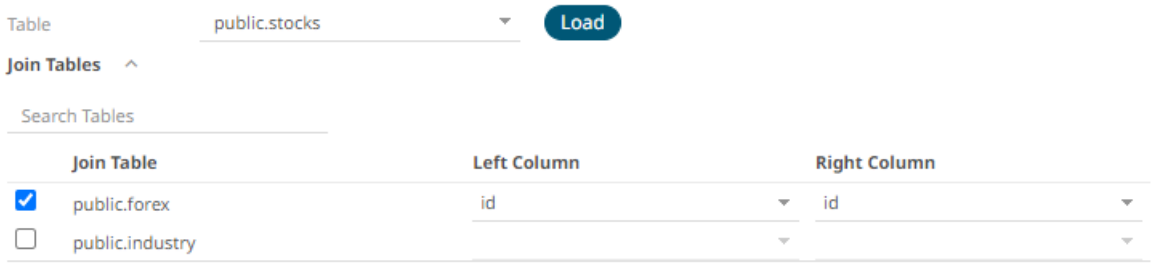
The SQL query is generated and displayed in the *Preview Query* text box.

Also, expanding the *Join Tables* displays the list of tables that you can join.



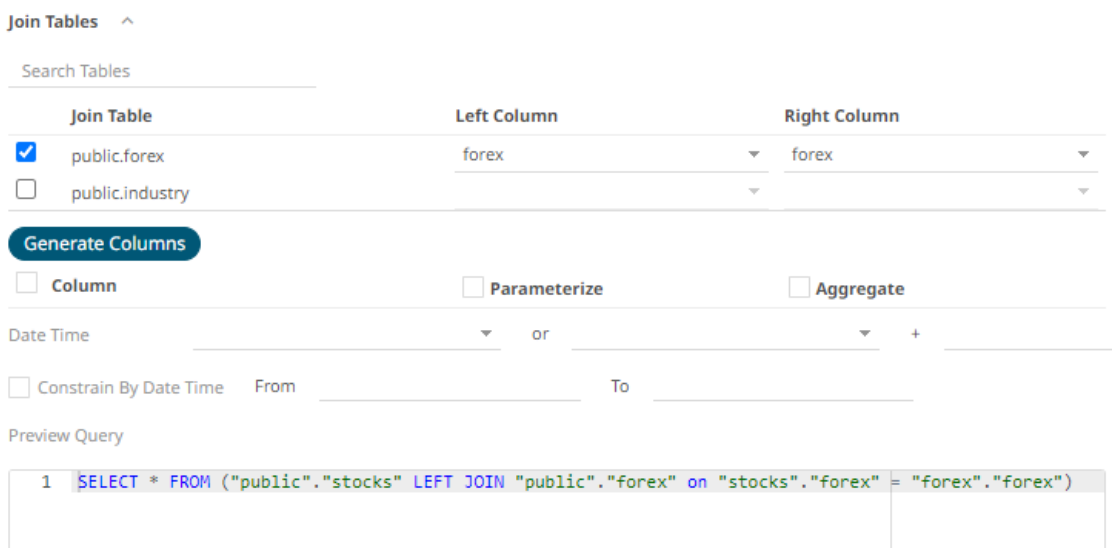
Use *Search Tables* to filter the list.

4.3. Perform a join by checking one or more tables in the list.



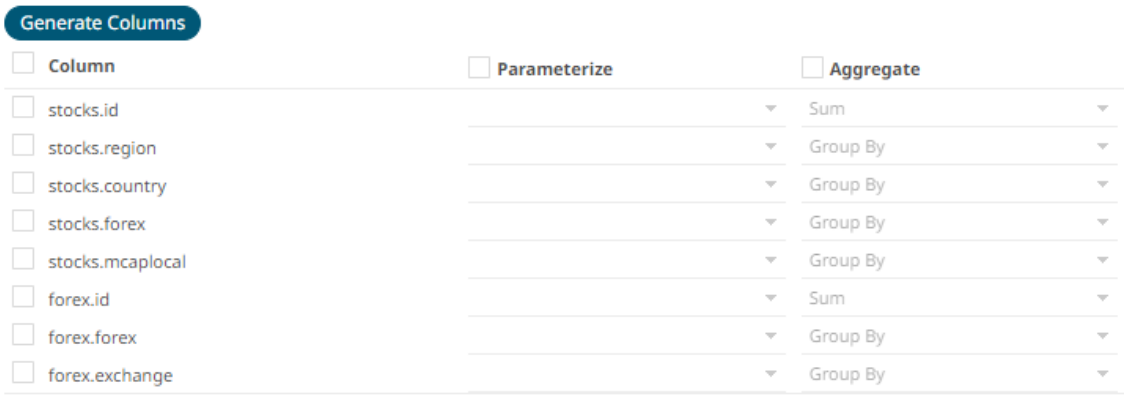
You can also opt to select other common fields.

The SQL query is generated and displayed in the *Preview Query* text box.



4.4. Click

. The columns populate the *Output Column* section.



4.5. Individual columns can be added by checking the corresponding *Column* box in the *Output Column* listing. To select all of the columns, check the topmost box.

The SQL query is generated and displayed in the *Preview Query* text box.

4.6. If the data returned is to be aggregated, then the **Aggregate** box should be checked. For each selected column, the possible aggregation methods are listed including:

- Text Columns: Count, Group By
- Date Columns: Count, Group By
- Numeric Columns: Sum, Count, Min, Max, Group By

4.7. Check the **Parameterize** box and match the parameter to the appropriate column. By default, they will be matched by name.

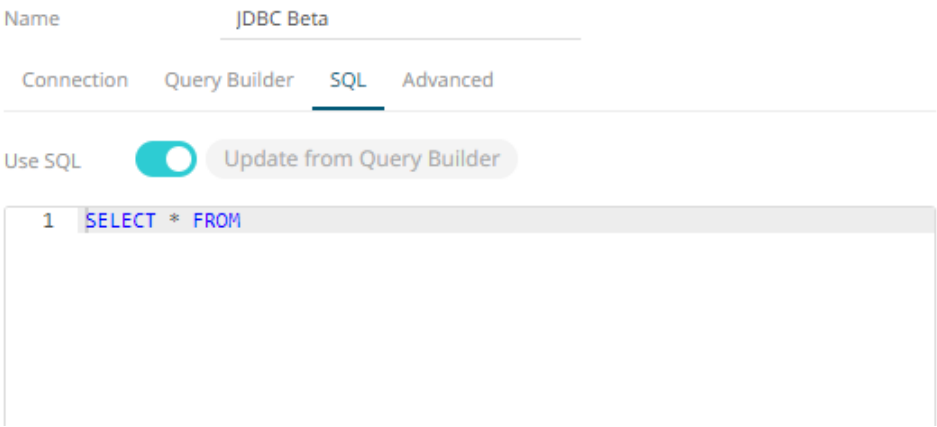
The appropriate SQL Query is updated in the *Preview Query* text box.

4.8. If the data is to be filtered or aggregated on Date/Times, then a valid *Date Time* field needs to be selected from either a single Date/Time field, or a compound column created from a selected *Date* and a selected *Time* column.

4.9. Check the **Constrain by Date Time** box, and enter *From* and *To* Date/Time constraints that are assumed to be in this time zone for incorporation into the query.

If the query is to filter/constrain the results on Date/Time, the constrain sections are completed.

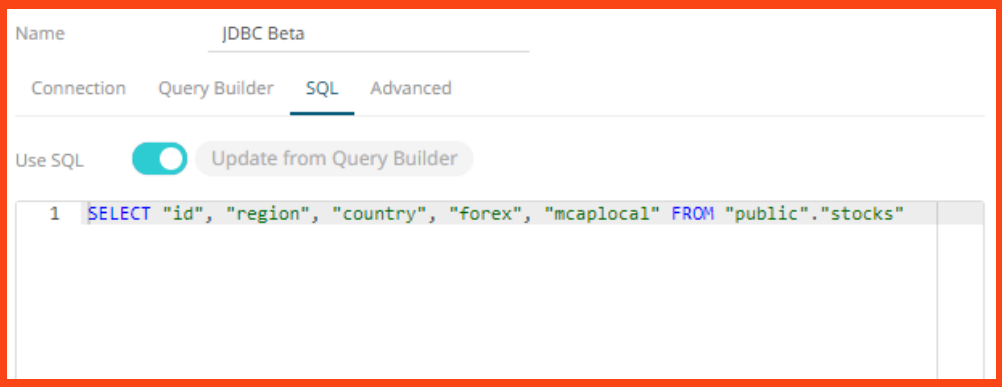
5. To use freeform SQL, select the **SQL** tab and turn on **Use SQL** toggle button.



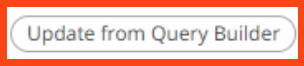
5.1. Modify the SQL-like query language in the *User Query* text box.

NOTE

- **If you initially used the Query Builder then switched to the freeform SQL option, the content of *Preview Query* is copied to the *User Query* text box.**



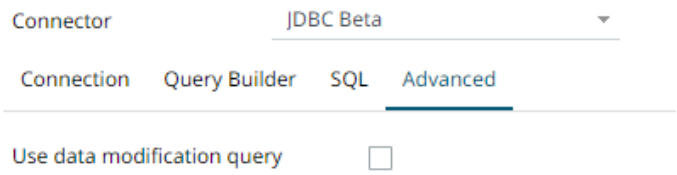
- **Switching back to the Query Builder, the *Preview Query* is updated, keeping the *User Query* unmodified.**

- 

is enabled when *User Query* is non-empty and different from *Preview Query*.

  Click this button to update the *User Query* from the query builder.

6. Select the **Advanced** tab.



Set the following property, if needed:

Property	Description
Use data modification query	Signals that the table is created for writing data. This property is also used for filtering out target data tables for further data update action configuration

7. Click

. The new data source is added in the *Data Sources* list.

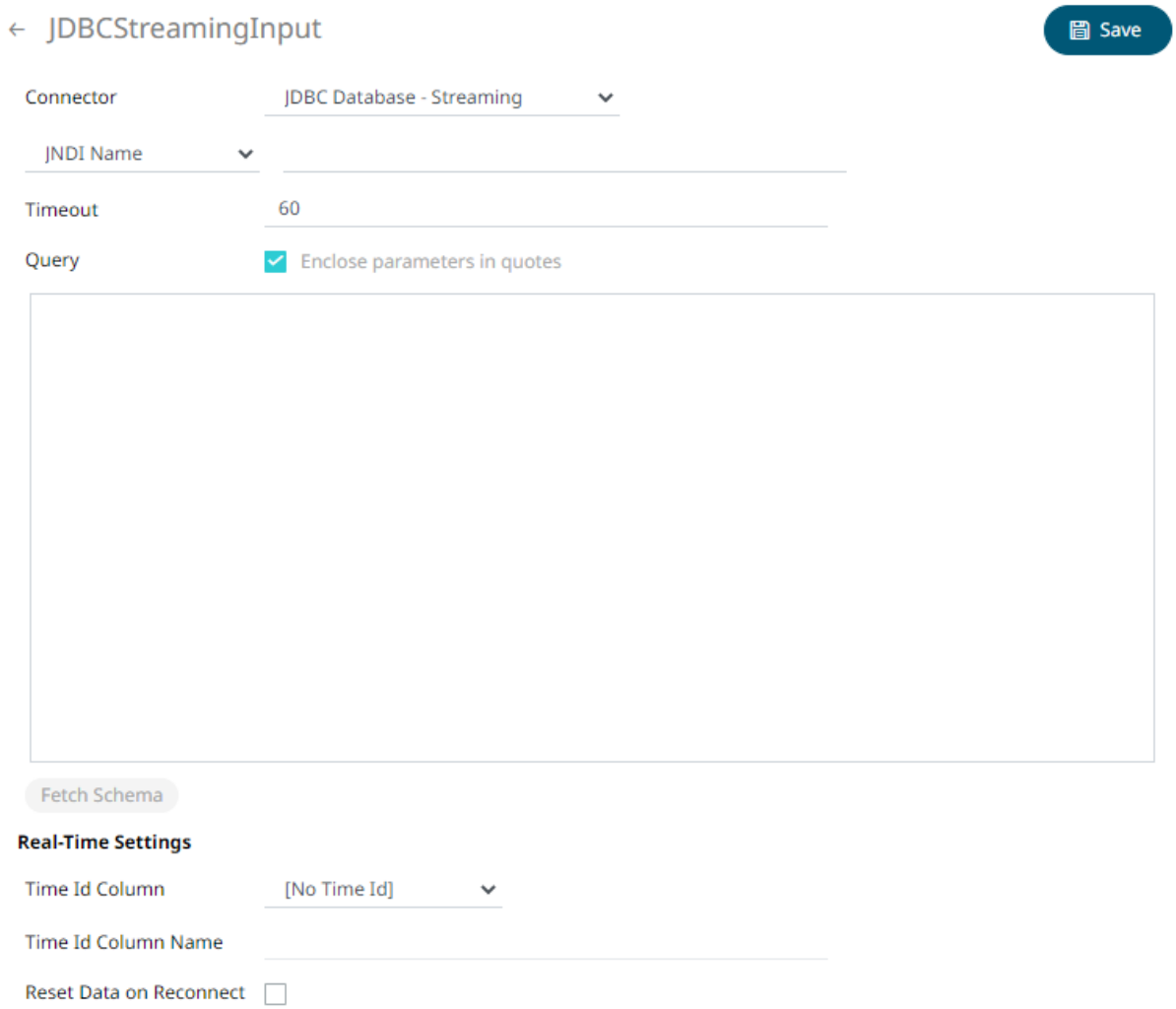
Creating JDBC Database – Streaming Input Data Source

The JDBC Database -Streaming connector allows subscription to a set of data, upserting existing received values in a JDBC SQL Database, by running micro batched queries.

The database must have the appropriate JDBC driver .jar files and JNDI connections.

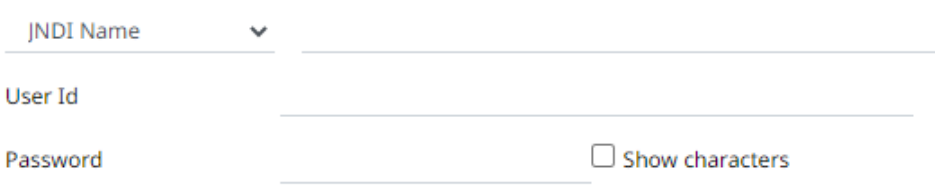
Steps:

1. In the *New Data Source* page, select **Input > JDBC Database – Streaming** in the *Connector* drop-down list.



2. You can either select:

   - JNDI Name



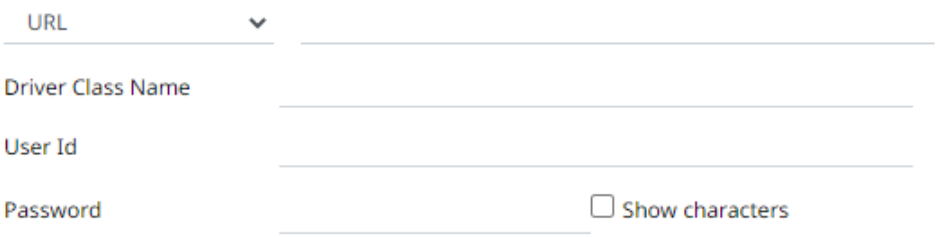
   Enter the *JNDI resource name* to be used, then the *User Id* and *Password*.

   Check the **Show Characters** box to display the entered characters.

NOTE **The JNDI resource name needs to be on the form:**

`java:/comp/env/jdbc/[resourcename]`

- URL



Enter the *URL* specific to the database's JDBC driver, the *Driver Class Name* specific to the driver, and the *User Id* and *Password*.

Check the **Show Characters** box to display the entered characters.

3. Enter the *Timeout* or the length of time to wait for the server response. Default is **60**.
4. Enter the *Query*, which can contain parameters in a similar manner to the database connector.
5. Select whether the parameters should be automatically enclosed in quotes, by checking the **Enclose parameters in quotes** box.
6. Click

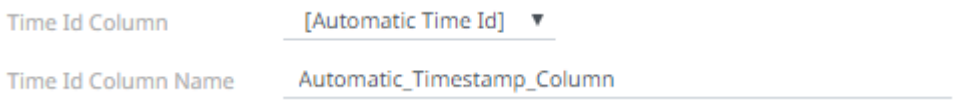
to retrieve the schema of the configured subscription.

   This populates the *Id Column* with the set of columns from the schema of type `sym` and the text array such as Character/Boolean/GUID, etc. The selected *Id Column* can be used to select a key column to manage data updates and inserts.

   NOTE: Every message definition needs a text column to be defined as the Id column. By default, only the latest data will be loaded into memory.

   Furthermore, a streaming time series window can be generated by creating a compound key with the *Id Column*, plus a separately specified *Time ID* column. This *Time ID* column can be from the source dataset, or alternatively automatically generated.

   If the *Time Id column* is selected, then a scrolling time window can be specified.



   For **Automatic Time Id**, define the *Time Id Column Name*.

   As new data arrives from the subscription, new time slices will automatically be added, and old ones will be deleted.

   If a new Id is received, a new row is added to the in-memory data set representing the JDBC Database - Streaming topic subscription. While if an existing ID is received, an existing row is updated.

7. The time zone of input parameters and output data is by default unchanged. Changing the time zone is supported through the *Show in Timezone* list box, based on the assumption that the data is stored in UTC time and outputs are presented in the selected time zone.

8. Define the Real-time Settings.

9. Click

. The new data source is added in the *Data Sources* list.

Creating a JSON Input Data Source

The JSON connector allows the retrieval and processing of JSON files, either from a disk, a Text, or from a defined URL.

Steps:

1. In the *New Data Source* page, select **Input > Json** in the *Connector* drop-down list.

2. Select the JSON *File Source*.

3. Select either the period (.) or comma (,) as the *Decimal Separator*.

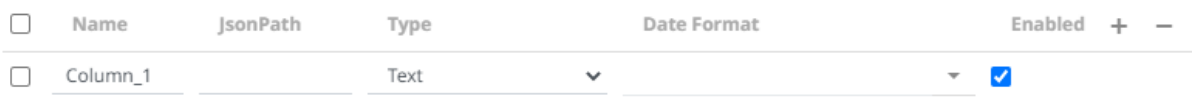
4. Click

to the fetch the schema based on the connection details. Consequently, the list of columns with the data type found from inspecting the first 'n' rows of the input data source is populated and the Save button is enabled.

6. You can also opt to load or save a copy of the column definition.

7. Click + to add columns and specify their properties:

Property	Description
Name	The column name of the source schema.
Json Path	The Json Path of the source schema.
Type	The data type of the column. Can be a **Text, Numeric,** or **Time**
Date Format	The format when the data type is **Time**.
Enabled	Determines whether the message field should be processed.



To delete a column, check its ☐ or all the column entries, check the topmost ☐, then click −.

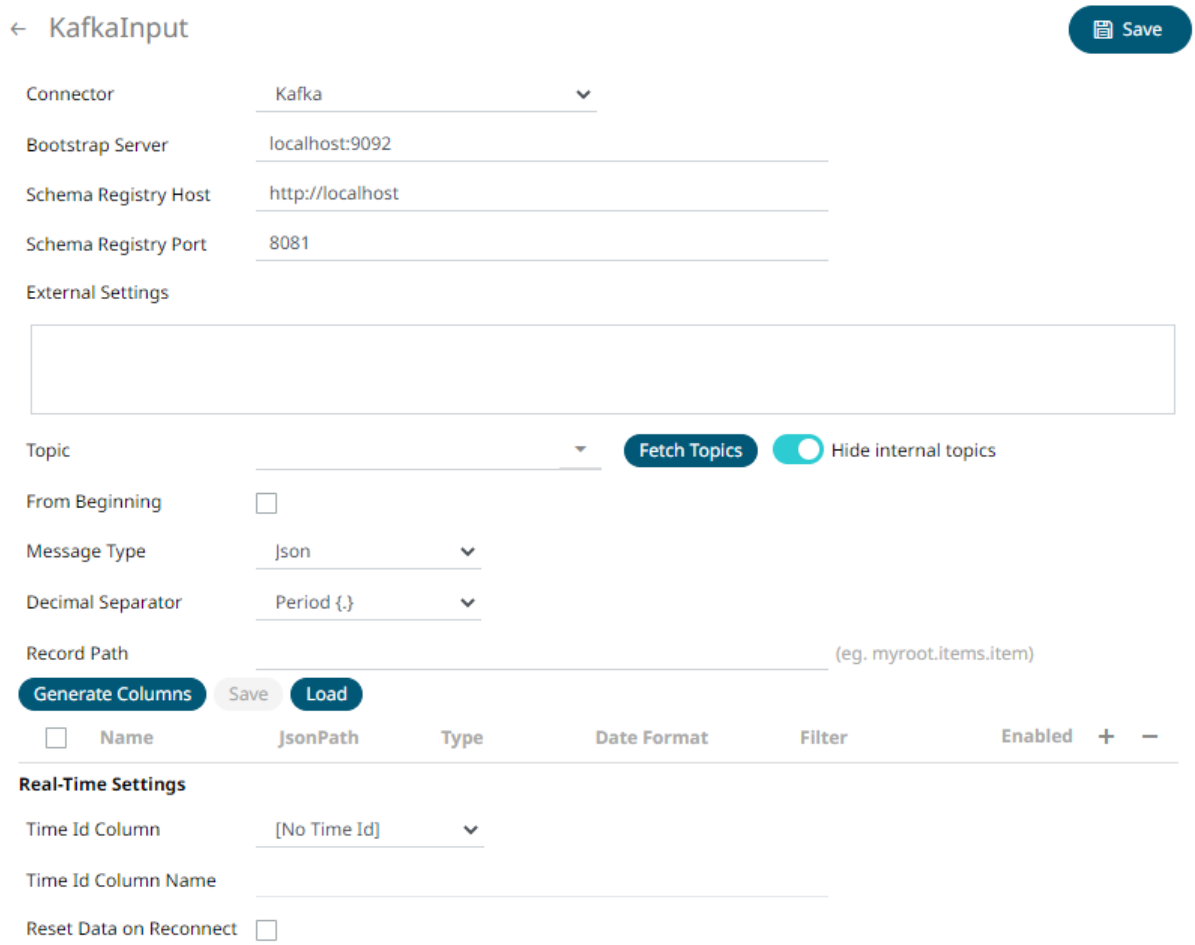
8. Click Save. The new data source is added in the *Data Sources* list.

Creating Apache Kafka Input Data Source

Allows Panopticon Streams to subscribe to Kafka topics on an external cluster.

Steps:

1. In the *New Data Source* page, select **Input > Kafka** in the *Connector* drop-down list.



2. Enter the connection details:

Property	Description
Bootstrap Server	List of host/port pairs of Kafka servers used to bootstrap connections to a Kafka cluster. By default, the value is `localhost:9092,broker:29092`. However, this can be overridden by specifying another bootstrap server in the *External Settings* text box (as specified in step 3).
Schema Registry Host	Where the Schema Registry is located. This can be in a different location from the Kafka cluster.
Schema Registry Port	The port number of the schema registry which provides the serving layer for the metadata. Default is **8081**.

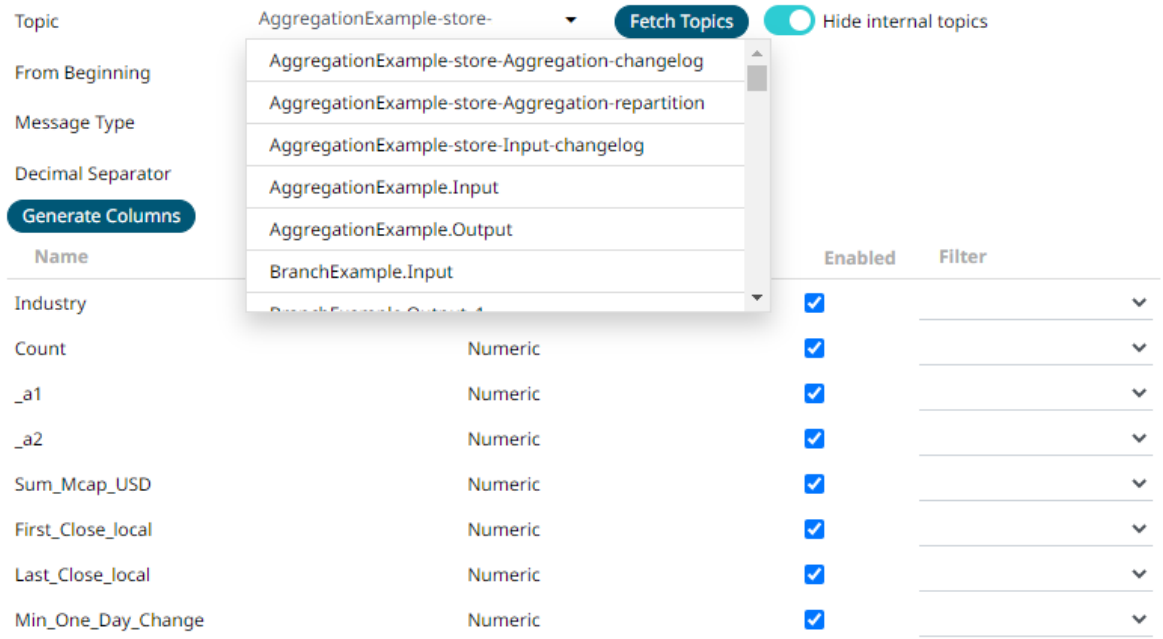
3. Enter the *External Settings* to support authentication (i.e., username and password). Note that if the bootstrap server is not secure, then there is no need to authenticate and you may leave this text box blank.

   Below is an example of system settings for an SASL authentication:

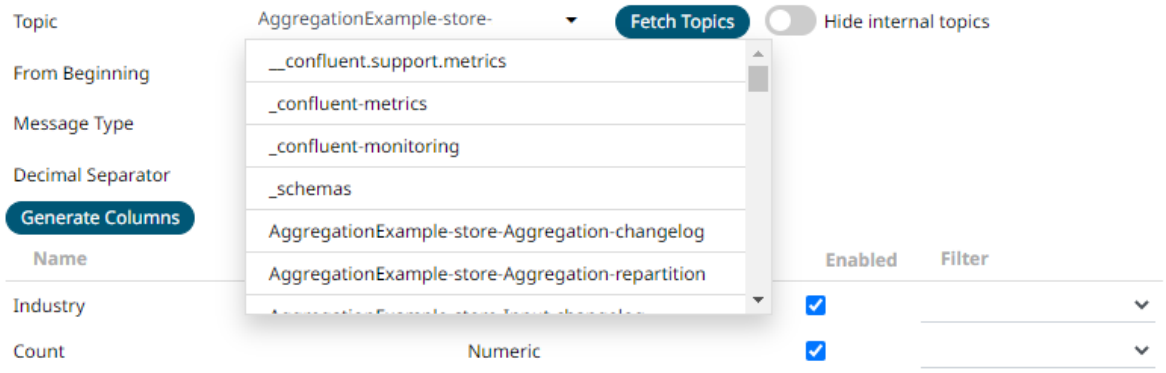
```
bootstrap.servers=localhost:9093
sasl.jaas.config=\
  org.apache.kafka.common.security.plain.PlainLoginModule
required \
    username="dwchuser" \
    password="dwchpwd";
```

4. Click **Fetch Topics.** The first topic in the *Topic* drop-down list is selected and the schema is displayed.

   By default, the **Hide Internal Topics** toggle button is enabled and the **Avro** message type is selected.



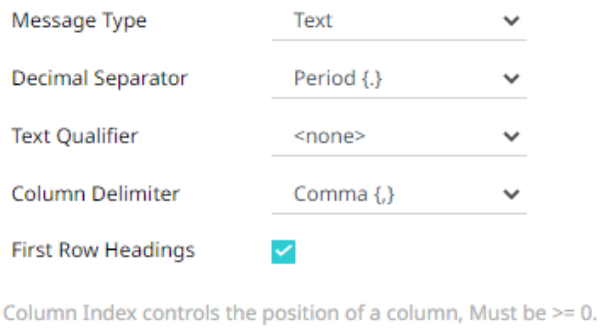
Tap the slider to turn it off. The internal Kafka topics are also displayed in the drop-down list.



Click the drop-down list to search and select the desired topic.

For non-Avro topics, select the *Message Type*: **Fix**, **JSON**, **Text**, **XML,** or **Protobuf**.

- If **Text** is selected, confirm the **Text Qualifier**, **Column Delimiter**, and if the first row of the message includes column headings.



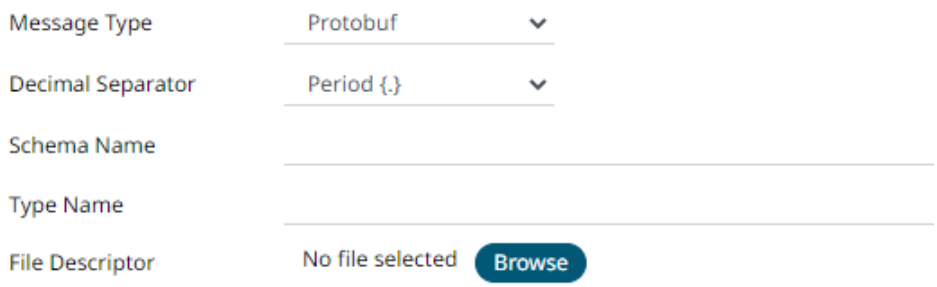
Property	Description
Text Qualifier	Specifies if fields are enclosed by text qualifiers, and if present to ignore any column delimiters within these text qualifiers.
Column Delimiter	Specifies the column delimiter to be used when parsing the text file.
First Row Headings	Determines if the first row should specify the retrieved column headings, and not be used in data discovery.

- If **JSON** is selected, enter *the Record Path* which allows the identification of multiple records within the JSON document (e.g., **myroot.items.item**).

Property	Description
Record Path	The record path that will be queried by the connector's path (e.g., **myroot.items.item**) .

- If **Protobuf** is selected, confirm the **Decimal Separator,** and enter the *Schema Name* and *Type Name*.

  Then click Browse to select the **File Descriptor** (`.desc` file) in the *Open* dialog.

Property	Description
Schema Name	The Protobuf schema.
Type Name	The message of Protobuf type that will be sent to Kafka.
File Descriptor	The `FileDescriptorSet` which: • is an output of the protocol compiler. • represents a set of `.proto` files, using the `--descriptor_set_out` option.

5. Check the **From Beginning** box to subscribe from the beginning to the latest messages.

   If un-checked, you will only be subscribed to the latest messages.

6. Select either the period (.) or comma (,) as the *Decimal Separator*.

   NOTE **Prepend 'default:' for the elements falling under default namespace.**

7. Click Generate Columns to fetch the schema based on the connection details. Consequently, the list of columns with the data type found from inspecting the first 'n' rows of the input data source is populated and the **Save** button is enabled.

8. For non-Avro message types, except **Protobuf**, click + to add columns to the Kafka connection that represent sections of the message. Then enter or select:

Property	Description
Name	The column name of the source schema.
Fix Tag/JsonPath/Text Column Index/XPath	The Fix Tag/JsonPath/Text Column Index/XPath of the source schema.
Type	The data type of the column. Can be a **Text**, **Numeric**, or **Time**
Date Format	The format when the data type is **Time**.

Filter	Defined parameters that can be used as filter. Only available for Avro, JSON, Text, and XML message types.
Enabled	Determines whether the message field should be processed.

NOTE **To parse and format times with higher than millisecond precision, the format string needs to end with a period followed by sequence of upper case S. There can be no additional characters following them.**

For example: yyyy-MM-dd HH:mm:ss.SSSSSS

9. You can also opt to load or save a copy of the column definition.
10. Define the Real-time Settings.
11. Click

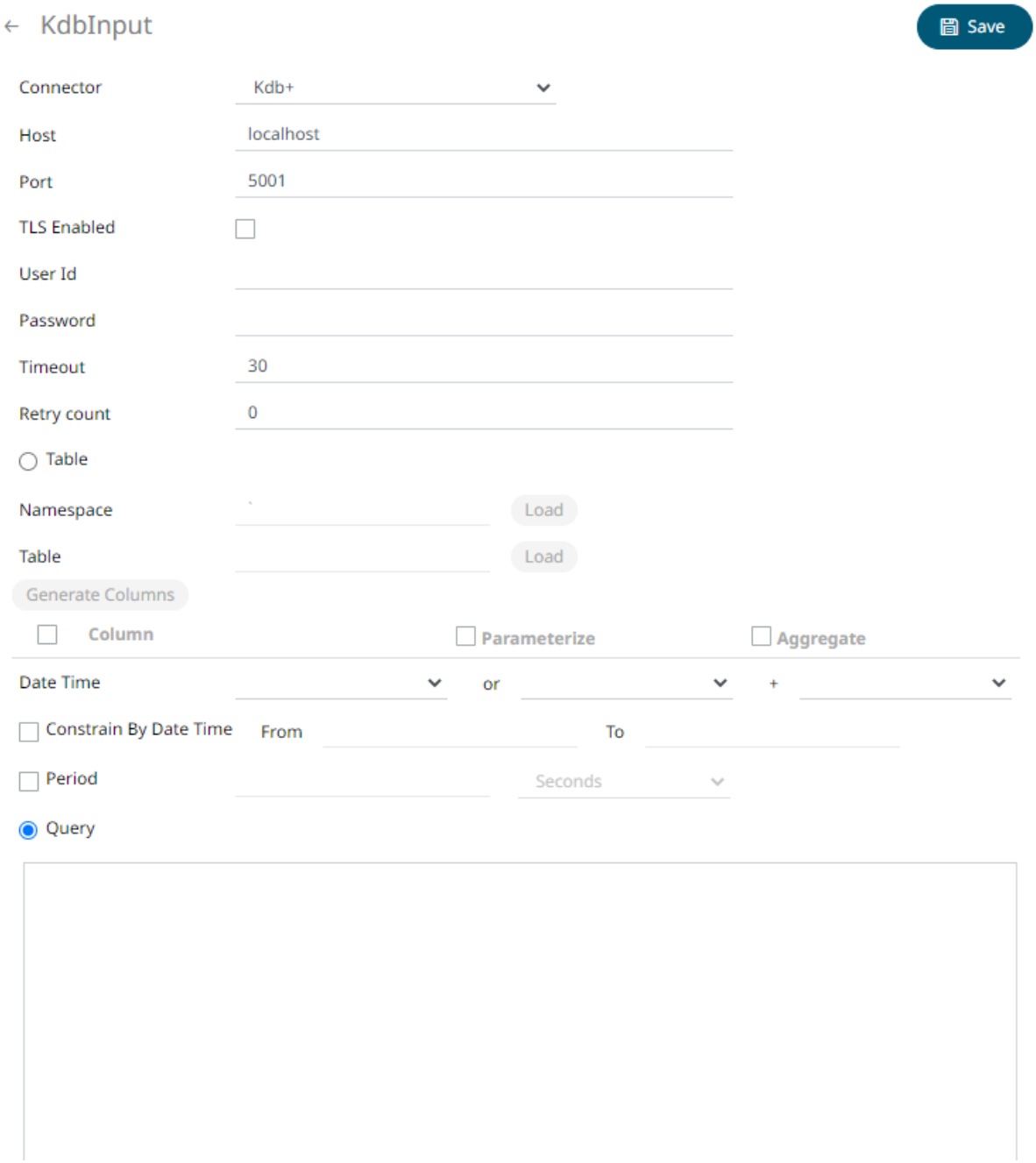
. The new data source is added in the *Data Sources* list.

Creating Kx kdb+ Input Data Source

The Kx kdb+ input data source allows connection to the Kx kdb+ databases on a polled basis.

Steps:

1. In the *New Data Source* page, select **Input > Kdb+** in the *Connector* drop-down list.

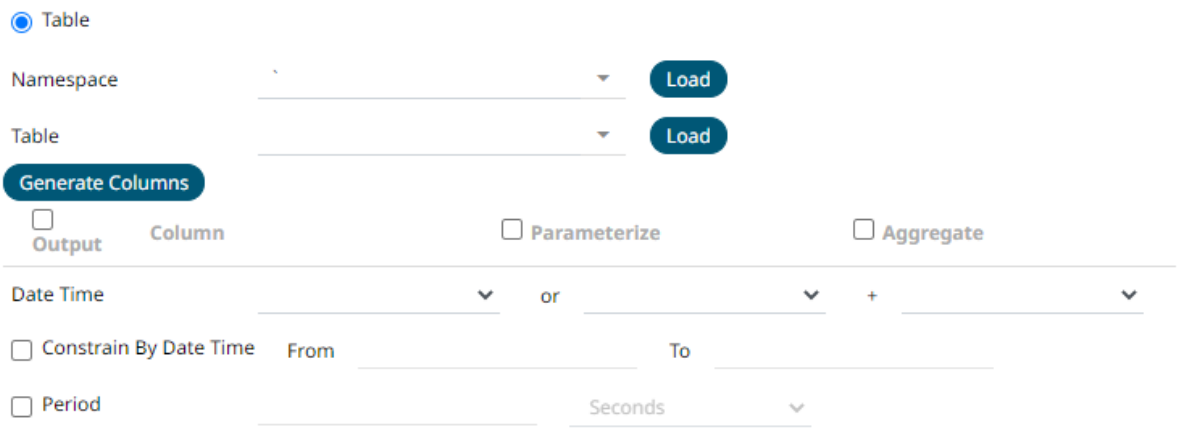


2. Enter the following properties:

Property	Description
Host	Kx kdb+ host address.
Port	Kx kdb+ host port. Default is **5001**.
TLS Enabled	Ensure to check if you have started q with TLS only.
User Id	The user Id that will be used to connect to Kx kdb+.

Password	The password that will be used to connect to Kx kdb+.
Timeout	The length of time to wait for the server response in seconds. Default is **30**.
Retry Count	For long running queries, a query timeout can be specified to prevent the server from locking up. Default is **0**.

3. When **Table** is selected, the section below is enabled:



The *Namespace* drop-down is an editable combo box.

You can either:

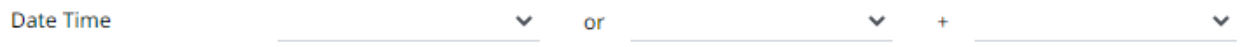
- click Load and select a namespace from the list of all root level namespaces. By default, the selected namespace will be root (backtick `).
- For nested namespaces, enter them in the *Namespace* box (e.g., `panopticon.test`) to get the tables that were created under these namespaces.

4. On the *Table* field, click Load to populate the drop-down list with tables and views. Select a table or view.
5. Click Generate Columns. The columns of the selected table or view populates the *Output Column* section.
6. Individual columns can be added by checking the corresponding *Column* box in the *Output Column* listing.
7. If the data returned is to be aggregated, then the **Aggregate** checkbox should be selected. For each selected column, the possible aggregation methods are listed including:
   - Text Columns: Group By
   - Date Columns: Count, Min, Max, Group By
   - Numeric Columns: Sum, Count, Min, Max, Group By

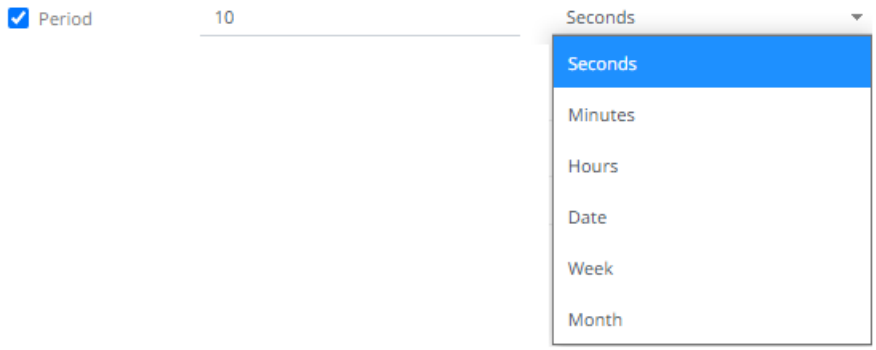
   In addition, the qSQL query is generated and displayed on the *Query* text box.
8. Check the **Parameterize** checkbox and match the parameter to the appropriate column. By default, they will be matched by name.

The appropriate qSQL query is updated on the *Query* text box. This shows the default parameter value for the preview, and at run time the qSQL will be updated to whatever the parameter value is.

9. If the data is to be filtered or aggregated on Date/Times, then a valid *Date Time* field needs to be selected from either a single Date/Time field, or a compound column created from a selected *Date* and a selected *Time* column.



10. Check the **Constrain by Date Time** box and enter *From* and *To* Date/Time constraints.
11. In Kx kdb+, you can modify the query to regroup the aggregated data per time units (i.e., Seconds, Minutes, Hours, Date, Week, Month). Check the **Period** box, enter the time duration and click ✓ then select the time unit.



12. Click the **Query** radio button to enable the text box and modify the qSQL query language.
13. Select the *Flatten List Limit*.

    This allows retrieval of the first 'n' items in the list and produce new columns in the output schema with a dot notation.

    For example, if there are two nested fields (BidPrices and OfferPrices) and the flatten list limit selected is five, then the output schema will be:

    BidPrices.1, BidPrices.2, BidPrices.3, BidPrices.4, BidPrices.5, OfferPrices.1, OfferPrices.2, OfferPrices.3, OfferPrices.4, OfferPrices.5

    If there are less than five items in the list, then the values will be null.

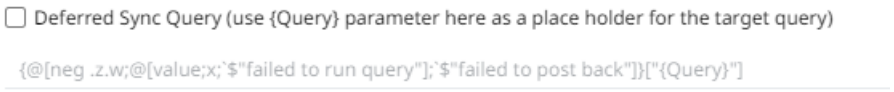
> **NOTE** **Currently, this feature works for the** Service **subscription type. Also, it only flattens numeric columns.**

14. Check **Pass to function** box to activate a connection to a server using a proxy. Enter the value.
15. You may also define a Deferred Sync Query.
16. Click

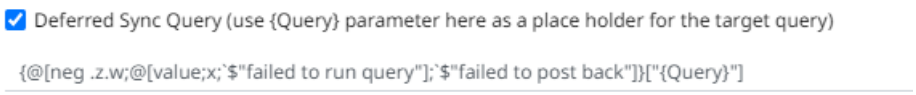
. The new data source is added in the *Data Sources* list.

Kx kdb+ - Deferred Sync Query

The Deferred Sync Query feature allows the Kx kdb+ connector to support synchronous and asynchronous reads. The advantage of using this option is that there is no queue on the Kx kdb+ server side, queries are farmed out to slaves and returned to asynchronous instead.



Checking the *Deferred Sync Query* box would enable the query box:



The {Query} parameter is used as a place holder for the target query that is defined in the *Query* builder.

Creating Kx kdb+tick Input Data Source

The Kx kdb+tick input data source allows connection to a Kx kdb+ ticker plant on a real-time streaming basis.

Specifically, it allows Panopticon Streams to subscribe to Kx kdb+tick through the definition of *Service*, *Table*, *Symbol*, or directly through *Functional Subscription*.

Steps:

1. In the *New Data Source* page, select **Input > KDB+ Tick** in the *Connector* drop-down list.

← KdbTickInput

Save

Connector: Kdb+ Tick

Host: localhost

Port: 5010

TLS Enabled: ☐

User Id:

Password:

Subscription Type: (●) Service ( ) Functional Subscription

Subscription Name: .u.sub

Table:

Symbol:

Multiple symbols should be seperated by comma.

Fetch Schema

☑ Constrain subscription to matching symbols [Id Column]

☑ Initialize with historic data

Host: localhost

Port: 5010

TLS Enabled: ☐

User Id:

Password:

Query:

☐ Deferred Sync Query (use {Query} parameter here as a place holder for the target query)

{@[neg .z.w;@[value;x;`$"failed to run query"];`$"failed to post back"]}["{Query}"]

Flatten List Limit:

Real-Time Settings

Time Id Column: [No Time Id]

Time Id Column Name:

Reset Data on Reconnect: ☐

2. Enter the following properties:

Property	Description
Host	Kx kdb+tick host address.
Port	Kx kdb+tick host port. Default is **5010**.
TLS Enabled	Ensure to check if you have started q with TLS only.
User Id	The user Id that will be used to connect to Kx kdb+tick.
Password	The password that will be used to connect to Kx kdb+tick.

NOTE **These properties can be parameterized.**

3. Select either Subscription Type:
   - Service

     Enter the following properties:

     - Subscription Name (e.g., **.u.sub**)

     NOTE **Instead of entering the table and symbol to subscribe against in the Table and Symbol text boxes, you can specify the full subscription syntax in the Subscription Name text box. For example:**

     .u.sub[`table;`symbol]

     To subscribe to the trade table and AAPL, AIG, and DOW symbols, enter this in the Subscription Name text box:

     .u.sub[`trade;`AAPL`AIG`DOW]

     - Table to subscribe against (e.g., **trade**)

     NOTE
     - **You may use just a back tick for the table name, intending to subscribe to all available tables.**
     - **When a table name is not entered in the Table text box, then the Symbol text box is disabled meaning it will not be used while doing subscription.**

     - Symbol to subscribe against (e.g., **AAPL**)

     NOTE **Multiple symbols should be separated by a comma.**

   - Functional Subscription

     Enter the functional subscription that needs to be issued (e.g., .**u.sub[`trade;`]**)

4. Click Fetch Schema to retrieve the schema of the configured subscription.

This populates the *Id Column* with the set of columns from the schema of type sym and the text array such as Character/Boolean/GUID, etc.

5. Check *Constrain subscription to matching symbols* to select the column which contains specific symbols. Otherwise, the filtering against these symbols will not take place.

> **NOTE** **The *Constrain subscription to matching symbols* only lists sym fields. Therefore, if you select a non sym type in the *Id Column,* it is not recommended to select the default value [Id Column] in the *Constrain subscription to matching symbols* drop-down list.**

6. Activate or deactivate *Initialize with historic data*. If unchecked, the data source will only be populated with streaming updates that are subscribed against. If checked, the data source is first initialized against a store of data, after which subscribed streaming updates are then applied.

7. Enter the following information:

   - Host
   - Port
   - User Id
   - Password
   - Query

     These entries can be parameterized.

8. Check *Deferred Sync Query* box to allow the Kxkdb+tick data source to support synchronous and asynchronous reads. The advantage of using this option is that there is no queue on the Kx kdb+tick server side, queries are farmed out to slaves and returned to asynchronous instead.

   The {Query} parameter is used as a place holder for the target query that is defined in the Query builder.

9. Select the *Flatten List Limit*.

   This allows retrieval of the first ‘n’ items in the list and produce new columns in the output schema with a dot notation.

   For example, if there are two nested fields (BidPrices and OfferPrices) and the flatten list limit selected is five, then the output schema will be:

   BidPrices.1, BidPrices.2, BidPrices.3, BidPrices.4, BidPrices.5, OfferPrices.1, OfferPrices.2, OfferPrices.3, OfferPrices.4, OfferPrices.5

   If there are less than five items in the list, then the values will be null.

> **NOTE** **Currently, this feature works for the** Service **subscription type. Also, it only flattens numeric columns.**

10. Define the Real-time Settings.

11. Click

. The new data source is added in the *Data Sources* list.

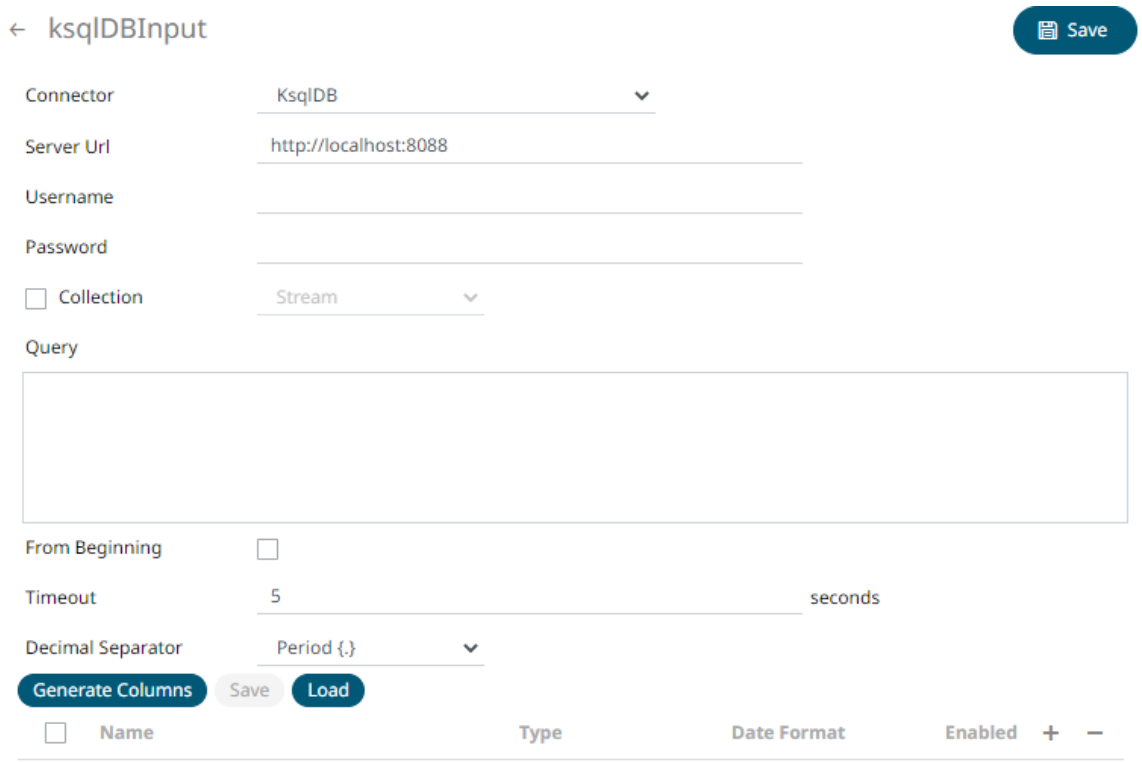
Creating ksqlDB Input Data Source

The ksqlDB connector allows executing ksqlDB pull queries and terminating push queries.

> **NOTE** **Pull queries fetch the current state of a materialized view which is incrementally updated as new events arrive.**

Steps:

1. In the *New Data Source* page, select **Input > ksqlDB** in the *Connector* drop-down list.



2. Enter the following properties:

Property	Description
Server URL	ksqlDB host address.
Username	User Id that will be used to connect to ksqlDB.
Password	Password that will be used to connect to ksqlDB.

3. Check the **Collection** box to enable and select either:
   - Stream

     Immutable and append-only collections which are useful for representing a series of historical facts. Adding multiple events with the same key allows these events to be appended to the end of the stream.
   - Table

Mutable collections. Adding multiple events with the same key allows the table to only keep the value for the last key. This collection is helpful in modeling change over time and often used to represent aggregations.

4. Click Fetch to populate the drop-down list. Select the collection.
5. Enter an SQL-like query language into the *Query* box.
6. Check the *From Beginning* box to subscribe from the beginning to the latest messages.

   From Beginning ☐

   If un-checked, you will only be subscribed to the latest messages.
7. Enter the *Timeout*. Default is **5** (in seconds).
8. Select either the dot (.) or comma (,) as the *Decimal Separator*.
9. Click Generate Columns to the fetch the schema based on the connection details. Consequently, the list of columns with the data type found from inspecting the first 'n' rows of the input data source is populated and the **Save** button is enabled.
10. You can also opt to load or save a copy of the column definition.
11. Click + . A new column entry displays. Enter or select the following properties:

Property	Description
Name	The column name of the source schema.
Type	The data type of the column. Can be a **Text**, **Numeric**, or **Time**
Date Format	The format when the data type is **Time**.
Enabled	Determines whether the message should be processed.

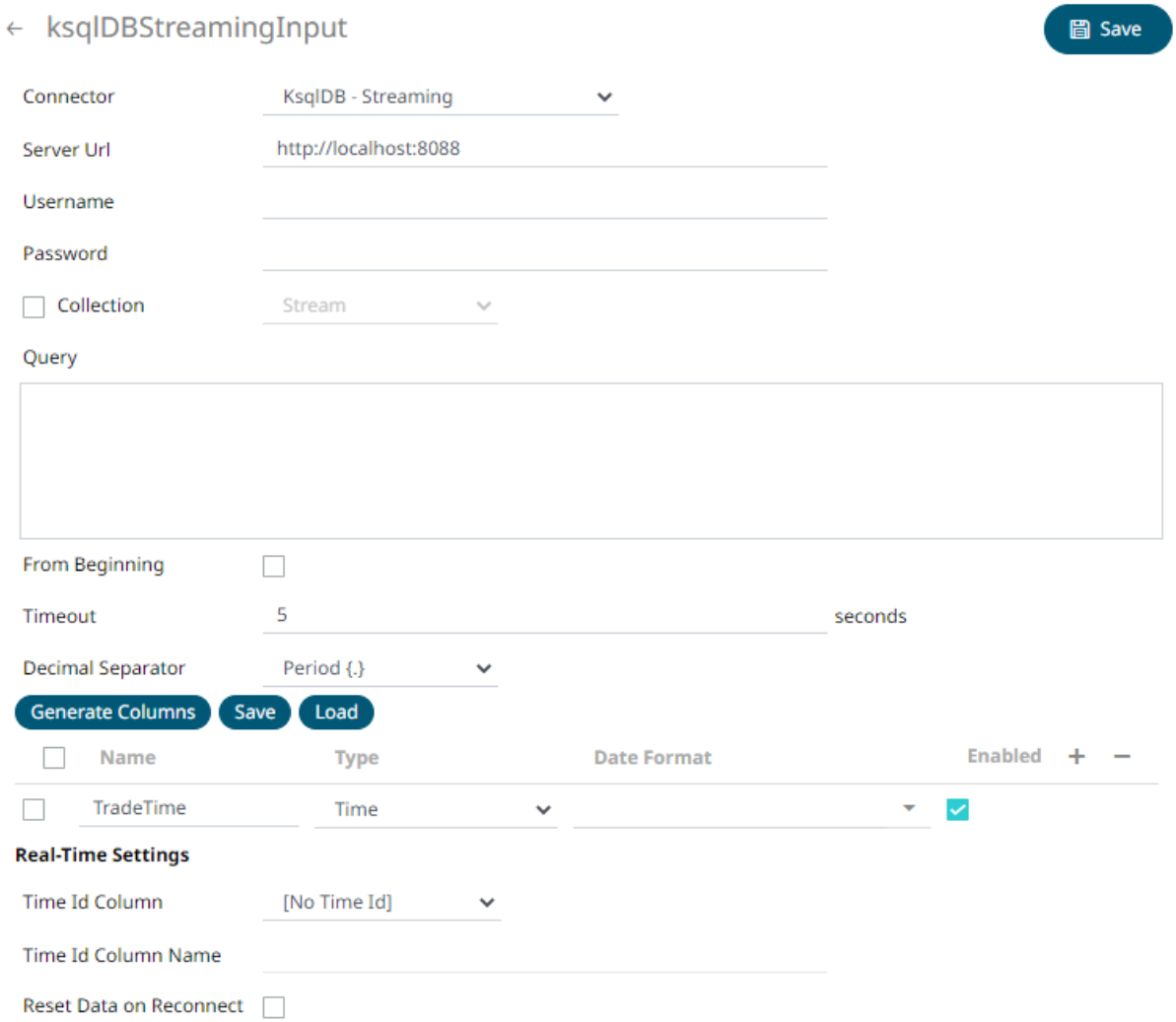
   To delete a column, check its ☐ or all the column entries, check the topmost ☐, then click − .
12. Click Save . The new data source is added in the *Data Sources* list.

Creating ksqlDB - Streaming Input Data Source

The ksqlDB - Streaming connector allows executing ksqlDB push queries.

Steps:

1. In the *New Data Source* page, select **Input > ksqlDB - Streaming** in the *Connector* drop-down list.



2. Enter the following properties:

Property	Description
Server URL	ksqlDB - Streaming host address.
Username	User Id that will be used to connect to ksqlDB - Streaming.
Password	Password that will be used to connect to ksqlDB - Streaming.

3. Check the **Collection** box to enable and select either:

- Stream

  Immutable and append-only collections which are useful for representing a series of historical facts. Adding multiple events with the same key allows these events to be appended to the end of the stream.

- Table

  Mutable collections. Adding multiple events with the same key allows the table to only keep the value for the last key. This collection is helpful in modeling change over time and often used to represent aggregations.

4. Click Fetch to populate the drop-down list. Select the collection.
5. Enter an SQL-like query language into the *Query* box.
6. Check the *From Beginning* box to subscribe from the beginning to the latest messages.

   From Beginning ☐

   If un-checked, you will only be subscribed to the latest messages.
7. Enter the *Timeout*. Default is **5** (in seconds).
8. Select either the dot (.) or comma (,) as the *Decimal Separator*.
9. Click Generate Columns to the fetch the schema based on the connection details. Consequently, the list of columns with the data type found from inspecting the first 'n' rows of the input data source is populated and the **Save** button is enabled.
10. You can also opt to load or save a copy of the column definition.
11. Click + . A new column entry displays. Enter or select the following properties:

Property	Description
Name	The column name of the source schema.
Type	The data type of the column. Can be a **Text**, **Numeric**, or **Time**
Date Format	The format when the data type is **Time**.
Enabled	Determines whether the message should be processed.

   To delete a column, check its ☐ or all the column entries, check the topmost ☐, then click − .
12. Define the Real-time Settings.
13. Click Save **.** The new data source is added in the *Data Sources* list.

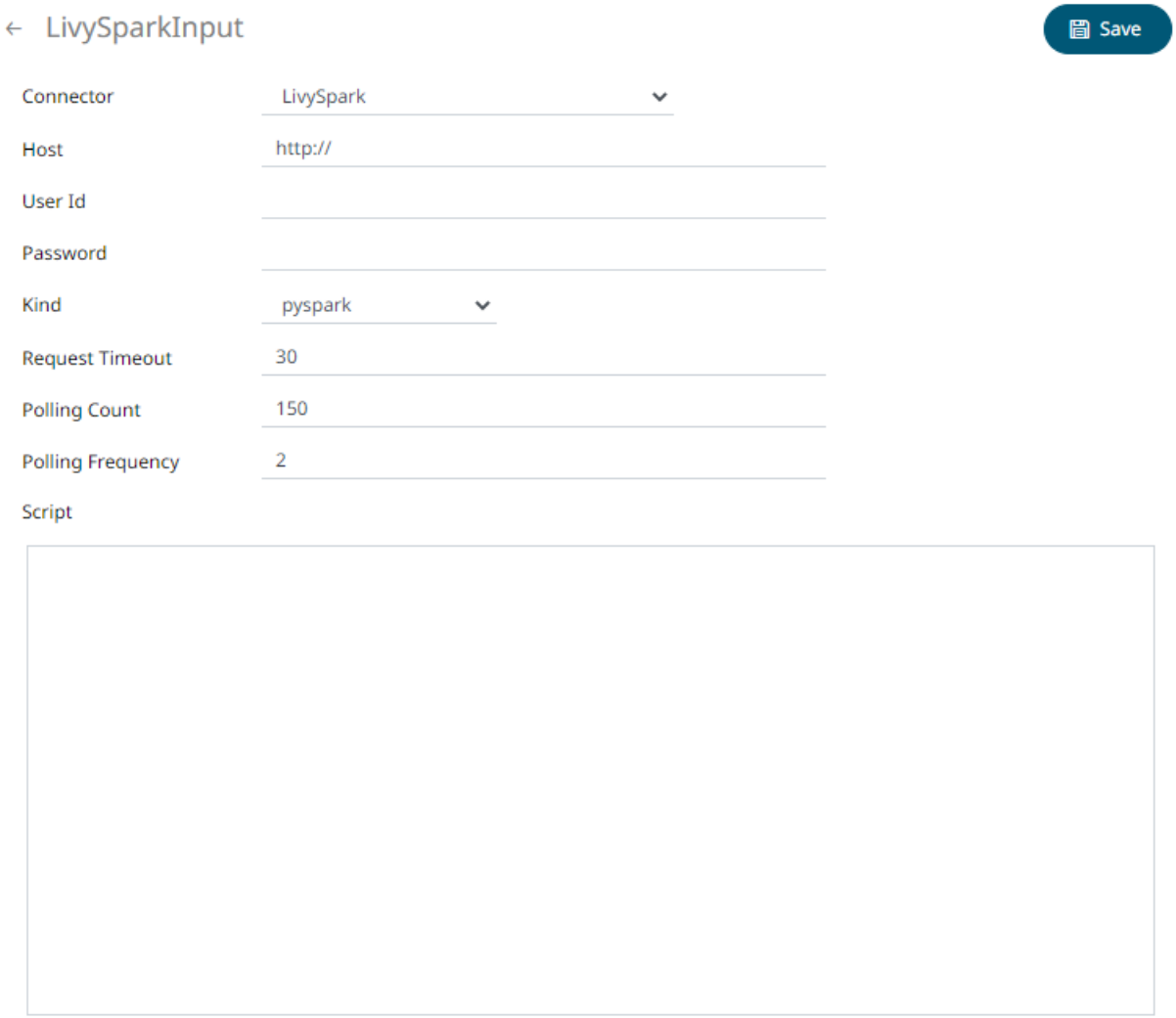
Creating Livy Spark Input Data Source

Livy is an open source REST interface for interacting with Apache Stark. It supports executing snippets of code or programs such as Scala, Python, Java, and R in a Spark context that runs locally or in Apache Hadoop YARN.

The Livy Spark connector allows you to run these codes and fetch the data in Panopticon Streams Server.

Steps:

1. In the *New Data Source* page, select **Input > LivySpark** in the *Connector* drop-down list.



2. Enter or select the following properties:

Property	Description
Host	Livy Spark host address.
User Id	User Id that will be used to connect to Livy Spark.
Password	Password that will be used to connect to Livy Spark.
Kind	Currently, the supported kind of connection to be used is **pyspark** (Interactive Python Spark session).
Request Timeout	Length of time to wait for the server response. Default is **30**.
Polling Count	The number of polling done to the Livy Spark server to check if the status of the app is successful. Default limit is **150**.
Polling Frequency (in seconds)	Frequency of the polling. Default is **2**.
Script	The script to use.

3. Click

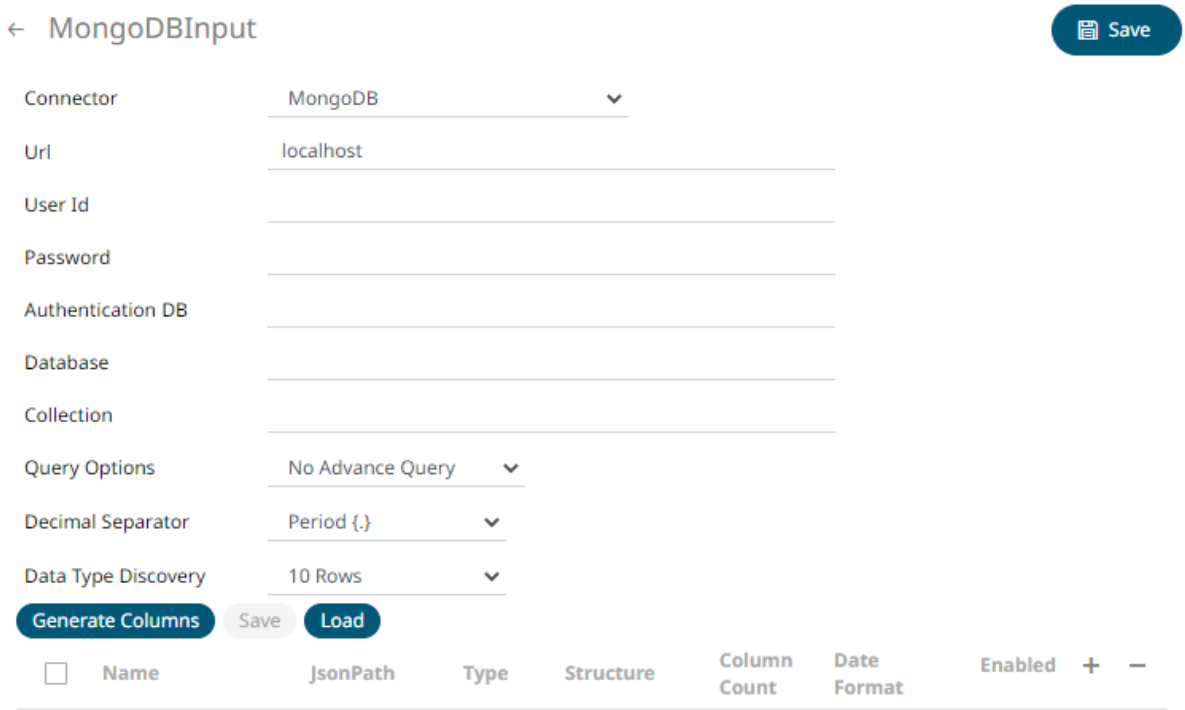
. The new data source is added in the *Data Sources* list.

Creating MongoDB Input Data Source

The MongoDB connector is an interface used to import MongoDB's schema-less BSON documents into a table schema that Panopticon Streams can interpret and analyze. It uses many BSON structure types and MongoDB query features.

Steps:

1. In the *New Data Source* page, select **Input > MongoDB** in the *Connector* drop-down list.

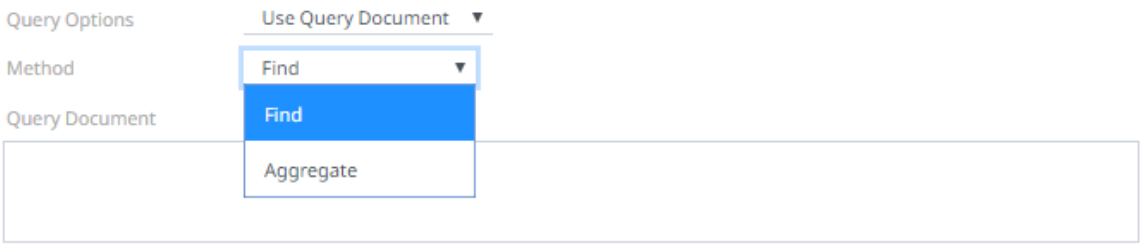


2. Enter the following properties:

Property	Description
URL	Enter either: • **localhost** if the database resides on the same computer, or • enter the IP address and port of the computer where MongoDB is installed (e.g., 192.168.1.1:27017). If no port is specified, the default is **27017**.
User Id	The user Id that will be used to connect to MongoDB.
Password	The password that will be used to connect to MongoDB.
Authentication DB	The database where the user is created.
Database	The database that will be used.
Collection	The collection that will be used.

3. You can also opt to make the Query Document feature of MongoDB to be available in Panopticon Streams. Select **Use Query Document** in the *Query Options* drop-down list.

   This also displays the *Method* drop-down. Select either **Find** (Default) or **Aggregate**.



   When **Aggregate** is selected, you can add all the columns generated by aggregation in the schema.

   In addition, the MongoDB command line interface displays query operations with a JSON style syntax.

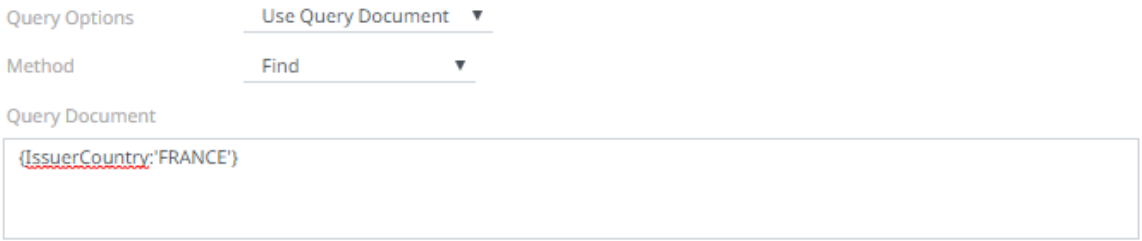
   Enter your desired JSON query document. Refer to http://docs.mongodb.org/manual/tutorial/query-documents/ for more information on the Query Documents feature on MongoDB.

   For example:

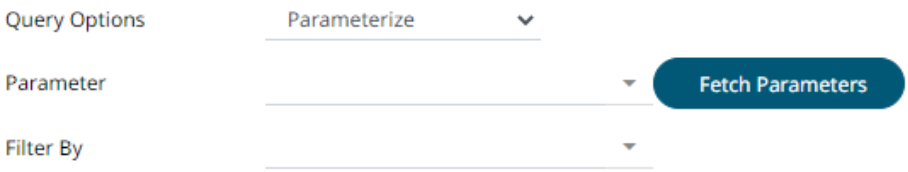
   Queries from the documentation look like this: `db.inventory.find ( {type: "snacks"} )`. The database and collection are already defined in the UI and the *Find* operation is handled in the code. The user only needs to enter the query document:

   { type : "snacks" }

   This query must include surrounding curly braces as well as matching internal braces should the user decide to make a more advanced query.



4. Instead of using **Use Query Document**, select the **Parameterize** query option.

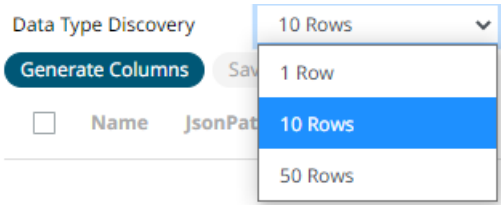


   Click **Fetch Parameters** to populate the *Parameter* drop-down and select a value. Then select what column to filter on in the *Filter By* drop-down.

5. Select either the period (.) or comma (,) as the *Decimal Separator*.

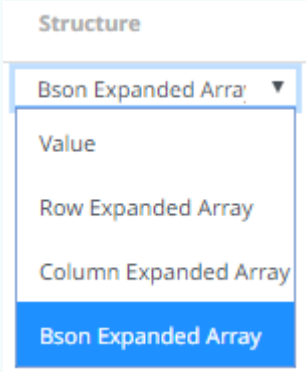
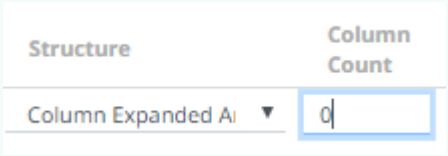
NOTE **Prepend 'default:' for the elements falling under default namespace.**

6. Select the *Data Type Discovery*. This property specifies how many rows to fetch from the input data source, when auto generating the schema after clicking Generate Columns.



7. You can also opt to load or save a copy of the column definition.

8. You can also opt to click +. A new row displays in the JSON list box. Enter the necessary information for each column.

Property	Description
Name	The column name of the source schema. **NOTE:** It is recommended to name the column the same as its JSON path for clarity and uniformity.
JsonPath	The JsonPath of the source schema.
Type	The data type of the column. Can be a **Text**, **Numeric**, or **Time**
Structure	Used for more advanced features and are covered in the Row-Wise Array Expansion, Column-Wise Array Expansion, and Bson-Wise Array Expansion sections. **Value** is the default structure and will always display data regardless of actual structure. 
Column Count	Enabled when **Column-Expanded Array** structure is selected.  Enter the number of columns for the plugin to generate as columns for that array.
Date Format	The format when the data type is **Time**. **NOTE:**

	To parse and format times with higher than millisecond precision, the format string needs to end with a period followed by sequence of upper case S. There can be no additional characters following them. For example: yyyy-MM-dd HH:mm:ss.SSSSSS
Enabled	Determines whether the message field should be processed.

To delete a column, check its ☐ or all the column entries, check the topmost ☐, then click −.

9. Click Save. The new data source is added in the *Data Sources* list.

Row-Wise Array Expansion

MongoDB's BSON document structure can store array data types. In order to interpret that data, the user has to decide how they want those multi-value fields to be displayed.

Row-wise array expansion takes an array of values and expands them in a single column creating a new row for each value in the array. If there are multiple row-expanded arrays in the same document, then the number of rows generated is equal to the largest array size. Additionally, an *Automatic x-axis* column is automatically generated for use as an x-axis value for visualizations using array data.

To use the row-wise array expansion feature, select **Row-Expanded Array** from the *Structure* drop-down box.

This feature will only work for an array data type. If the actual data type in MongoDB is not array or the array is empty, the column will not populate.

Column-Wise Array Expansion

MongoDB's BSON document structure can store array data types. In order to interpret that data, the user has to decide how they want those multi-value fields to be displayed.

Column-wise array expansion takes an array of values and expands them into multiple table columns creating a number of columns equal to an array specific number set by the user. If there are multiple column-expanded arrays in the same document, the combined number of new columns is appended to the end of the table with their respective array indices and the original columns are removed.

To use the column-wise expansion feature, select **Column-Expanded Array** in the *Structure* drop-down box.

The corresponding *Column Count* text box will be enabled and the user can enter the number of columns for the plugin to generate as columns for that array.

Bson-Wise Array Expansion

MongoDB's BSON document structure can store array data types. In order to interpret that data, the user has to decide how they want those multi-value fields to be displayed.

Bson-wise array expansion allows parsing of all the fields of a nested hierarchy in a sub document of a JSON array. During data retrieval, the column value is converted to JSON, and nested columns are flattened based on a JSON parser logic.

To use the Bson-wise expansion feature, select **Bson-Expanded Array** in the *Structure* drop-down box.

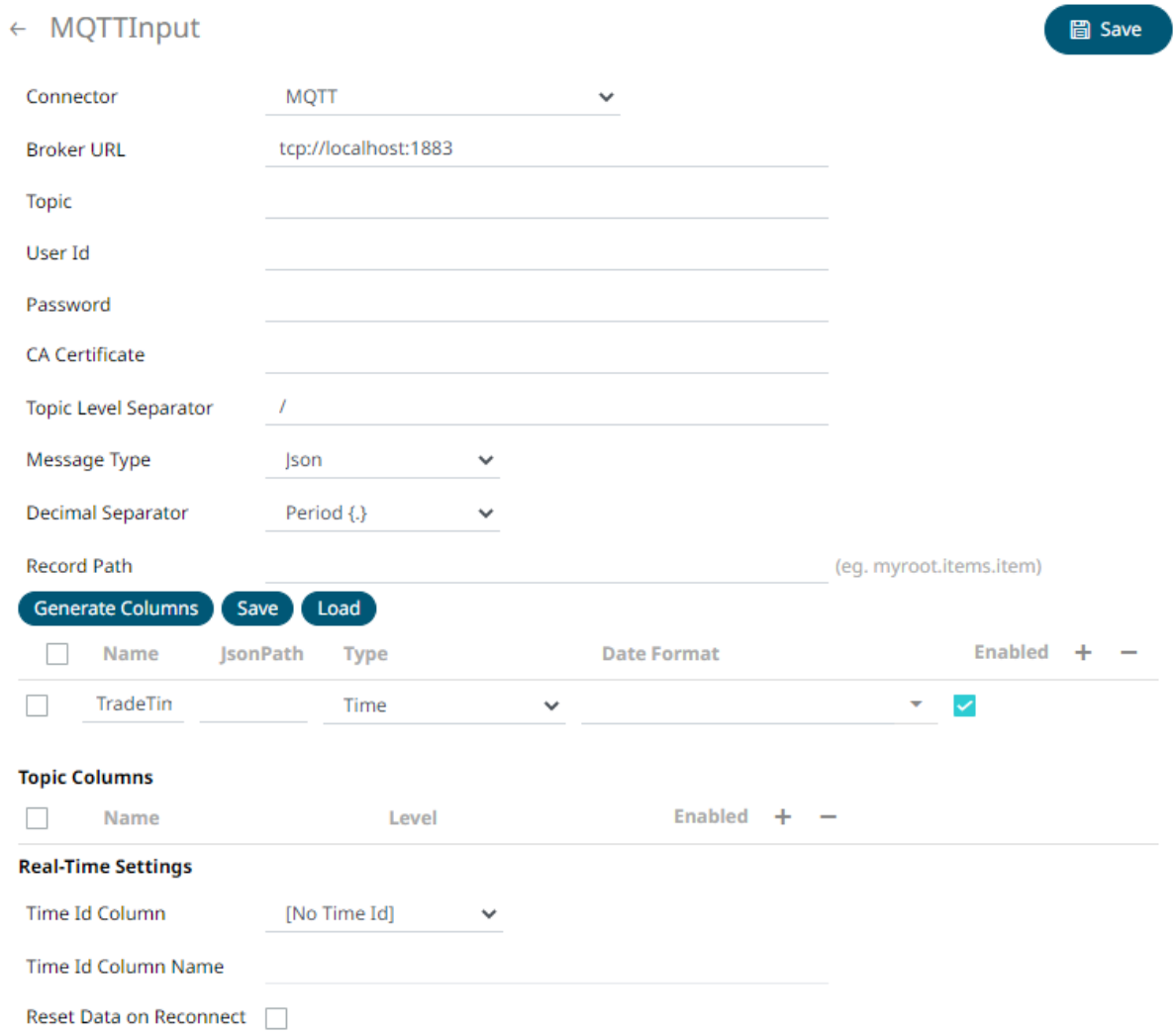
Creating MQTT Input Data Source

The MQTT connector allows:

- connection to MQTT's message bus on a real-time streaming basis.
- Panopticon Streams server to subscribe to FIX, JSON, Text or XML based messages that are published on particular topics. The data format itself is arbitrary, and consequently, the connection includes the message definition.
- encrypted/SSL connections using a generated CA certificate file.

Steps:

1. In the *New Data Source* page, select **Input > MQTT** in the *Connector* drop-down list.



2. Enter the following properties:

Property	Description
Broker URL	The location of the message broker. Default is **tcp://localhost:1883**.
Topic	The topic or the queue physical name.

	Example: level1/level2/level3/level4 etc. **NOTES:** You can also opt to use a wild card in the topic name specification. • The plus sign symbol (+) can be used as a wild card for any value at one specific level. Example: **level1/level2/+/level4** • The hash sign symbol (#) can be used as a wild card for any values across more than one level. Example: **level1/#/level4**
User Id	The user Id that will be used to connect to MQTT.
Password	The password that will be used to connect to MQTT.

3. To allow encrypted connections, enter the *CA Certificate* path of the file.
4. In MQTT, a topic consists of one or more topic levels. Enter the *Topic Level Separator* to use. Default is / (forward slash).
5. Select the *Message Type*.
6. Select either the period (.) or comma (,) as the *Decimal Separator*.

NOTE **Prepend 'default:' for the elements falling under default namespace.**

7. Click Generate Columns to the fetch the schema based on the connection details. Consequently, the list of columns with the data type found from inspecting the first 'n' rows of the input data source is populated and the Save button is enabled.
8. You can also opt to load or save a copy of the column definition.
9. You can opt to click + to add columns to the MQTT connection that represent sections of the message. Then enter or select:

Property	Description
Name	The column name of the source schema.
XPath/JsonPath/Fix Tag/Column Index	The XPath/JsonPath/Fix Tag/Column Index of the source schema.
Type	The data type of the column. Can be a **Text**, **Numeric**, or **Time**
Date Format	The format when the data type is **Time**. **NOTE:** To parse and format times with higher than millisecond precision, the format string needs to end with a period followed by sequence of upper case S. There can be no additional characters following them. For example: yyyy-MM-dd HH:mm:ss.SSSSSS

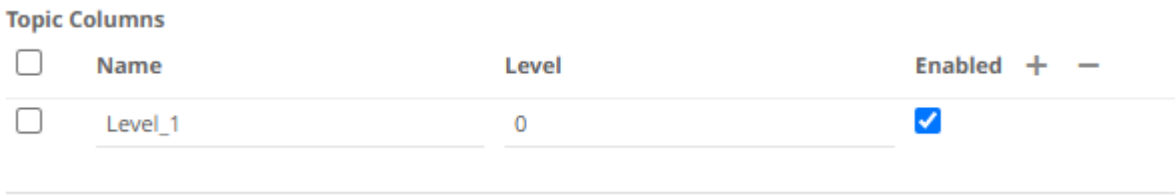
Filter	Defined parameters that can be used as filter. Only available for JSON, Text, and XML message types.
Enabled	Determines whether the message field should be processed.

To delete a column, check its ☐ or all the column entries, check the topmost ☐, then click −.

10. Text for topic levels can be consumed as additional columns into the data table.

    The *Topic Columns* section shows and allows defining data table columns and mapping them to topic hierarchy levels (index based from left, 0 based).

    Like columns from message data, manually add them by clicking +. A new entry displays.



    Name can be any unique topic level within the topic name. The *Level* is the hierarchy level of the topic column.

    Check the *Enabled* box to enable a topic column.

    To delete a topic column, check its ☐ or all the topic column entries, check the topmost ☐, then click −.

11. Define the Real-time Settings.

12. Click Save. The new data source is added in the *Data Sources* list.

Creating MS Excel (Legacy) Input Data Source

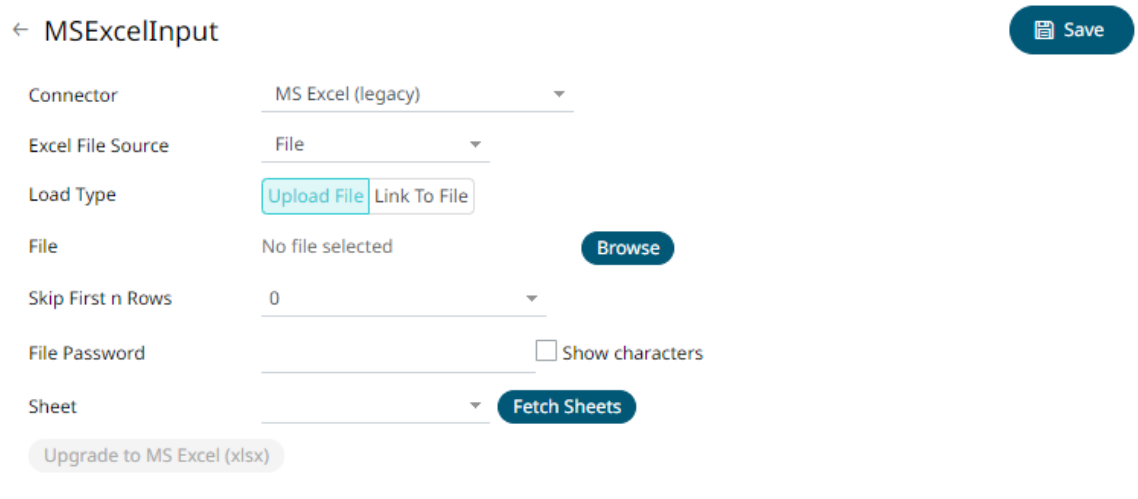
This is the most commonly used data connector when prototyping and is used for retrieving data from MS Excel workbooks or spreadsheets, where for each selected sheet, the first row contains the field/column names, and subsequent rows contain the data.

NOTE **In production use, it is not advised to use a single Excel file as multiple Panopticon data sources. This is because, when using the same Excel file with the data on several sheets, conflicts may occur in reading the file.**

A workaround could be to set up a Data Extract with scheduled refresh for each of the datasets in the Excel file, and then let the data tables in your workbook load the data from the Data Extracts.

Steps:

1. In the *New Data Source* page, select **Input > MS Excel (legacy)** in the *Connector* drop-down list.



2. Select the MS Excel (legacy) *File Source*.
3. Select the number of rows that will be skipped in the Excel file from the *Skip First n Rows* drop-down list.
4. If the MS Excel file is password-protected, enter the *File Password*.

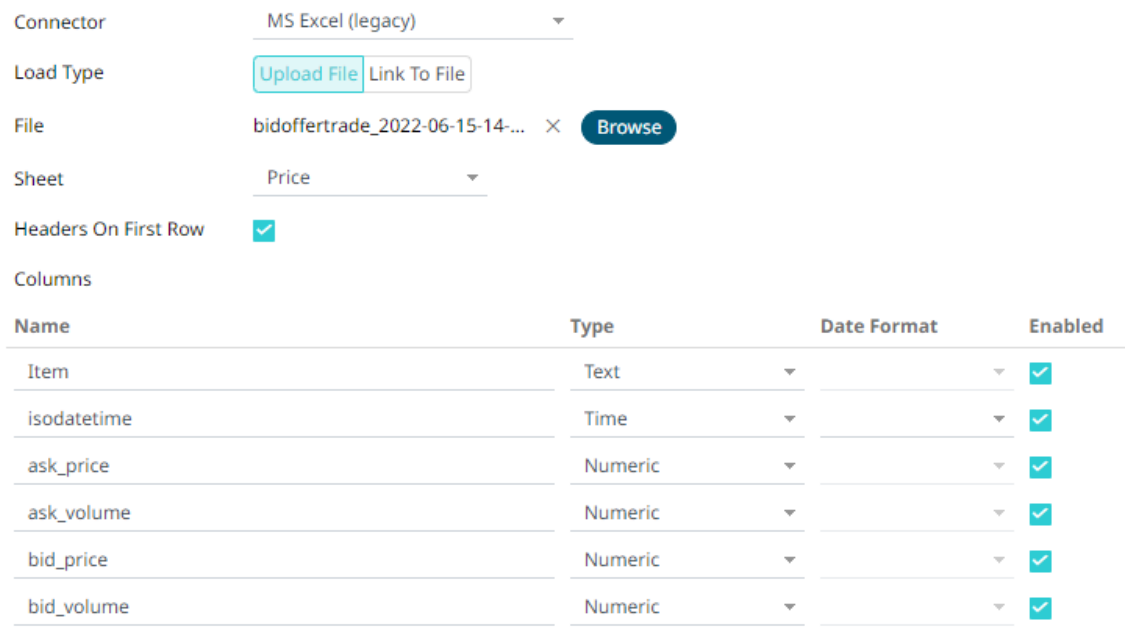
   Check the **Show Characters** box to display the entered password characters.

   NOTE **The password is case-sensitive.**

   Otherwise, proceed to step 5.

5. Click Fetch Sheets. This updates the *Sheet* drop-down list box and the **Upgrade to MS Excel (xlsx)** Upgrade to MS Excel (xlsx) button is enabled if the selected file is **XLSX**.
6. You can either:
   - select the required sheet then proceed to step 7, or
   - click Upgrade to MS Excel (xlsx) to upgrade the XLSX data source to MS Excel (xlsx).

     Once upgraded, the first available sheet will be automatically selected and columns will be populated.



See [Creating MS Excel (xlsx) Input Data Source]() for more information.

7. Click

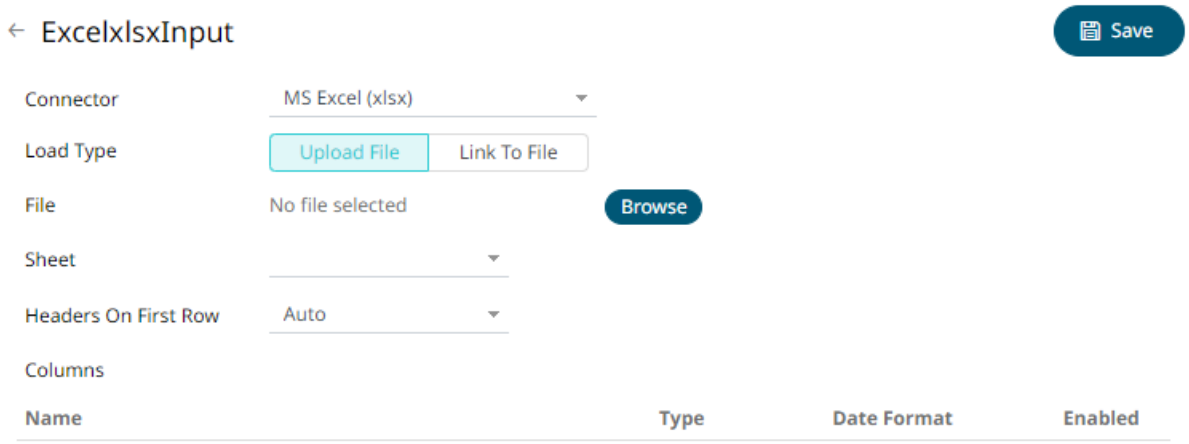
. The new data source is added in the *Data Sources* list.

Creating MS Excel (xlsx) Input Data Source

The MS Excel (xlsx) connector allows for retrieval and processing of MS Excel workbook data stored using the XLSX file format. The MS Excel (xlsx) connector processes data on a row-by-row basis, allowing for better performance and less memory consumption compared to the MS Excel (legacy) connector.

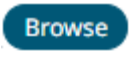
Steps:

1. On the *New Data Extract* page, select **MS Excel (xlsx)** in the *Connector* drop-down list.



2. Select the MS Excel (xlsx) file source. Do one of the following:

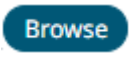
- Upload a data source snapshot by clicking **Upload File**

then **Browse** 

to browse to the file source.

  After selecting the file, it is displayed with the timestamp of the snapshot.

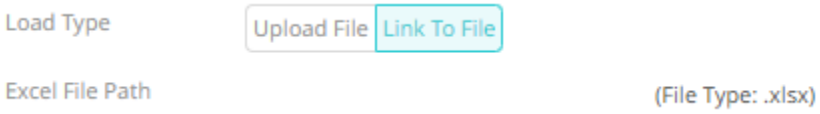
  The data source is placed in the repository and locked, synchronized, and bundled with the workbook version.

  To change the data source, click ✕ then **Browse** 

to browse to a new version of the file, which is uploaded into the repository, and also create a new version of the workbook that reads it.

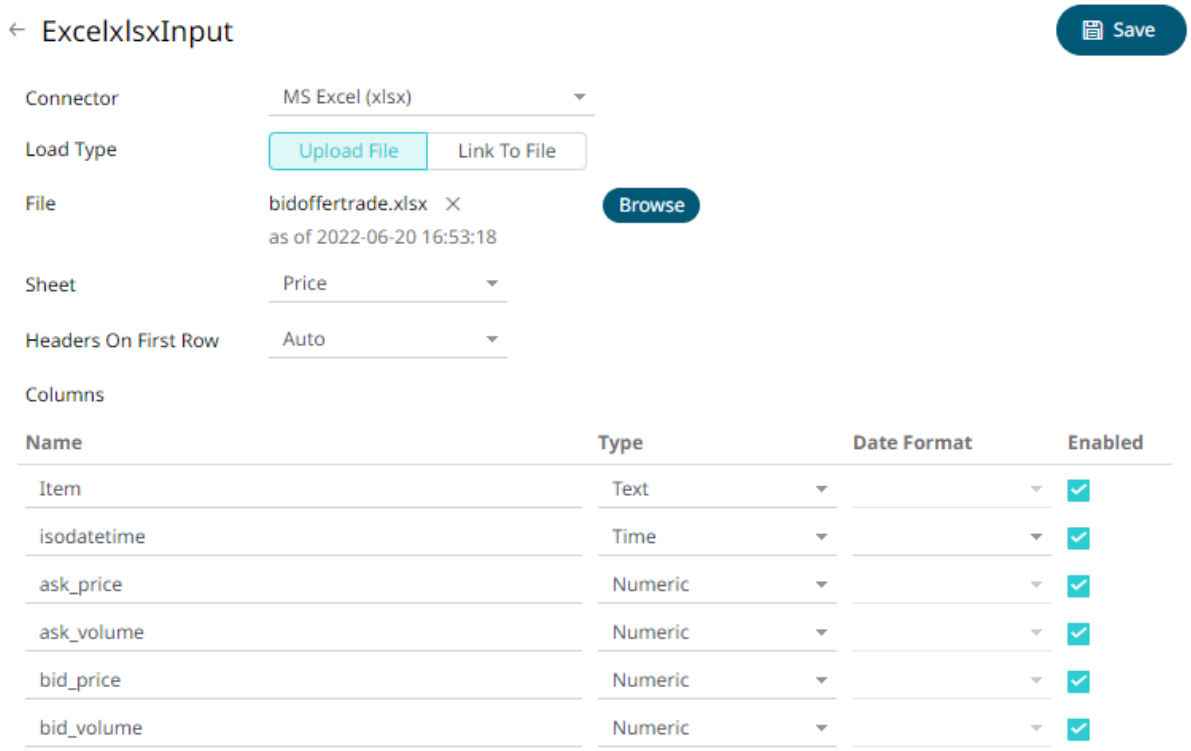
- Link to a XLSX data source file by clicking **Link to File**

and entering a *File Path*.

  Ensure that in a cluster, you need to use a shared path, or put it on every node and use a path that resolves on every node. You can update its contents whenever you want.

When a file is selected, the MS Excel (xlsx) connector will automatically select the first available sheet, set the first row as headers, and populate available columns.

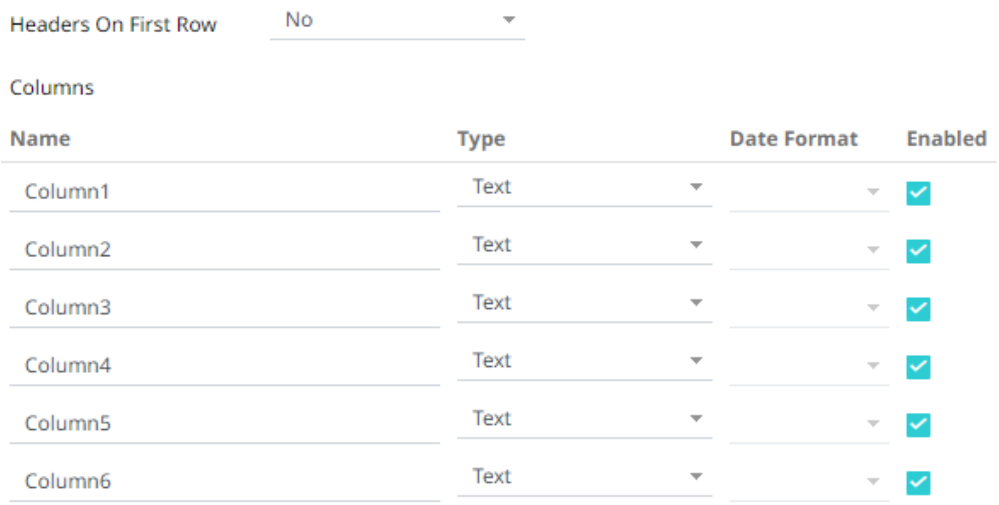


3. Adjust *Sheet* selection, if required. Selecting a new sheet will re-populate the *Columns* list.

4. Adjust the **Headers On First Row** if needed. By default the connector will pick up headers from the first row if all cells on the first row contain text data.

   You can opt to select one of the following:

   - Leave headers on first row as **Auto** if you want the connector to automatically pick up column names from sheet.
   - Select **Yes** to force picking first row as headers.
   - Select **No** to force not picking first row as headers. This will auto generate all column names.



5. Adjust column *Type* or *Date Format* to adjust data interpretation.

6. Click

. The new data source is added in the *Data Sources* panel.

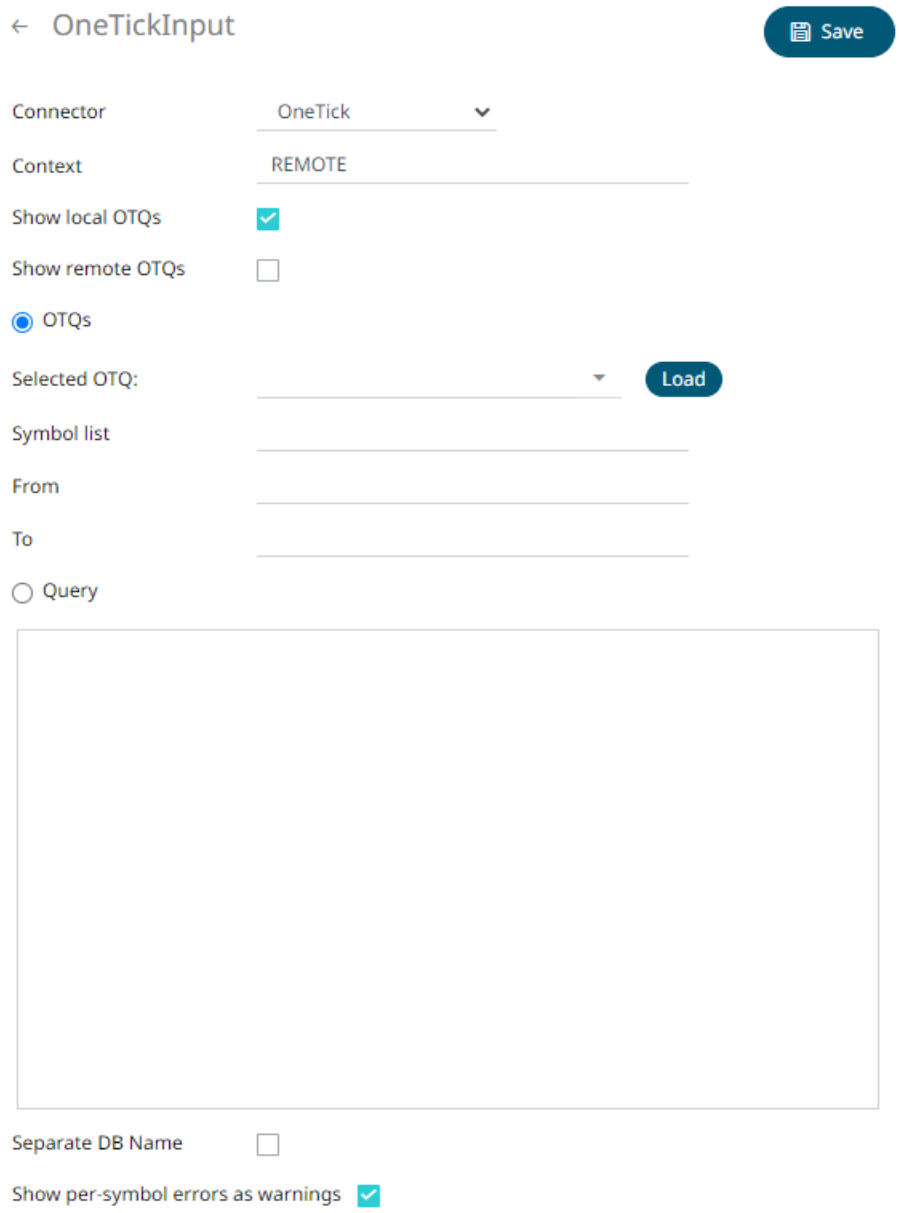
Creating OneTick Input Data Source

The OneTick connector allows connection to OneMarketData OneTick tick history databases on a polled basis. In general, it is used to retrieve conflated time series data sets. The connector supports either:

- Execution of a specified OTQ
- Execution of a specified parameterized OTQ
- Execution of a custom SQL Query

Steps:

1. In the *New Data Source* page, select **Input > OneTick** in the *Connector* drop-down list.



2. Enter the *Context* (for example, **REMOTE**).
3. You can either check:
   - **Show Local OTQs** box to display the local OTQs in the *Selected OTQ* drop-down list.
   - **Show Remote OTQs** box to display the remote OTQs in the *Selected OTQ* drop-down list.

   An OTQ can be specified for execution, or a custom SQL query can be executed, through selection of the appropriate radio button:

   - OTQs
   - Query

4. Click **Load**

to populate the *Selected OTQ* drop-down list. Select an OTQ.

The list of input parameters that the OTQ expects is displayed. In addition, the basic SQL query is generated allowing the OTQ to be executed and the input parameters specific to the selected OTQ. The following are generic to all OTQs:

- Symbol List

NOTE **This property will accept comma-separated values either hardwired or parameterized.**

- From
- To

These add additional filter criteria such as symbol, and time window onto the basic OTQ.

5. Check the **Separate DB Name** box to generate a separate field for the database name.
6. Check the **Show per symbol errors as warnings** box to proceed with warnings in the log if symbol errors are returned.

   The result is a fully generated OneTick SQL query. This can be edited as required.

7. Click

. The new data source is added in the *Data Sources* list.

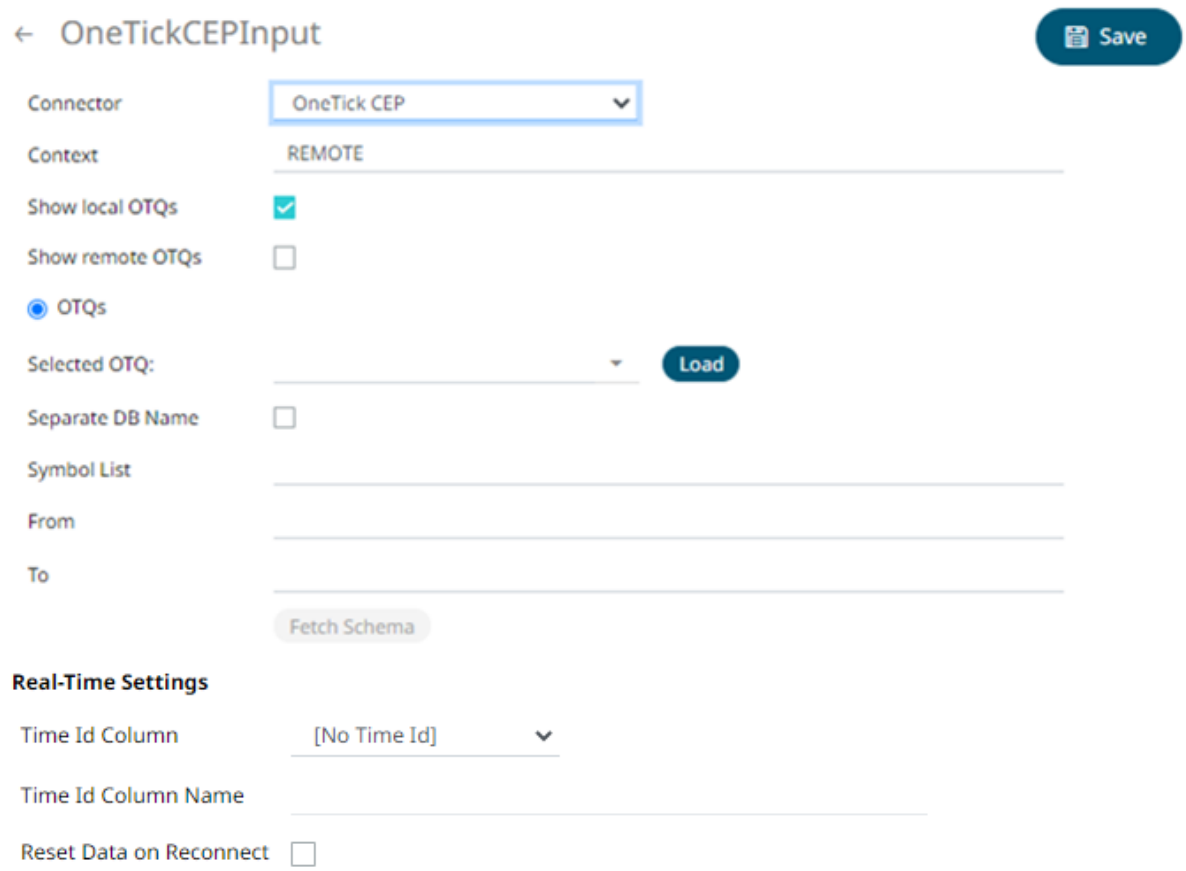
Creating OneTick CEP Input Data Source

The OneTick CEP connector allows connection to OneMarketData OneTick tick history databases on a streaming subscription basis. The connector supports either:

- Execution of a specified OTQ
- Execution of a specified parameterized OTQ
- To use the OneTick CEP connector, it requires a JAR file to be added and some configurations to be performed. Further details are provided in the Panopticon Real Time Installation and Troubleshooting Guide.

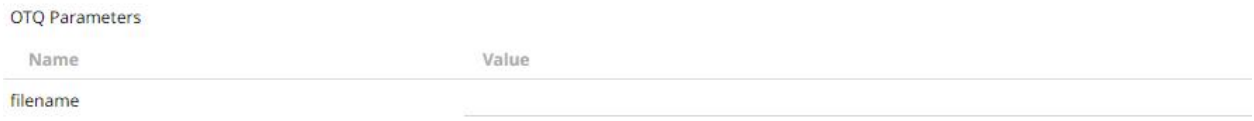
Steps:

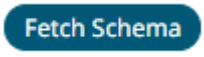
1. In the *New Data Source* page, select **Input > OneTick CEP** in the *Connector* drop-down list.



2. Enter the *Context* (for example, **REMOTE**).
3. You can either check:
   - **Show Local OTQs** box to display the local OTQs in the *Selected OTQs* drop-down list.
   - **Show Remote OTQs** box to display the remote OTQs in the *Selected OTQs* drop-down list.
4. Click **Load** to populate the *Selected OTQ* drop-down list. Select an OTQ.

   The *OTQ Parameters* section displays with the list of input parameters based on the selected OTQ.



5. Check/uncheck the *Separate DB Name* box.
6. Click  to populate the *Id Column* list box.
7. From this list box select the field which will define a unique data record to subscribe against.

   The following are generic to all OTQs

   - Symbol List
   - From

- To

  These add additional filter criteria such as symbol, and time window onto the basic OTQ.

8. Define Real-time Settings.

9. Click

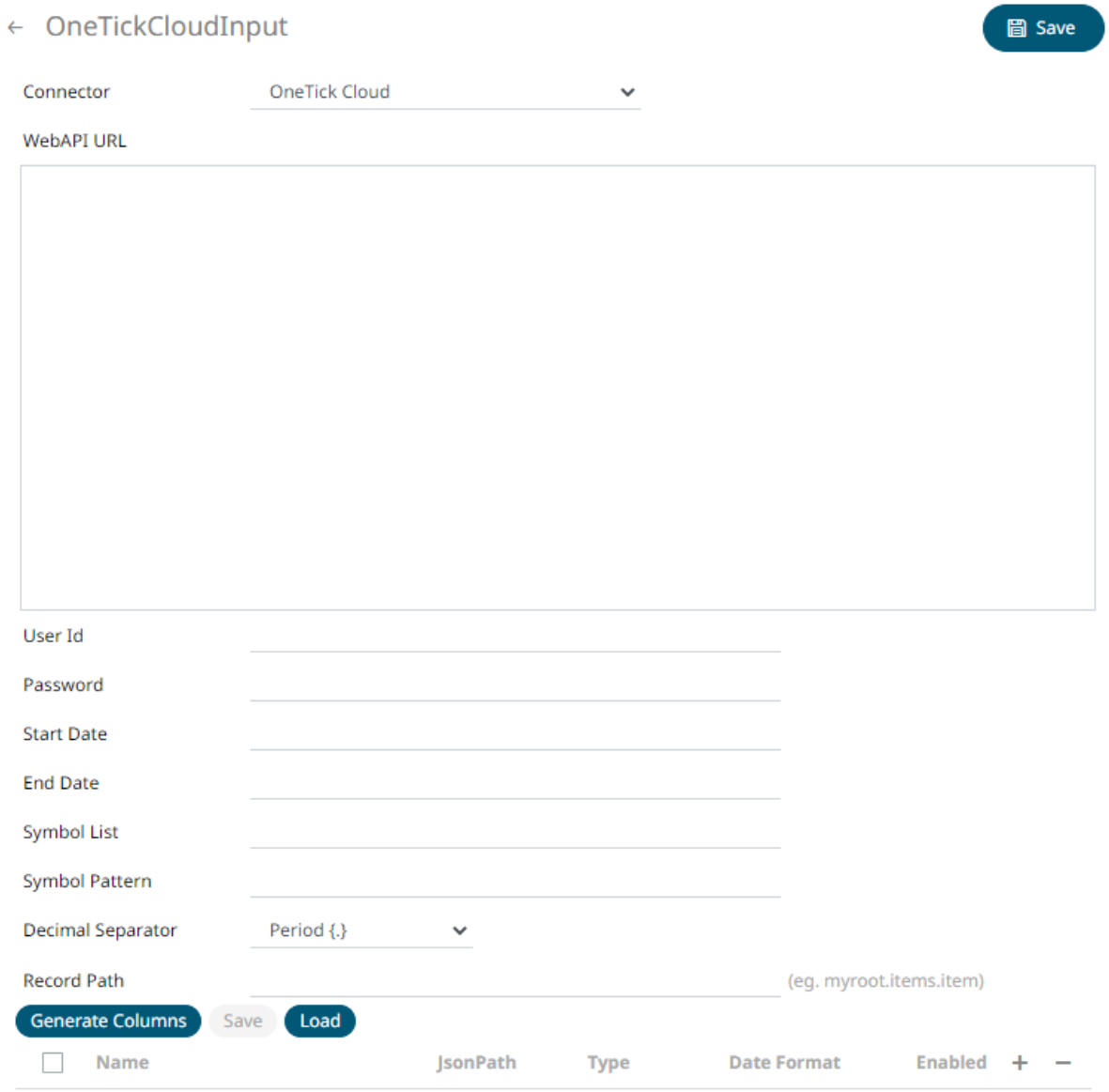
. The new data source is added in the *Data Sources* list.

Creating OneTick Cloud Input Data Source

The OneTick Cloud connector allows access to historic market data with no software dependencies by using the OneTick Cloud and their web API.

Steps:

1. In the *New Data Source* page, select **Input > OneTick Cloud** in the *Connector* drop-down list.



2. Enter the OneTick Cloud WebAPI URL into the *WebAPI URL* box with the following form:

```
http://<host>/omdwebapi/rest/?params={"context":"DEFAULT","query_type":"otq",
"otq":"1/12/otq/71b50459-8431-48dc-829f
"s":"20150305130802",
"e":"20150305140805",
"timezone":"America/New_York", "response":"csv",
"compression":"gzip"}
```

   Where:

   - s, e, timezone – the start and end time of the query YYYYMMDDhhmmss form. The timezone used to interpret this value is taken from the timezone parameter.
   - response – the supported response format is csv.
   - compression – if available, this option enables gzip compression of the results stream. Large data should always be pulled with compression on.

3. Enter the *User Id* (email) and *Password* to execute the query and retrieve the data. Note that the *User Id* is case sensitive.
4. Enter the time window *Start Date* and *End Date*.
5. Enter the *Symbol List.* This value filters the query output with matching symbols.

   To make it work, ensure to include `Symbol` in the *Query URL*. Consequently, the data will be filtered out for the input (Symbols) provided in the *Symbol List* field.

6. Enter the *Symbol Pattern*. This value filters the query output with the data for all the symbols with matching pattern.

   To make it work, ensure to include `Symbol_Pattern` in the *Query URL*. Consequently, the data will be filtered (for all the Symbols) with matching pattern provided in the *Symbol Pattern* field.

7. Select either the period (.) or comma (,) as the *Decimal Separator*.

   NOTE **Prepend 'default:' for the elements falling under default namespace.>**

8. Click Generate Columns to the fetch the schema based on the connection details. Consequently, the list of columns with the data type found from inspecting the first 'n' rows of the input data source is populated and the Save button is enabled.
9. You can also opt to load or save a copy of the column definition.
10. You can opt to click + . A new column entry displays. Enter or select the following properties:

Property	Description
Name	The column name of the source schema.
Column Index	The column index controls the position of a column. Must be **>= 0**.
Type	The data type of the column. Can be a **Text**, **Numeric**, or **Time**
Date Format	The format when the data type is **Time**.
Filter	Defined parameters that can be used as filter.

Enabled	Determines whether the message should be processed.

To delete a column, check its ☐ or all the column entries, check the topmost ☐, then click ⊟.

11. Click Save. The new data source is added in the *Data Sources* list.

Creating Python Input Data Source

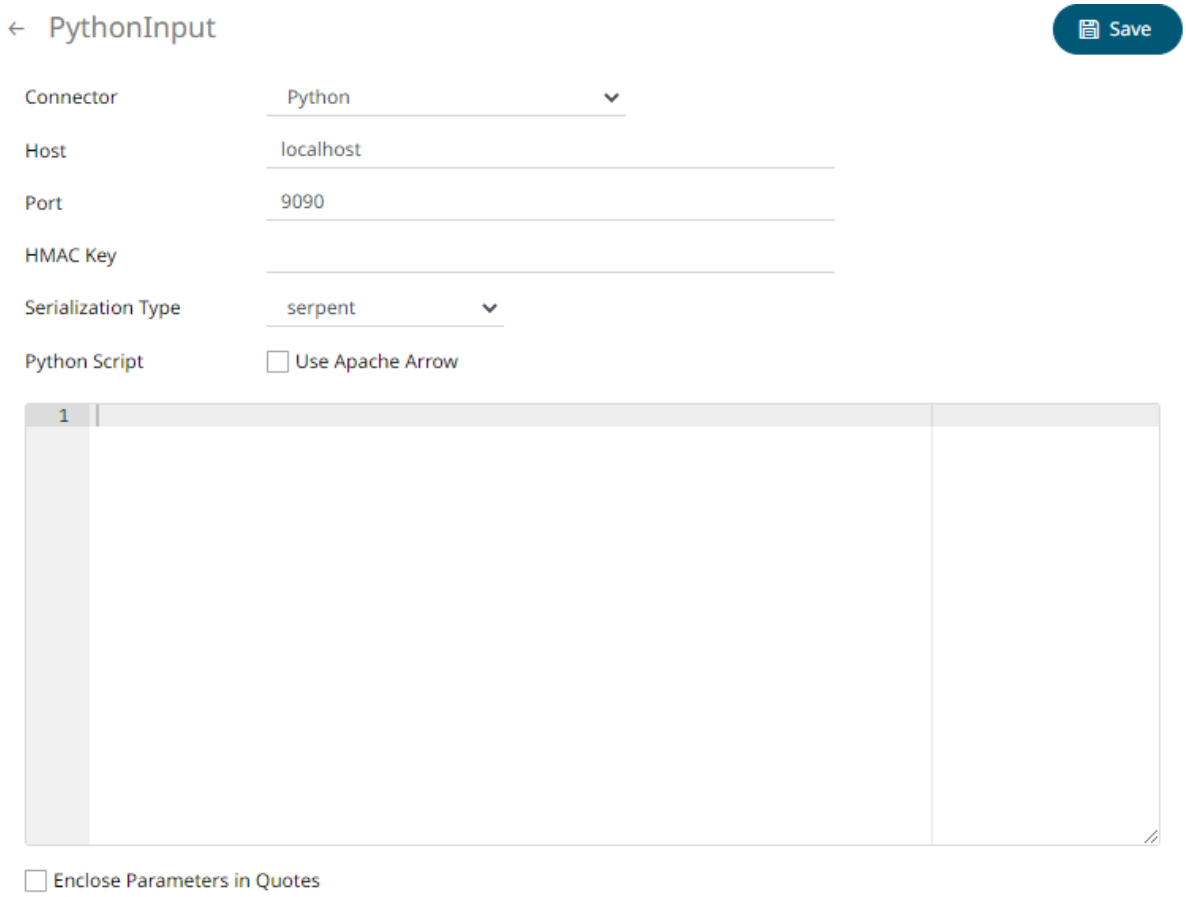
The Python connector allows the retrieval of output data from a Python Pyro (Python Remote Objects) process.

For Python connectivity, Python must be first installed, together with the latest version of Pyro4. In addition, Pyro must be initiated manually or through using the batch file **start_Python_connectivity.bat**.

If the scripts utilize additional modules such as Numpy & Scipy in the shipped example, these also need to be installed into the existing Python installation.

Steps:

1. In the *New Data Source* page, select **Input > Python** in the *Connector* drop-down list.



2. Enter the following fields:

Field	Description
Host	Python Pyro instance host address.
Port	Python Pyro host port. Default is **9090**.
HMAC Key	Set to **password**.

3. Select the *Serialization Type*: **Serpent** or **Pickle.**
   - Serpent – simple serialization library based on `ast.literal_eval`
   - Pickle – faster serialization but less secure

   Modify the `configuration.py` file located in `..\Anaconda3\Lib\site-packages\Pyro4` to specify the serialization to be used.

For example, if **Pickle** is selected, `self.SERIALIZER` value should be changed to **pickle** and `self.SERIALIZERS_ACCEPTED` value should be changed to include **pickle**:

```
def reset(self, useenvironment=True):
        """
        Set default config items.
        If useenvironment is False, won't read environment variables
settings (useful if you can't trust your env).
        """
        self.HOST = "localhost"  # don't expose us to the outside world
by default
        self.NS_HOST = self.HOST
        self.NS_PORT = 9090  # tcp
        self.NS_BCPORT = 9091  # udp
        self.NS_BCHOST = None
        self.NATHOST = None
        self.NATPORT = 0
        self.COMPRESSION = False
        self.SERVERTYPE = "thread"
        self.COMMTIMEOUT = 0.0
        self.POLLTIMEOUT = 2.0  # seconds
        self.SOCK_REUSE = True  # so_reuseaddr on server sockets?
        self.SOCK_NODELAY = False  # tcp_nodelay on socket?
        self.THREADING2 = False  # use threading2 if available?
        self.ONEWAY_THREADED = True  # oneway calls run in their own
thread
        self.DETAILED_TRACEBACK = False
        self.THREADPOOL_SIZE = 16
        self.AUTOPROXY = True
        self.MAX_MESSAGE_SIZE = 0  # 0 = unlimited
        self.BROADCAST_ADDRS = "<broadcast>, 0.0.0.0"  # comma
separated list of broadcast addresses
        self.FLAME_ENABLED = False
        self.PREFER_IP_VERSION = 4  # 4, 6 or 0 (let OS choose
according to RFC 3484)
        self.SERIALIZER = "pickle"
        self.SERIALIZERS_ACCEPTED = "pickle,marshal,json"   # these are
the 'safe' serializers
        self.LOGWIRE = False  # log wire-level messages
        self.PICKLE_PROTOCOL_VERSION = pickle.HIGHEST_PROTOCOL
        self.METADATA = True  # get metadata from server on proxy
connect
        self.REQUIRE_EXPOSE = False  # require @expose to make members
remotely accessible (if False, everything is accessible)
        self.USE_MSG_WAITALL = hasattr(socket, "MSG_WAITALL") and
platform.system() != "Windows"      # not reliable on windows even
though it is defined
        self.JSON_MODULE = "json"
        self.MAX_RETRIES = 0
```

NOTE **The *Host*, *Port*, *HMAC Key*, and *Serialization Type* fields will be hidden if their corresponding properties are set in the `Streams.properties` file.**

Field	Corresponding Property in Streams.properies
Host	`connector.python.host`
Port	`connector.python.port`
HMAC Key	`connector.python.password`
Serialization Type	`connector.python.serializertype`

4. Enter the required *Python script* to execute on the active Pyro instance.
5. Check the **Use Apache Arrow** box to enable fast serialization of data frames.
6. Select whether the parameters should be automatically enclosed in quotes by checking the **Enclose Parameters in Quotes** box.
7. Click

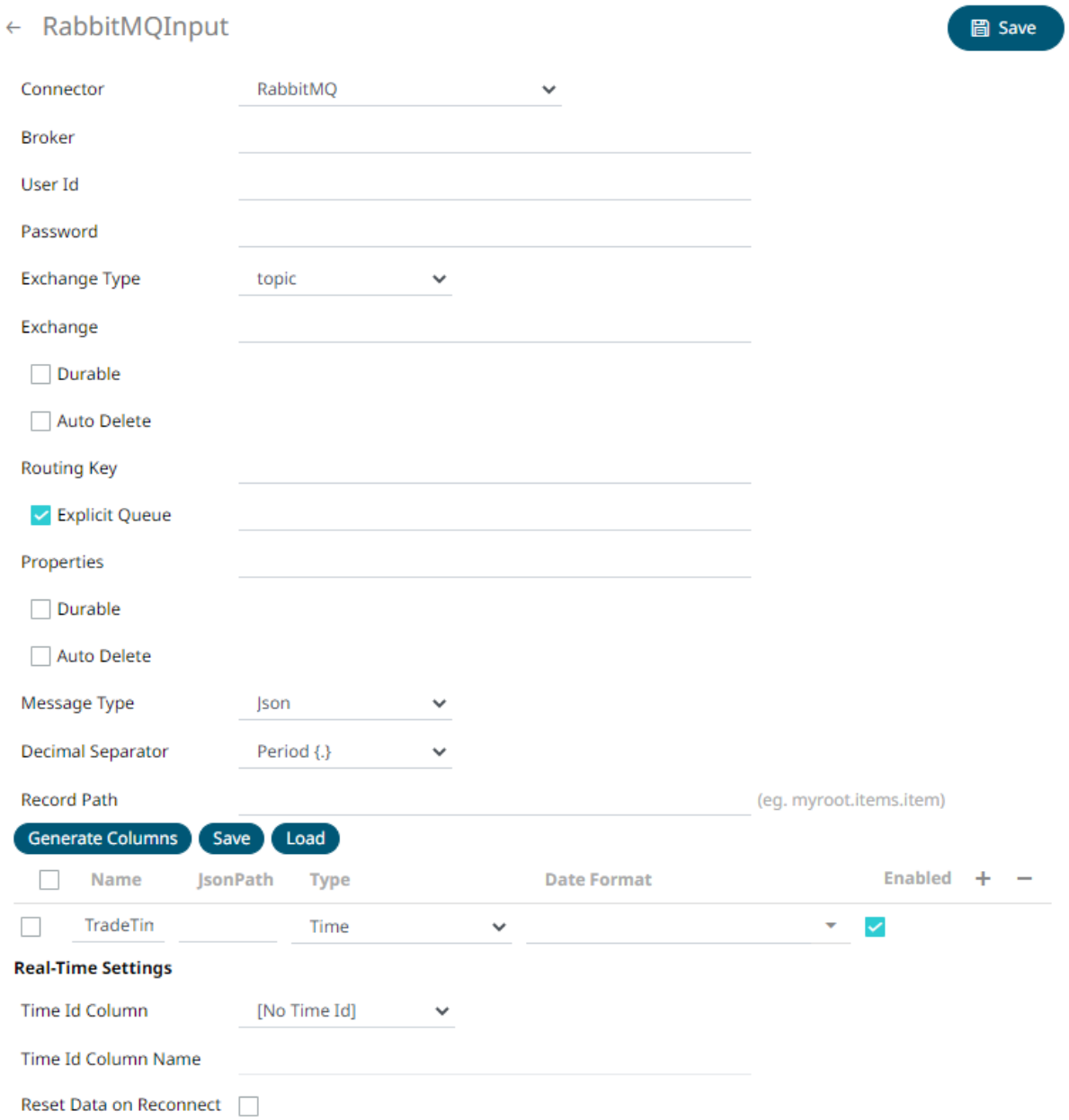
. The new data source is added in the *Data Sources* list.

Creating RabbitMQ Input Data Source

The RabbitMQ connector allows connection to RabbitMQ's message bus on a real-time streaming basis. Specifically, the connector allows Panopticon Streams to subscribe to XML, JSON, Text or FIX based messages that are published on particular topics.

Steps:

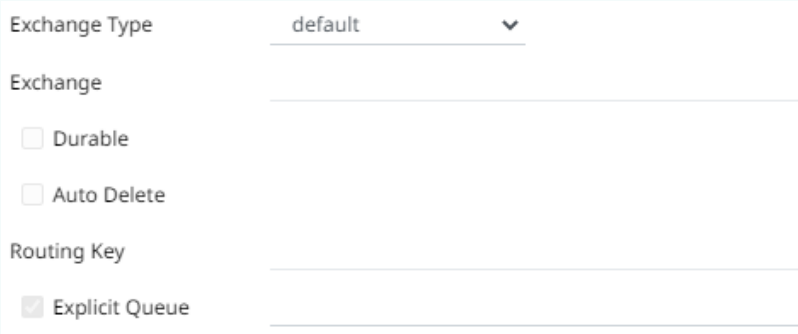
1. In the *New Data Source* page, select **Input > RabbitMQ** in the *Connector* drop-down list.



2. Enter the connection details including:

Property	Description
Broker	The location of the message broker.
User Id	The user Id that will be used to connect to RabbitMQ.
Password	The password that will be used to connect to RabbitMQ.

3. Select any of the following *Exchange Types*:

Exchange Type	Description
Default	A direct exchange with no name that is pre-declared by the broker. Selecting this exchange type disables the *Exchange* section (*Exchange* and *Routing Key* properties). 
Fanout	Broadcasts all of the messages it receives to all of the queues it knows and the routing key is ignored (the *Routing Key* field is disabled).
Direct	Delivers messages to queues based on a message routing key. It is ideal for the unicast routing of messages, although it can be used for multicast routing as well.
Topic	A message sent with a particular routing key will be delivered to all of the queues that are bound with a matching binding key.
Headers	Exchanges routed based on arguments containing headers and optional values.

4. Depending on the selected *Exchange Type*, select or define the following:

Exchange Type Property	Description
Exchange	Name of the exchange.
Durable	Enable so the exchange can survive a broker restart.
Auto Delete	Enable so the exchange is deleted when the last queue is unbound from it.
Routing Key	The routing key used to deliver messages to queues.
Headers	This field is only available when the message type is **Header**. Binding a queue to a Headers exchange is possible using more than one header for matching. Setting *x-match* to **any**, means just one matching value is sufficient. Setting it to **all** means that all values must match. Default is **x-match=all**.

5. Check the *Explicit Queue* box and enter the custom queue name. Then enter or enable the following properties:

Queue Property	Description
Properties	The custom queue property.
Durable	Enable so the queue can survive a broker restart.

Auto Delete	Enable so the queue that had the least consumer will be deleted when that connection closes.

6. Select the *Message Type*.
7. Select either the period (.) or comma (,) as the *Decimal Separator*.

NOTE **Prepend 'default:' for the elements falling under default namespace.**

8. Click Generate Columns to the fetch the schema based on the connection details. Consequently, the list of columns with the data type found from inspecting the first 'n' rows of the input data source is populated and the Save button is enabled.
9. You can also opt to load or save a copy of the column definition.
10. You can opt to click + to add columns to the RabbitMQ connection that represent sections of the message. Then enter or select:

Property	Description
Name	The column name of the source schema.
Fix Tag/Json Path/Text Column Index/Xpath	The Fix Tag/Json Path/Text Column Index/Xpath of the source schema.
Type	The data type of the column. Can be a **Text**, **Numeric**, or **Time**
Date Format	The format when the data type is **Time**. **NOTE:** To parse and format times with higher than millisecond precision, the format string needs to end with a period followed by sequence of upper case S. There can be no additional characters following them. For example: yyyy-MM-dd HH:mm:ss.SSSSSS
Filter	Defined parameters that can be used as filter. Only available for JSON, Text, and XML message types.
Enabled	Determines whether the message field should be processed.

To delete a column, check its ☐ or all the column entries, check the topmost ☐, then click −.

11. Define the Real-time Settings.
12. Click Save. The new data source is added in the *Data Sources* list.

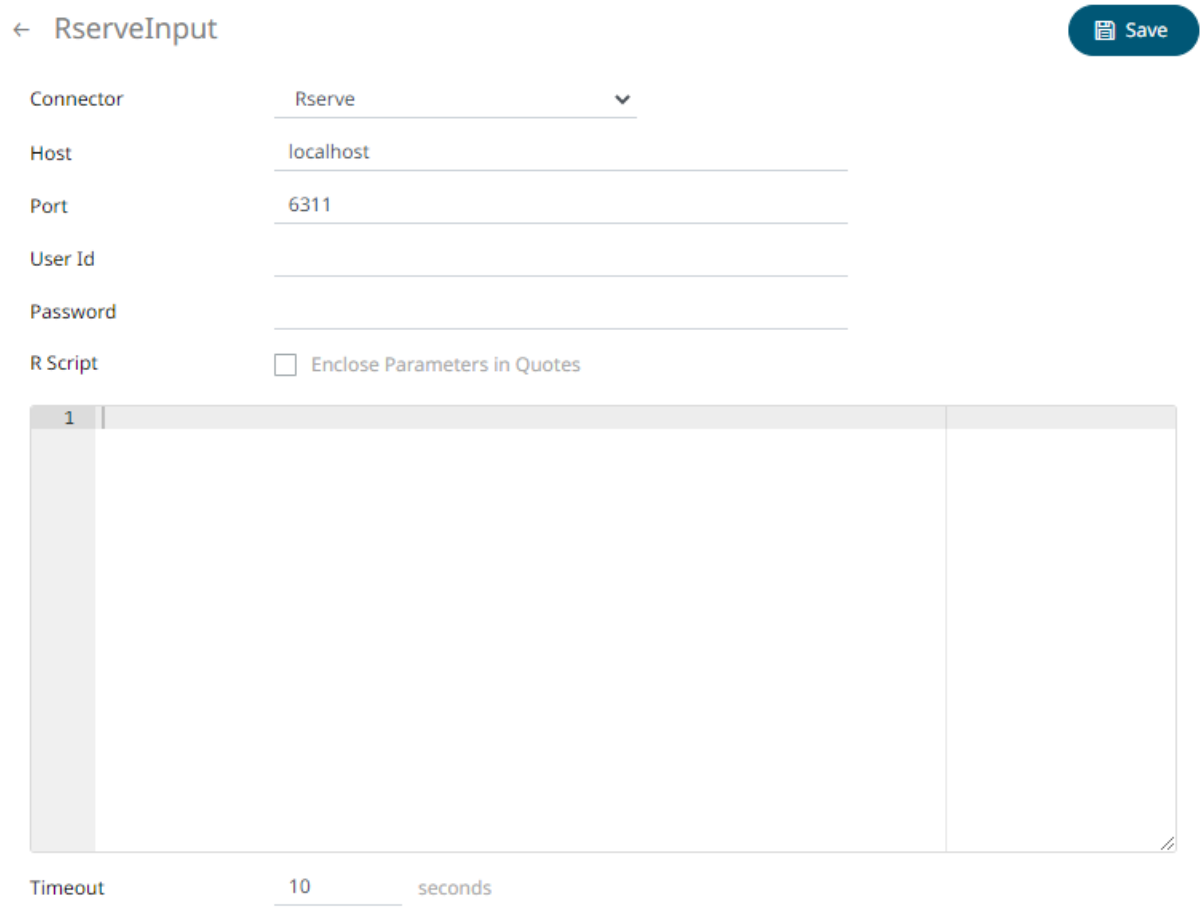
Creating Rserve Input Data Source

The Rserve connector allows the retrieval of an output data frame from a running Rserve process.

For R connectivity, R must be first installed, together with the Rserve library. In addition, R must be open, and the Rserve library must be loaded and initialized.

Steps:

1. In the *New Data Source* page, select **Input > Rserve** in the *Connector* drop-down list.



2. Enter the following properties:

Property	Description
Host	Rserve host address.
Port	Rserve host port. Default is **6311**.
User Id	The user Id that will be used to connect to the Rserve service.
Password	The password that will be used to connect to the Rserve service.

3. Enter the required *R script* to execute on the active Rserve instance.
4. The Timeout is set to **10** seconds by default to ensure that slow running R scripts do not impact other areas of the product. You can opt to enter a new value.

5. Select whether the parameters should be automatically enclosed in quotes by checking the *Enclose parameters in quotes* box.
6. Click

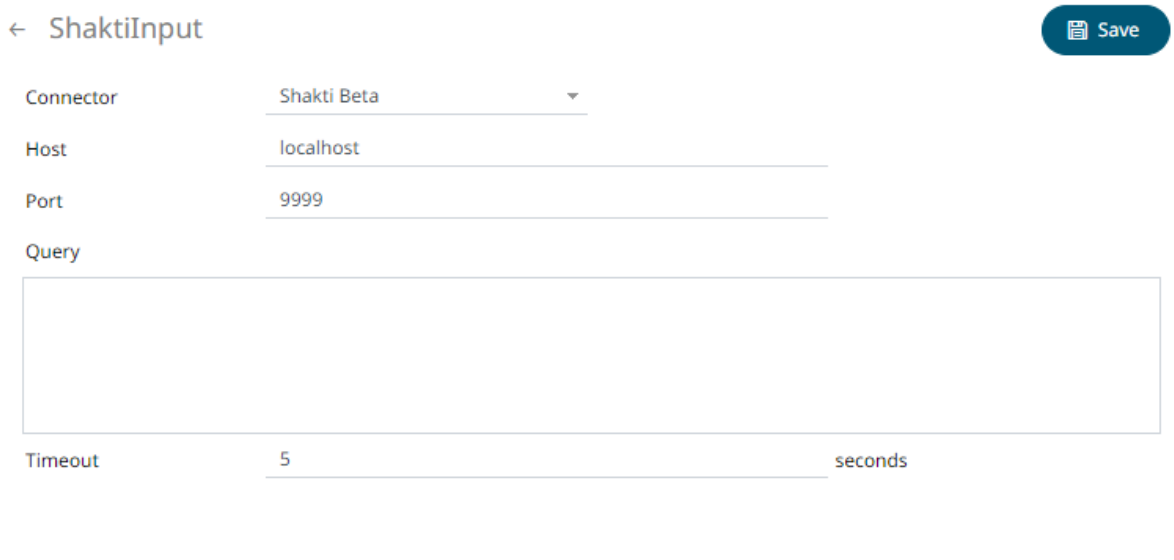
. The new data source is added in the *Data Sources* list.

Creating Shakti Beta Input Data Source

The Shakti connector allows connecting to Shakti databases on a polled basis.

Steps:

1. In the *New Data Source* page, select **Input > Shakti Beta** in the *Connector* drop-down list.



2. Enter the following properties:

Property	Description
Host	Shakti host address.
Port	Shakti host port. Default is **9999**.

3. Enter a free text *Query*.
4. Adjust the *Timeout,* if needed. Default is **5** (in seconds).
5. Click

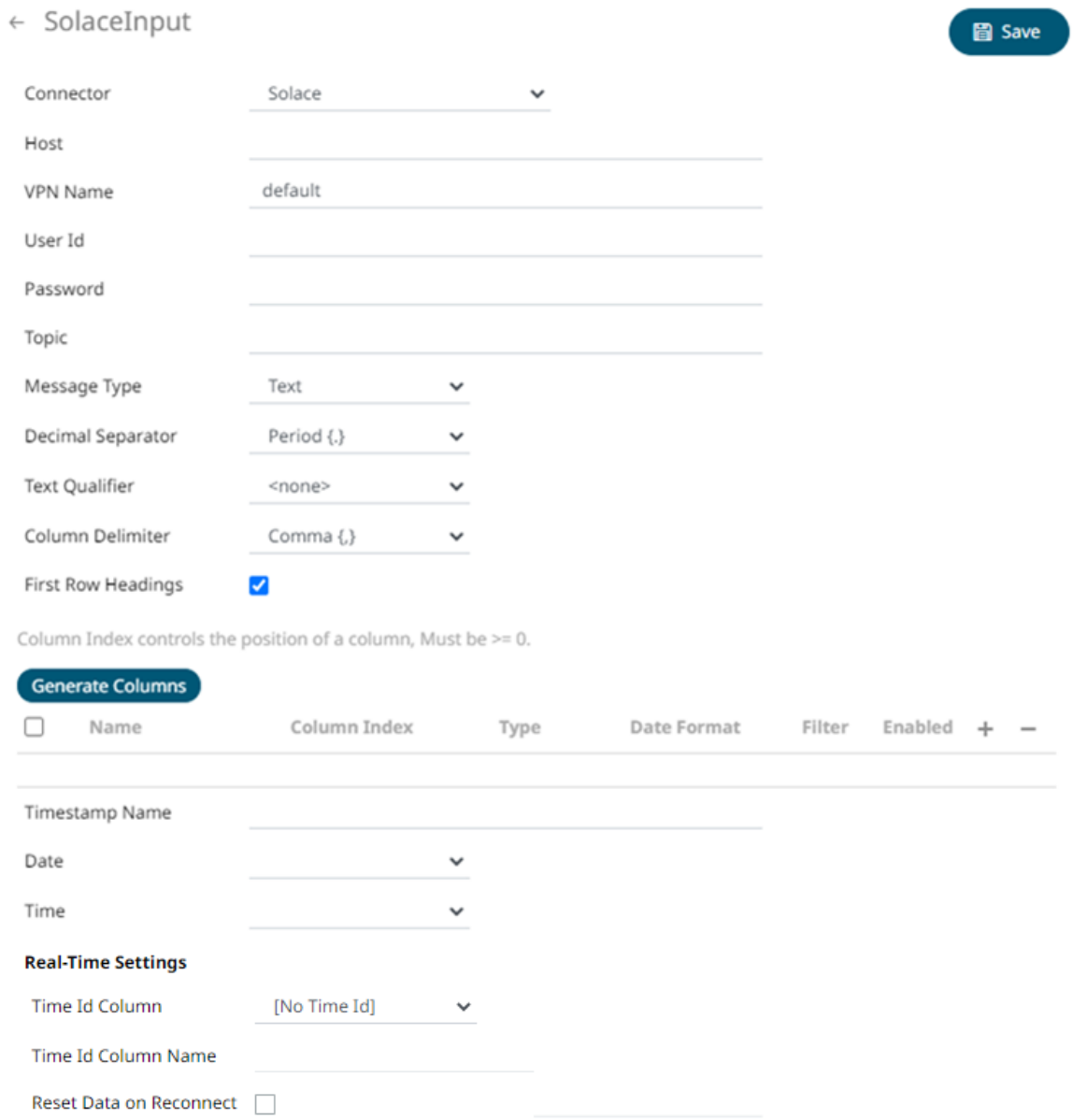
. The new data source is added in the *Data Sources* list.

Creating Solace Input Data Source

The Solace connector allows connection to Solace's message bus on a real time streaming basis. Specifically, the connector allows Panopticon Streams to subscribe to messages that are published in particular topics in Solace and consequently, perform operational analytics.

Steps:

1. In the *New Data Source* page, select **Input > Solace** in the *Connector* drop-down list.



2. Enter the connection details including:

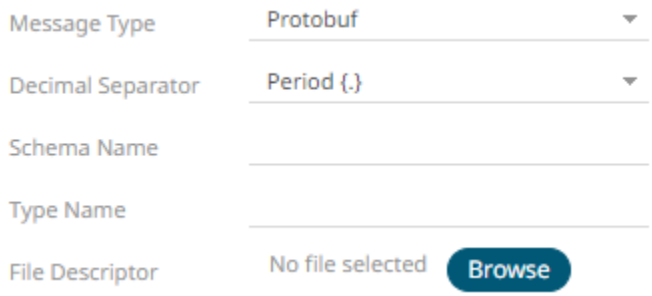
Property	Description
Host	Solace host address.
VPN Name	Message VPN name. Default is **default**.
User Id	The user Id that will be used to connect to Solace.
Password	The password that will be used to connect to Solace.

3. Enter the *Topic* or the queue physical name.
4. Select the *Message Type*. This will specify the format of the data within the message.

Aside from the **Fix**, **Json**, **Text**, and **XML** message types, **Protobuf** is also supported in Solace.

If **Protobuf** is selected, confirm the **Decimal Separator,** and enter the *Schema Name* and *Type Name*.

Then click Browse to select the **File Descriptor** (`.desc` file) in the *Open* dialog.



Property	Description
Schema Name	The Protobuf schema.
Type Name	The message of Protobuf type that will be sent to Kafka.
File Descriptor	The `FileDescriptorSet` which: • is an output of the protocol compiler. • represents a set of `.proto` files, using the `--descriptor_set_out` option.

5. Select either the period (.) or comma (,) as the *Decimal Separator*.

NOTE **Prepend 'default:' for the elements falling under default namespace.**

6. Click Generate Columns to the fetch the schema based on the connection details. This populates the list of columns with the data type found from inspecting the first 'n' rows of the input data source.

7. You can opt to click + to add columns to the Solace connection that represent sections of the message. Then enter or select:

Property	Description
Name	The column name of the source schema.
Type/JsonPath/Column Index/XPath	The SDTMap Type/JsonPath/Text Column Index/XPath of the source schema.
Type	The data type of the column. Can be a **Text**, **Numeric**, or **Time**
Date Format	The format when the data type is **Time**. **NOTE:** To parse and format times with higher than millisecond precision, the format string needs to end with a period followed by sequence of upper case S. There can be no additional characters following them.

	For example: yyyy-MM-dd HH:mm:ss.SSSSSS
Filter	Defined parameters that can be used as filter. Only available for JSON, Text, and XML message types.
Enabled	Determines whether the message field should be processed.

To delete a column, check its ☐ or all the column entries, check the topmost ☐, then click ▬.

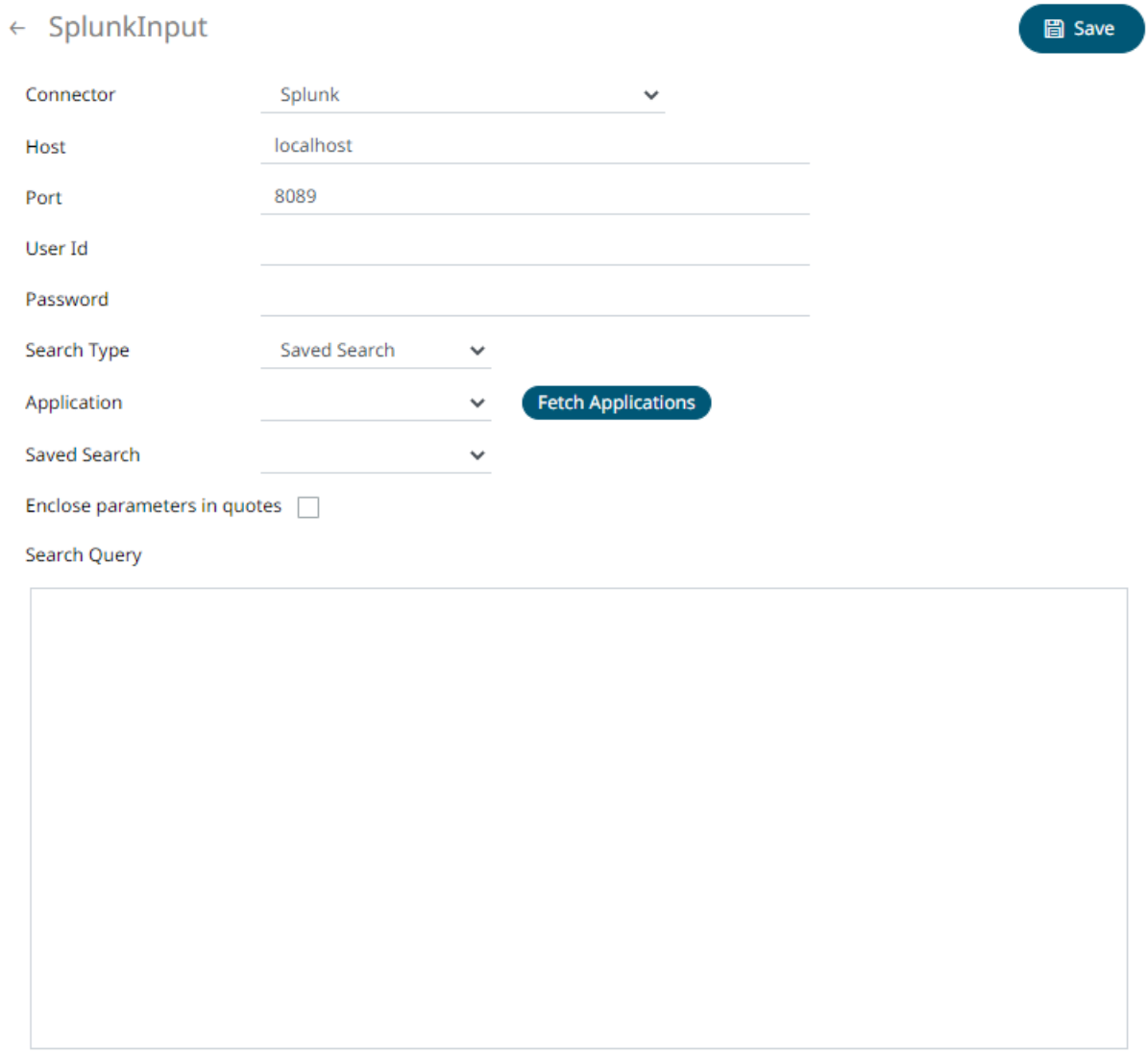
8. To create a new Timestamp field, enter a new *Timestamp Name* and then select the valid Date/Time from either a single *Date* or *Time* field, or a compound column created from *Date* and *Time* fields.
9. Define the Real-time Settings.
10. Click **Save**. The new data source is added in the *Data Sources* list.

Creating Splunk Input Data Source

The Splunk connector allows the retrieval of data from a Splunk instance.

Steps:

6. In the *New Data Source* page, select **Input > Splunk** in the *Connector* drop-down list.



7. Enter the connection details including:

Property	Description
Host	Splunk host address.
Port	Splunk host port. Default is **8089**.
User Id	The user Id that will be used to connect to the Splunk service.
Password	The password that will be used to connect to the Splunk service.

8. Select the *Search Type*:
   - Manual

     Proceed to step 6 to define a new search query.
   - Saved Search

     Allows you to select in the *Saved Search* drop-down list.

9. Click Fetch Applications to populate the *Application* drop-down list and select one.
10. Select whether the parameters should be automatically enclosed in quotes by checking the **Enclose parameters in quotes** box.
11. Enter a *Search Query*.
12. Click Save **.** The new data source is added in the *Data Sources* list.

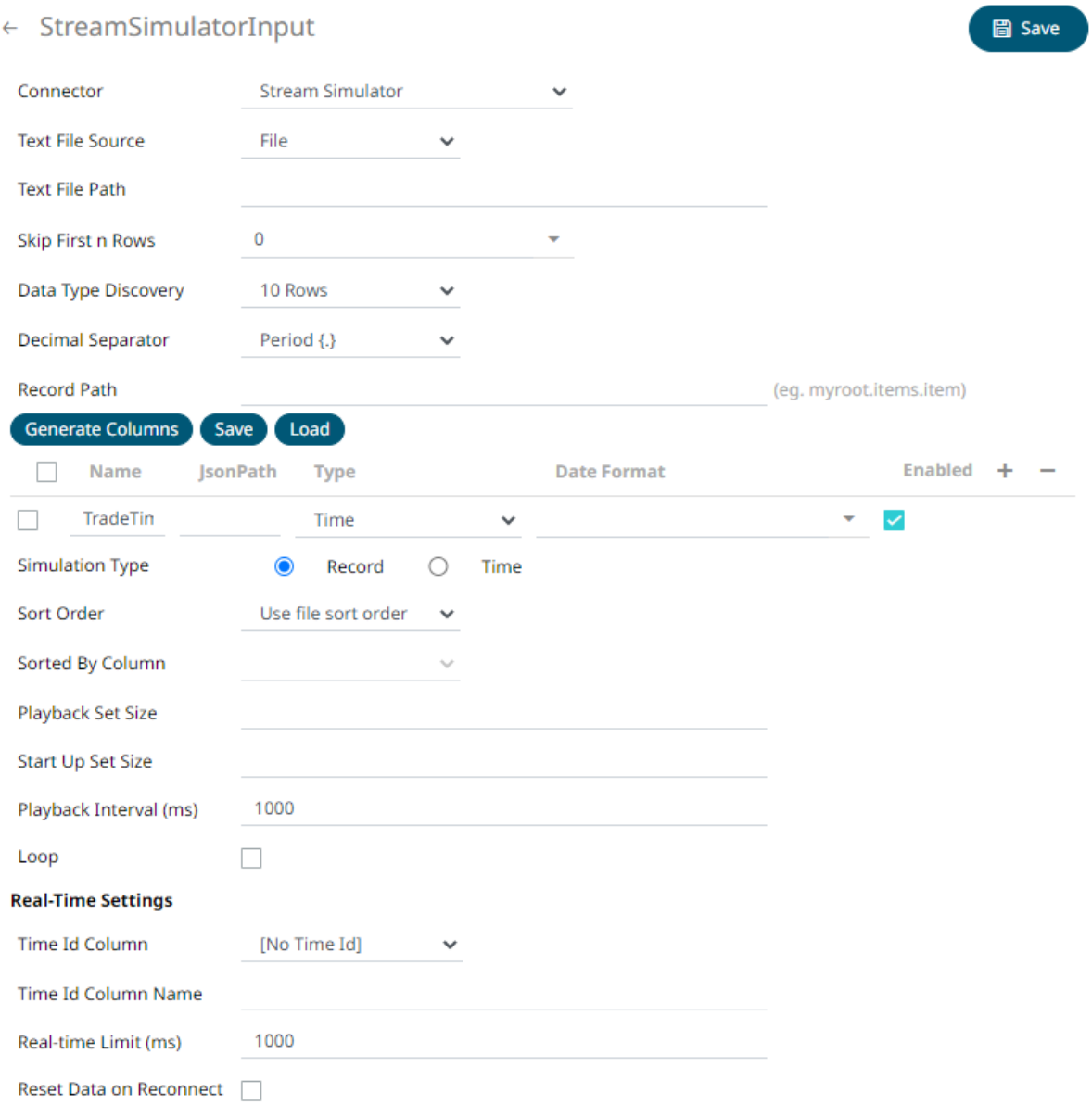
Creating Stream Simulator Input Data Source

The Stream Simulator connector is very similar to the Text connector with the addition of the time windowing of message queue connectors.

Creating the Stream Simulator input data source includes setting for how fast and how many messages are pushed through in each batch.

Steps:

1. In the *New Data Source* page, select **Input > Stream Simulator** in the *Connector* drop-down list.



2. Select the Text *File Source*.

   The standard settings controlling how the text file is parsed, is listed.

These include:

Property	Description
Skip First N Rows	Specifies the number of rows that will be skipped.
Data Type Discovery	Specifies how many rows from the text file should be used when automatically determining the data types of the resulting columns.
Decimal Separator	Select either the period (.) or comma (,) as the decimal separator.
Text Qualifier	Specifies if fields are enclosed by text qualifiers, and if present to ignore any column delimiters within these text qualifiers.
Column Delimiter	Specifies the column delimiter to be used when parsing the text file.
First Row Headings	Determines if the first row should specify the retrieved column headings, and not be used in data discovery.

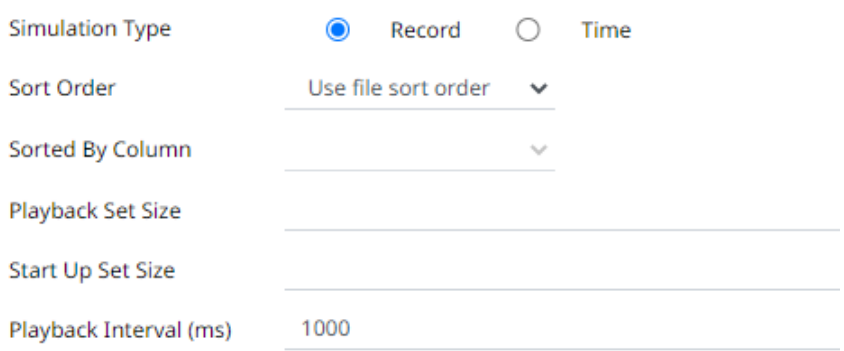
3. Click Generate Columns to the fetch the schema based on the connection details. Consequently, the list of columns with the data type found from inspecting the first 'n' rows of the input data source is populated and the **Save** button is enabled.
4. You can also opt to load or save a copy of the column definition.
5. You can opt to click + . A new column entry displays. Enter or select the following properties:

Property	Description
Name	The column name of the source schema.
Column Index	The column index controls the position of a column. Must be >= **0**.
Type	The data type of the column. Can be a **Text**, **Numeric**, or **Time**
Date Format	The format when the data type is Time.
Enabled	Determines whether the message should be processed.

To delete a column, check its ☐ or all the column entries, check the topmost ☐, then click − .

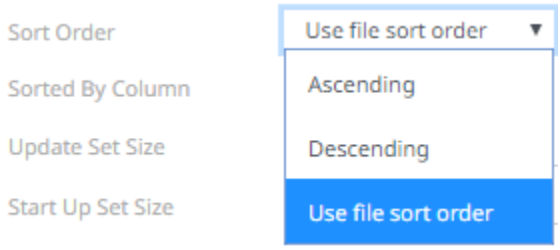
6. Select the *Simulation Type*:
   - Record

     Sends the number of records for each interval of time. By default, records are sent in the same order of the source.



This simulation type allows the specification of the following:

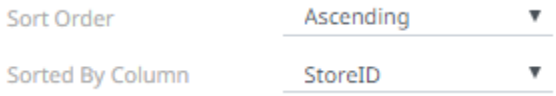
- Sort Order



  When you select the **Use file sort order**, it will use the default sorting order of the file.

  When you either select **Ascending** or **Descending** as the Sort Order, this enables the *Sorted by Column* drop down list.

  Select the column that will be used for the sorting.



- Playback Set Size

  The number of records set to be updated during simulate/playback.

- Start Up Set Size

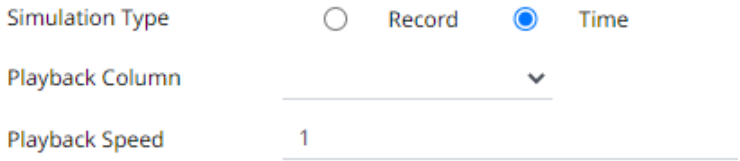
  The number of records set to be published initially (on start-up).

- Playback Interval (ms)

  The update interval period for the record-based playback. Default is **1000 (ms)**.

- Time

Simulates records as they occur in real-time.



This simulation type allows the specification of the following:

- Playback Column

  The playback column which is a Date/Time type.

- Playback Speed

  A multiplier which to either speed up or slow down the playback. Default is **1**.

  - If 0 < value < 1 slow down
  - If value = 1 records will be published as they occur
  - if value > 1 speed up

NOTE **For time-based simulation, if the Date/Time column have improper dates, it will fail and stop.**

7. Check the **Loop** box to enable looping through the file.
8. Define the Real-time Settings.
9. Modify the *Real-time Limit* to vary the data throttling. This defaults to **1000** milliseconds.
10. Click Save. The new data source is added in the *Data Sources* list.

Creating StreamBase Input Data Source

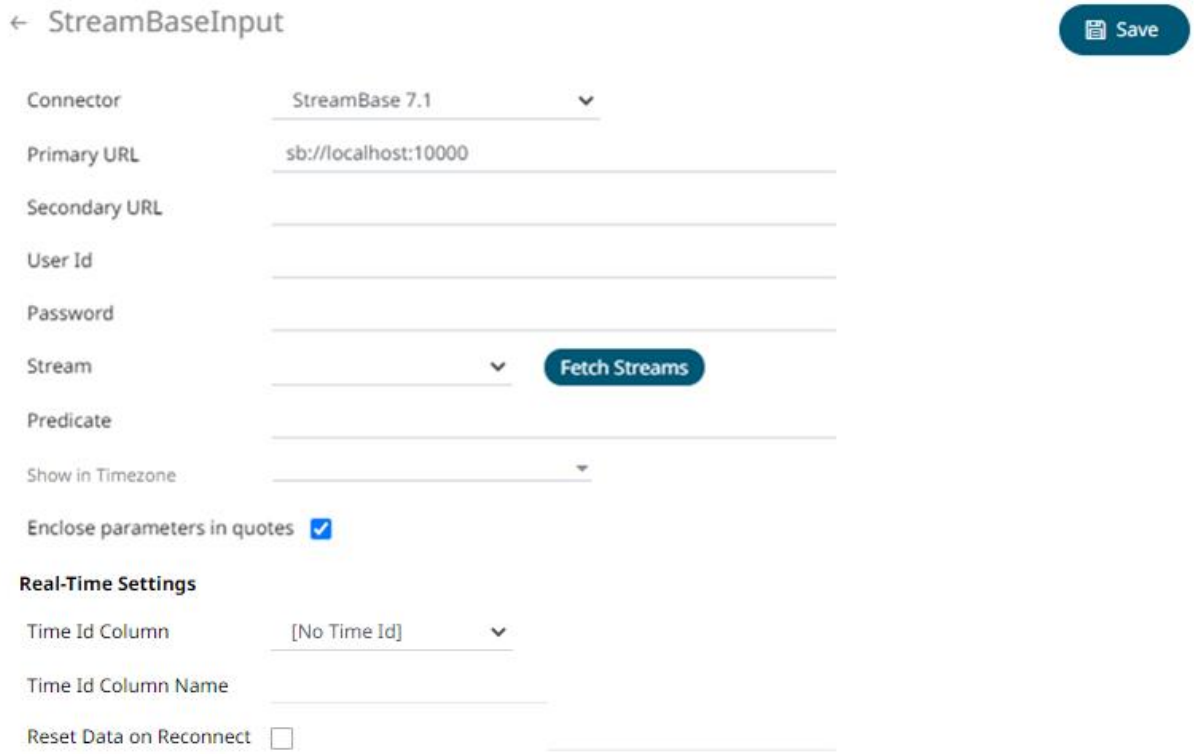
The StreamBase 7.1 connector allows connection to the StreamBase CEP engine instance on a real-time streaming basis.

To use the StreamBase connector, Streambase 7.1 redistributable must be installed.

Refer to `http://www.streambase.com/products/streambasecep/download-streambase/` for more information in downloading StreamBase products.

Steps:

1. In the *New Data Source* page, select **Input > StreamBase 7.1** in the *Connector* drop-down list.



2. Enter the following properties:

Property	Description
Primary URL	Primary URL of the StreamBase 7.1. Default is sb://localhost:**100000**.
Secondary URL	Secondary URL of the StreamBase 7.1. **NOTE:** More than two StreamBase server URLs can be specified by comma separation.
User Id	User Id that will be used to connect to StreamBase 7.1.
Password	Password that will be used to connect to StreamBase 7.1.

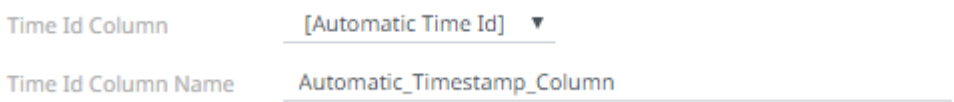
3. Click Fetch Streams to return a list of updated streams. Selection of a stream returns a list of available Id columns for the stream.

   This populates the *Id Column* with the set of columns from the schema of type `sym` and the text array such as Character/Boolean/GUID, etc. The selected *Id Column* can be used to select a key column to manage data updates and inserts.

   NOTE: Every message definition needs a text column to be defined as the *Id column*. By default, only the latest data will be loaded into memory.

   Furthermore, a streaming time series window can be generated by creating a compound key with the Id Column, plus a separately specified *Time Id* column. This *Time Id* column can be from the source dataset, or alternatively automatically generated.

   If the *Time Id* column is selected, then a scrolling time window can be specified.

   For *Automatic Time Id*, define the *Time Id Column Name*.

   As new data arrives from the subscription, new time slices will automatically be added, and old ones will be deleted.

   If a new ID is received, a new row is added to the in-memory data set representing the StreamBase topic subscription. While if an existing ID is received, an existing row is updated.

4. Enter the *Predicate* expression to force emission.

5. The time zone of input parameters and output data is by default unchanged. Changing the time zone is supported through the *Show in Timezone* list box, based on the assumption that data is stored in UTC time and outputs are presented in the selected time zone.

6. Select whether the parameters should be automatically enclosed in quotes, by checking the **Enclose parameters in quotes** box.

7. Check the **Reset Data on Reconnect** box to flush out the stale data and reload data after reconnection.

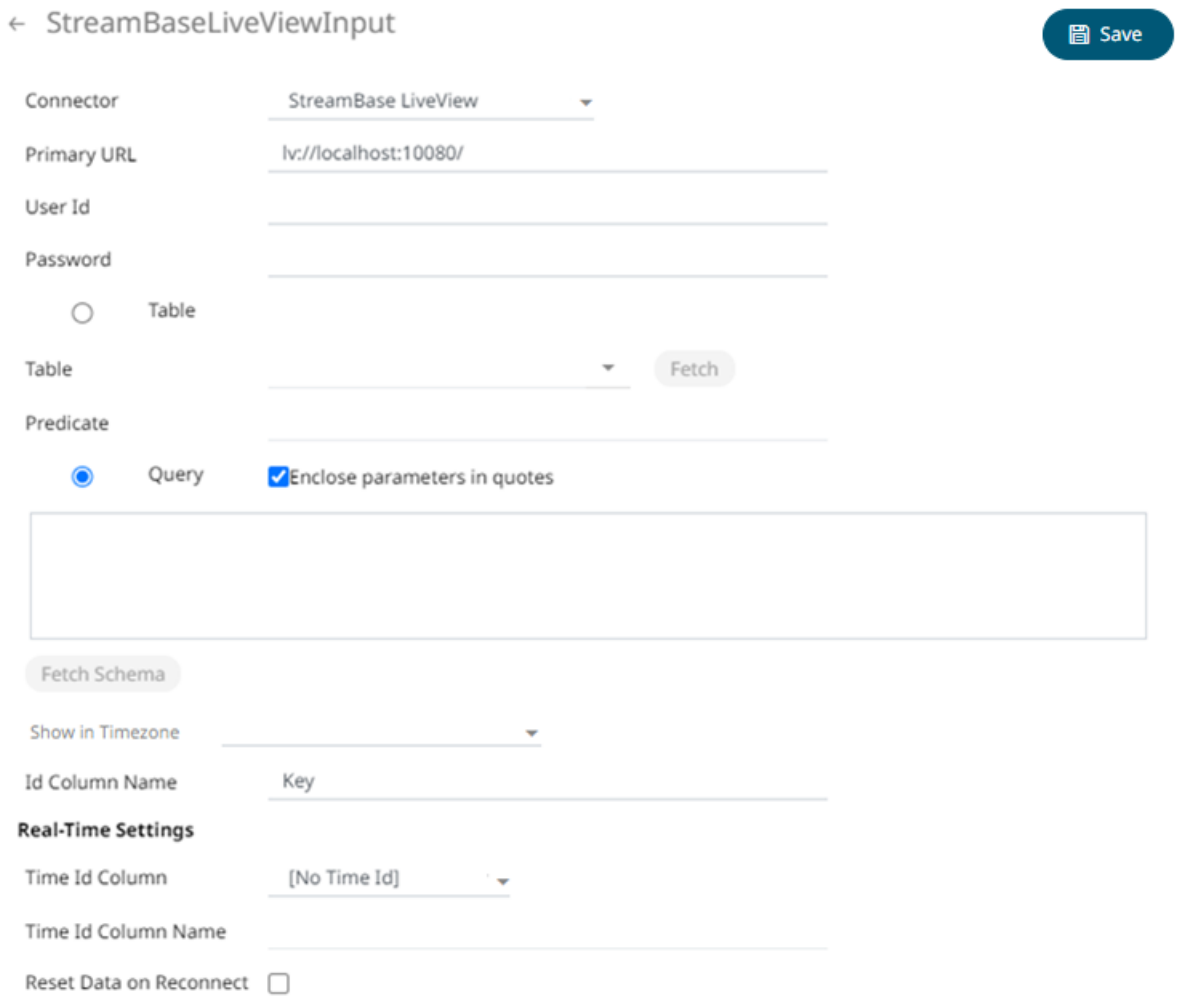
8. Click Save. The new data source is added in the *Data Sources* list.

Creating StreamBase LiveView Input Data Source

The StreamBase LiveView connector allows connection to the StreamBase LiveView instance on a real-time streaming basis.

Steps:

1. In the *New Data Source* page, select **Input > StreamBase LiveView** in the *Connector* drop-down list.



2. Enter the following properties:

Property	Description
Primary URL	Primary URL of the StreamBase LiveView.
User Id	User Id that will be used to connect to StreamBase LiveView.
Password	Password that will be used to connect to StreamBase LiveView.

3. You can either:

   - select the **Table** radio button then click

to return a list of updated *Tables*.

Select the required table.

By default, the whole table will be subscribed against. To subscribe against a subset, enter a predicate.

The `IN` syntax is recommended for use of parameters to support multiple values. The square bracket notation should be used for the `IN` clause.

Example: `color IN [{color}]`

- select the **Query** radio button, enter a full query, then click Fetch Schema .

4. The time zone of input parameters and output data is by default unchanged. Changing the time zone is supported through the *Show in Timezone* list box, based on the assumption that data is stored in UTC time and outputs are presented in the selected time zone.

5. Enter the *ID Column Name*.

   LiveView supplies a unique Id for each row. This Id field is by default given a title of **Key**.

   Id Column Name Key

   Furthermore, a streaming time series window can be generated by creating a compound key with the *Id Column*, plus a separately specified *Time Id* column. This *Time Id* column can be from the source dataset, or alternatively automatically generated.

   If the *Time Id* column is selected, then a scrolling time window can be specified.

   Time Id Column [Automatic Time Id]

   Time Id Column Name Automatic_Timestamp_Column

   For *Automatic Time Id*, define the *Time Id Column Name*.

   As new data arrives from the subscription, new time slices will automatically be added, and old ones will be deleted.

   If a new Id is received, a new row is added to the in-memory data set representing the StreamBase LiveView topic subscription. While if an existing ID is received, an existing row is updated.

6. Check the **Reset Data on Reconnect** box to flush out the stale data and reload data after reconnection.

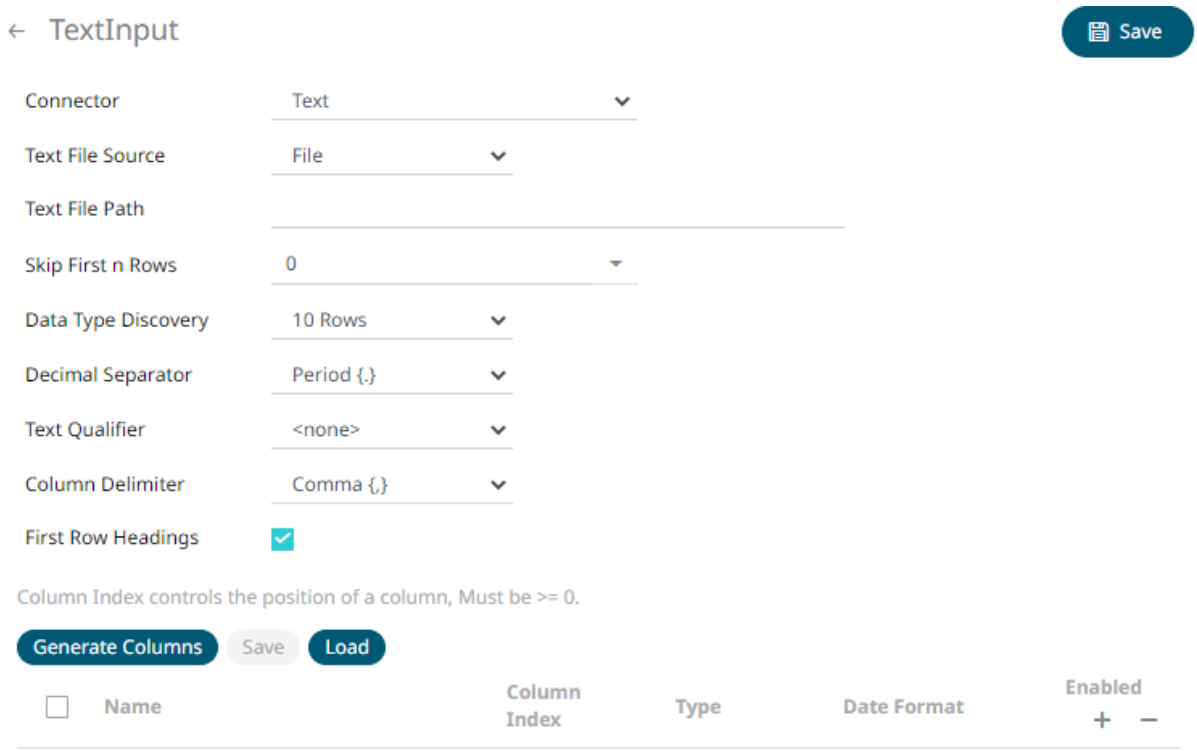
7. Click Save . The new data source is added in the *Data Sources* list.

Creating Text Input Data Source

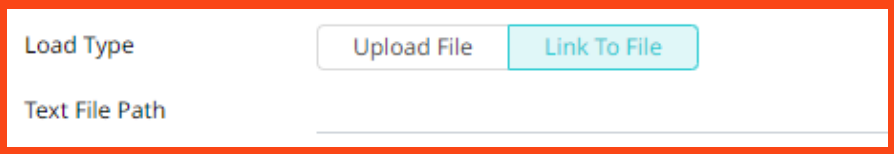
The Text connector allows the retrieval and processing of delimited Text files (such as CSV, TSV, and so on), either from a disk or from a defined URL.

Steps:

2. In the *New Data Source* page, select **Input > Text** in the *Connector* drop-down list.



3. Select the Text *File Source*.



NOTE

The Upload File button, when clicked, allows the user to choose files from their own computer. To choose files that resides on the Panopticon Server machine, use the Link to File option and fill in the *Text File Path*.

The standard settings controlling how the text file is parsed, is listed.

These include:

Property	Description
Skip First N Rows	Specifies the number of rows that will be skipped.
Data Type Discovery	Specifies how many rows from the text file should be used when automatically determining the data types of the resulting columns.
Text Qualifier	Specifies if fields are enclosed by text qualifiers, and if present to ignore any column delimiters within these text qualifiers.
Column Delimiter	Specifies the column delimiter to be used when parsing the text file.
First Row Headings	Determines if the first row should specify the retrieved column headings, and not be used in data discovery.

4. Click Generate Columns to the fetch the schema based on the connection details. Consequently, the list of columns with the data type found from inspecting the first 'n' rows of the input data source is populated and the Save button is enabled.

5. You can also opt to load or save a copy of the column definition.

6. You can opt to click + . A new column entry displays. Enter or select the following properties:

Property	Description
Name	The column name of the source schema.
Column Index	The column index controls the position of a column. Must be **>= 0**.
Type	The data type of the column. Can be a **Text**, **Numeric**, or **Time**
Date Format	The format when the data type is **Time**.
Enabled	Determines whether the message should be processed.

To delete a column, check its ☐ or all the column entries, check the topmost ☐ , then click − .

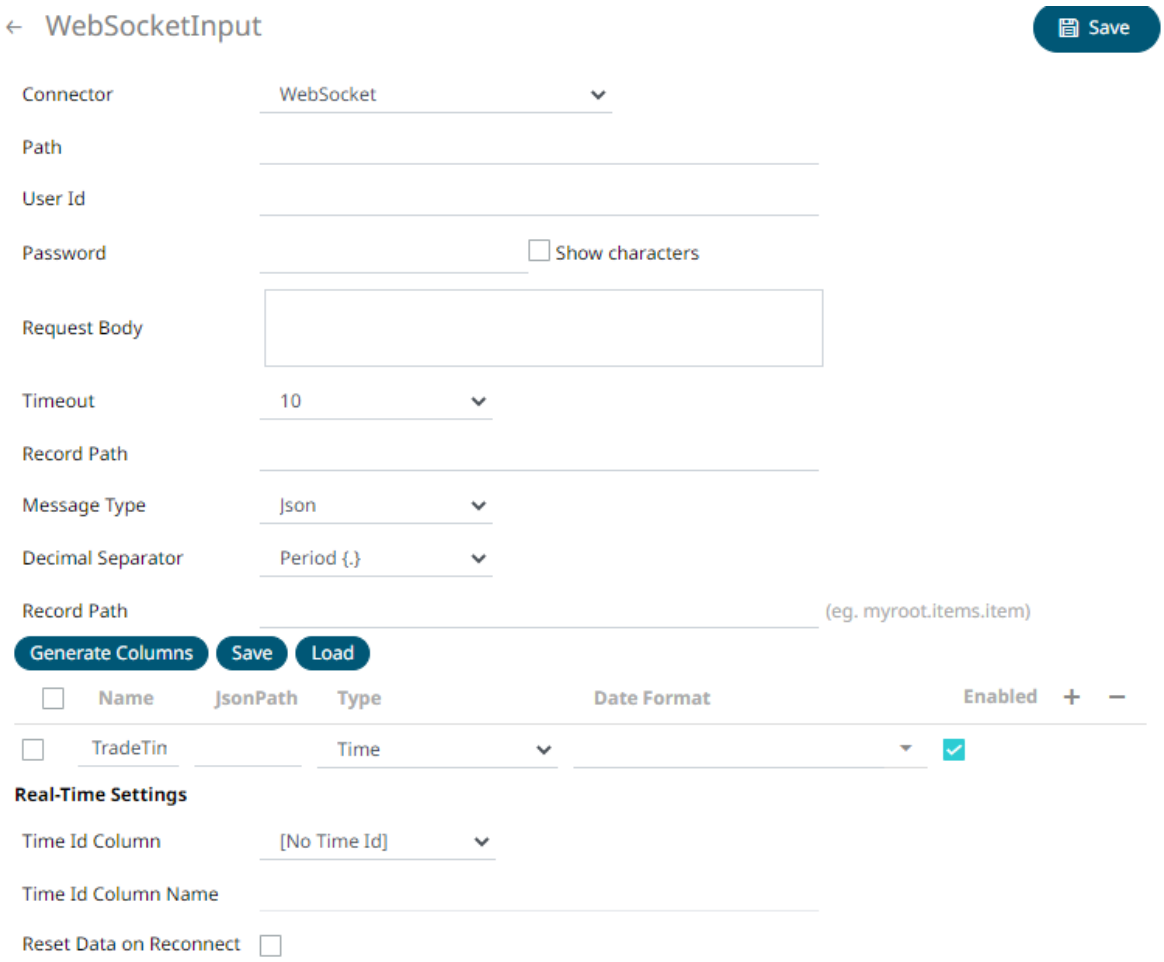
7. Click Save . The new data source is added in the *Data Sources* list.

Creating WebSocket Input Data Source

The WebSocket connector is very similar to the Stream Simulator connector, except that rather than looping through a file, it would either connect through web sockets, long polling, or repeatedly poll an external URL for new records to process.

Steps:

1. In the *New Data Source* page, select **Input > WebSocket** in the *Connector* drop-down list.



2. Enter the connection details:

Property	Description
Path	The path to which the WebSocket server will respond to.
Proxy Server URI	he HTTP Proxy setting that will allow the WebSocket connector to reach the endpoint
User ID	The User ID that will be used to connect to the WebSocket server.
Password	The password that will be used to connect to the WebSocket server. Check the **Show Characters** box to display the entered characters.
Request Body	For both the HTTP and ws:// POST requests sent to the WebSocket server.
Timeout	The length of time to wait for the server response (10 to 300). Default is **10**.

3. Select the *Message Type*.

4. Select either the period (.) or comma (,) as the *Decimal Separator*.

NOTE **Prepend 'default:' for the elements falling under default namespace.**

5. Click Generate Columns to the fetch the schema based on the connection details. Consequently, the list of columns with the data type found from inspecting the first 'n' rows of the input data source is populated and the Save button is enabled.

6. You can also opt to load or save a copy of the column definition.

7. You can opt to click + to add columns to the WebSocket connection that represent sections of the message. Then enter or select:

Property	Description
Name	The column name of the source schema.
JsonPath/Text Column Index/XPath	The JsonPath/Text Column Index/XPath of the source schema.
Type	The data type of the column. Can be a **Text**, **Numeric**, or **Time**
Date Format	The format when the data type is **Time**.
Filter	Defined parameters that can be used as filter.
Enabled	Determines whether the message field should be processed.

NOTE **To parse and format times with higher than millisecond precision, the format string needs to end with a period followed by sequence of upper case S. There can be no additional characters following them.**

For example: yyyy-MM-dd HH:mm:ss.SSSSSS

8. Define the Real-time Settings.

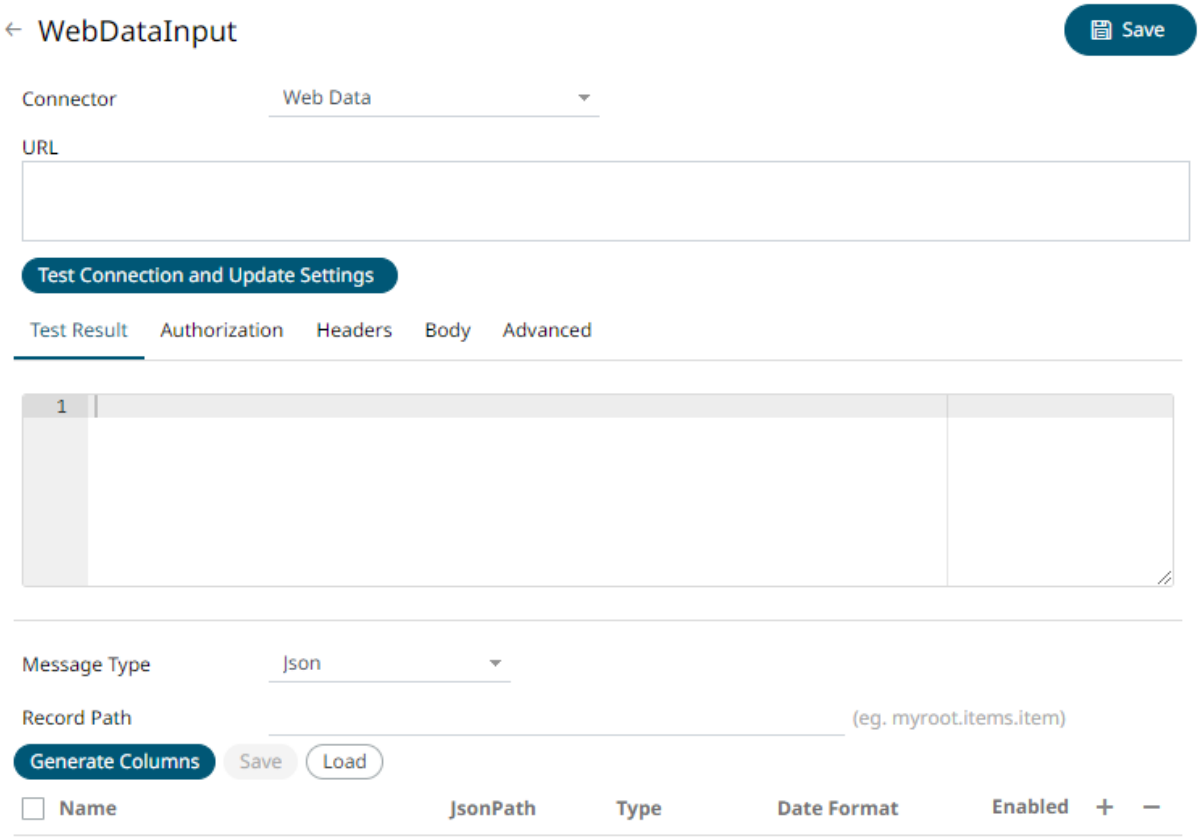
9. Click Save **.** The new data source is added in the *Data Sources* list.

Creating Web Data Input Data Source

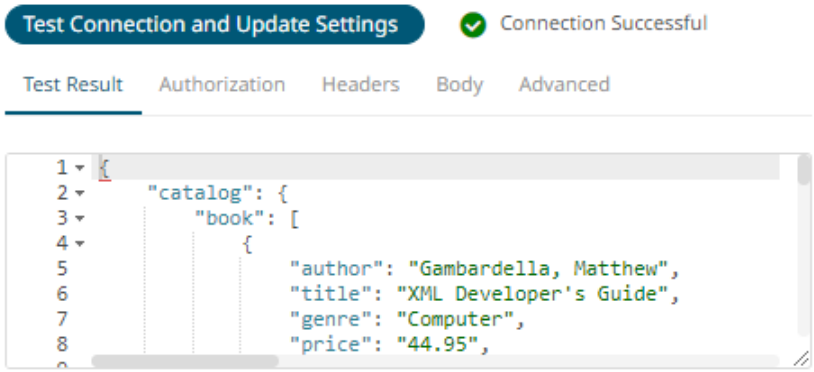
The Web Data connector allows the retrieval and processing of JSON, delimited Text (such as CSV, TSV), and XML data that is accessible over HTTP/HTTPS.

Steps:

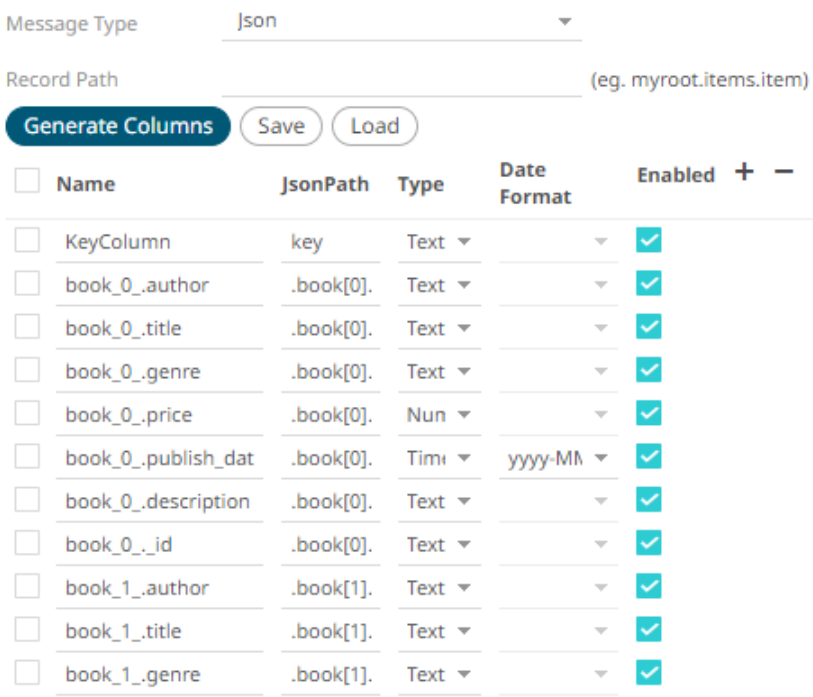
1. In the *New Data Source* page, select **Input > Web Data** in the *Connector* drop-down list.



2. Enter the absolute URL of the web data including scheme (HTTP/HTTPS) into the *URL* field.
3. Click **Test Connection and Update Settings**. A successful connection will result to the following:
   - The **Connection Successful** status is displayed along with some raw data returned by the server.



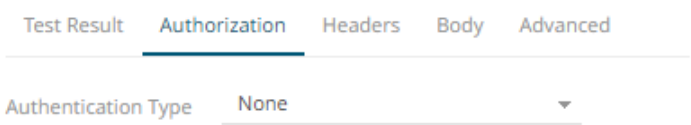
   - Connector tries auto discovery of the *Message Type*.



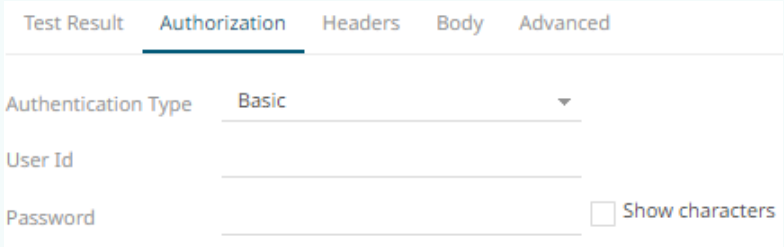
You can also opt to load or save a copy of the column definition.

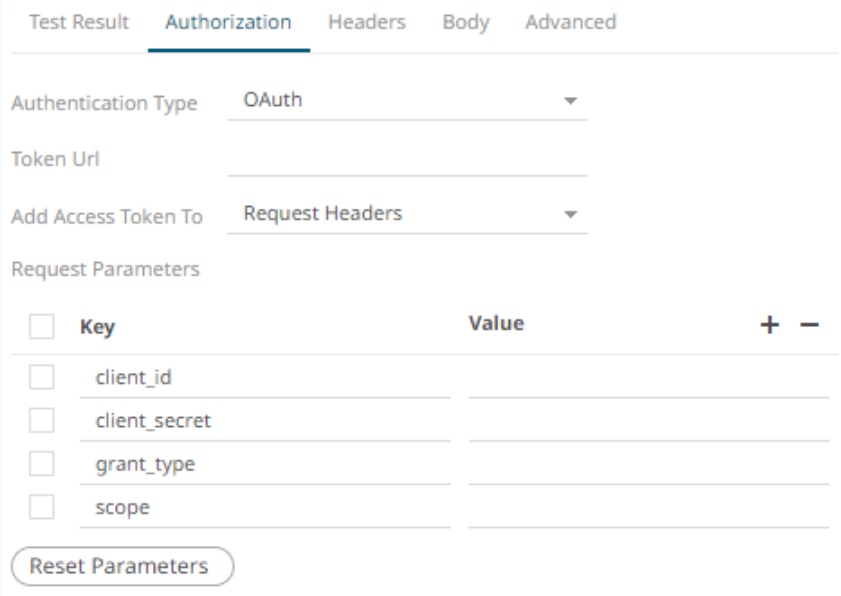
4. Adjust the *Authentication Type*, if required.

   4.1. Select the **Authorization** tab.



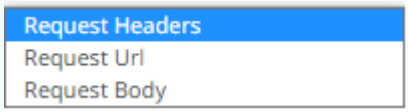
   4.2. Set the required settings:

Authentication Type	Description
None	No authentication needed.
Basic	 Enter the *User ID* and *Password* to connect to the connector's service. Select the **Show Characters** box to display the entered characters.
OAuth	Some standard OAuth token request parameters come initialized with empty values. You can set these values and also add/remove any/all of the keys.



Enter or select the following settings:

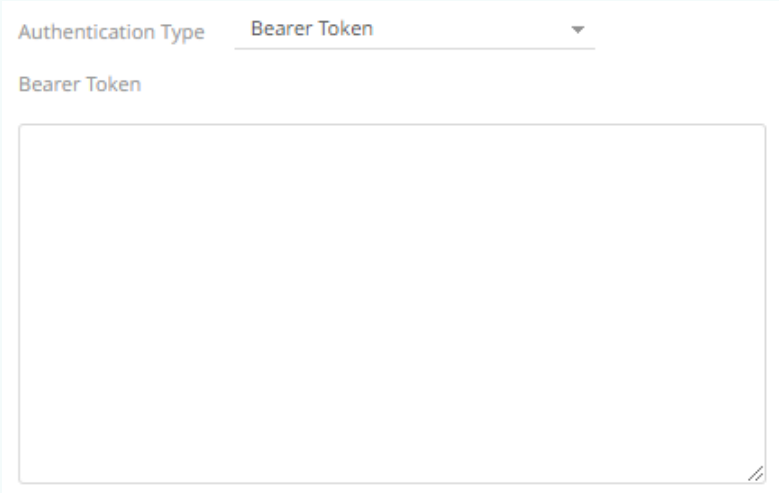
- **Token URL** – The URL to retrieve the access token from.
- **Add Access Token To** - The Access token retrieved from the *Token URL* can be added to headers, URL or request body, depending on how the endpoint needs the token.



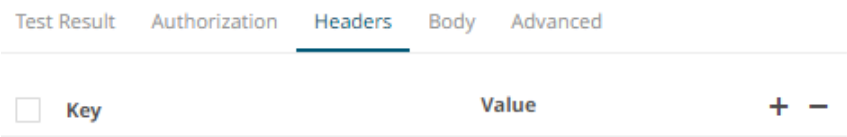
  - Request Headers - A header is automatically added to the REST API request.
  - Request URL - The URL needs to be manually parameterised with a {access_token} parameter, before calling the REST API, the parameter is replaced with the actual token.
  - Request Body - The Request Body needs to be manually parameterised with a {access_token} parameter, before calling the REST API, the parameter is replaced with the actual token.

  NOTE:

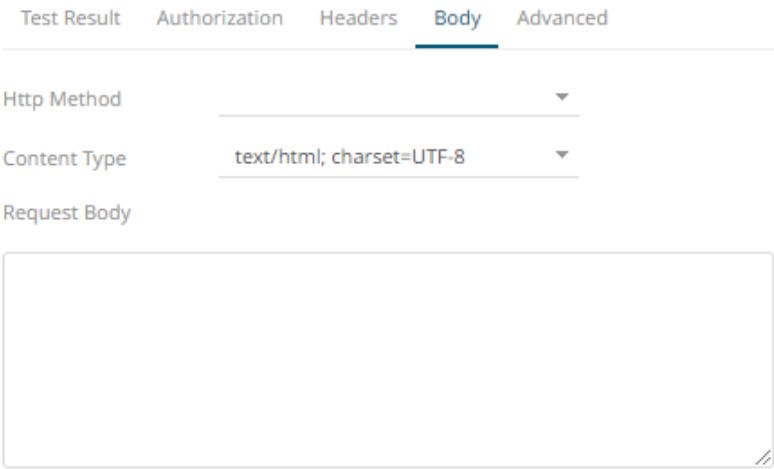
  The given request parameters key/value pairs are formatted by the connector as **application/xwww-form-urlencoded**, and posted to *Token URL*.

Bearer Token	 If you already have a authentication token, enter the token string into *Bearer Token* input box. This can be parameterized also.

5. The **Headers** tab allows you to enter any custom headers required to be passed to the URL endpoint, typically to provide additional metadata. Enter any key/value pairs you need and the connector will send them along with request.



6. Set the *Body*, if a POST/PUT request is required.
   6.1. Select the **Body** tab.

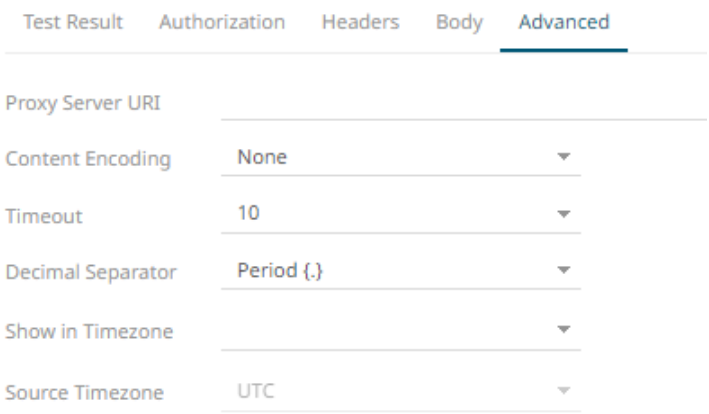


   6.2. Set the required settings:

Property	Description
HTTP Method	Select the appropriate HTTP method for the request from the following options:

	• POST – add new data • PUT – replace existing data
Content Type	Select or enter content-type based on request body (payload) format.
Request Body	The Request Body for the HTTP POST method.

7. Set the *Advanced* settings, if needed.
   7.1. Select the **Advanced** tab.



   7.2. Set the required settings:

Property	Description
Proxy Server URI	The HTTP Proxy setting that will allow the connector to reach the endpoint.
Content Encoding	Select the *Content Encoding* with the HTTP Header: **None, GZip, Deflate,** or **GZip and Deflate**
Timeout	The length of time to wait for the server response (10 to 300). Default is **10**.
Decimal Separator	Select either the dot (.) or comma (,) as the decimal separator.

8. Click

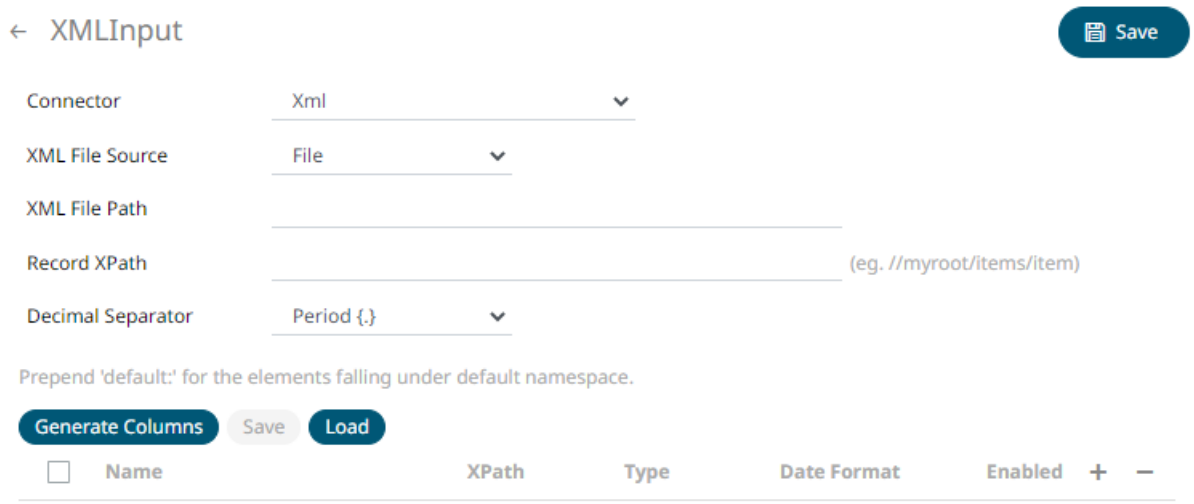
. The new data source is added in the *Data Sources* list.

Creating XML Input Data Source

The XML connector allows the retrieval and processing of XML files, either from a disk, a Text, or from a defined URL.

Steps:

1. In the *New Data Source* page, select **Input > Xml** in the *Connector* drop-down list.



2. Select the XML *File Source*.
3. Enter the Record XPath (e.g., `//myroot/items/item`).
4. Select either the period (.) or comma (,) as the *Decimal Separator*.
5. Click **Generate Columns** to the fetch the schema based on the connection details. Consequently, the list of columns with the data type found from inspecting the first 'n' rows of the input data source is populated and the Save button is enabled.
6. You can also opt to load or save a copy of the column definition.
7. You can opt to click + . A new column entry displays. Enter or select the following properties:

Property	Description
Name	The column name of the source schema.
XPath	The XPath of the source schema.
Type	The data type of the column. Can be a **Text, Numeric,** or **Time**
Date Format	The format when the data type is **Time**.
Enabled	Determines whether the message should be processed.

To delete a column, check its ☐ or all the column entries, check the topmost ☐ , then click − .

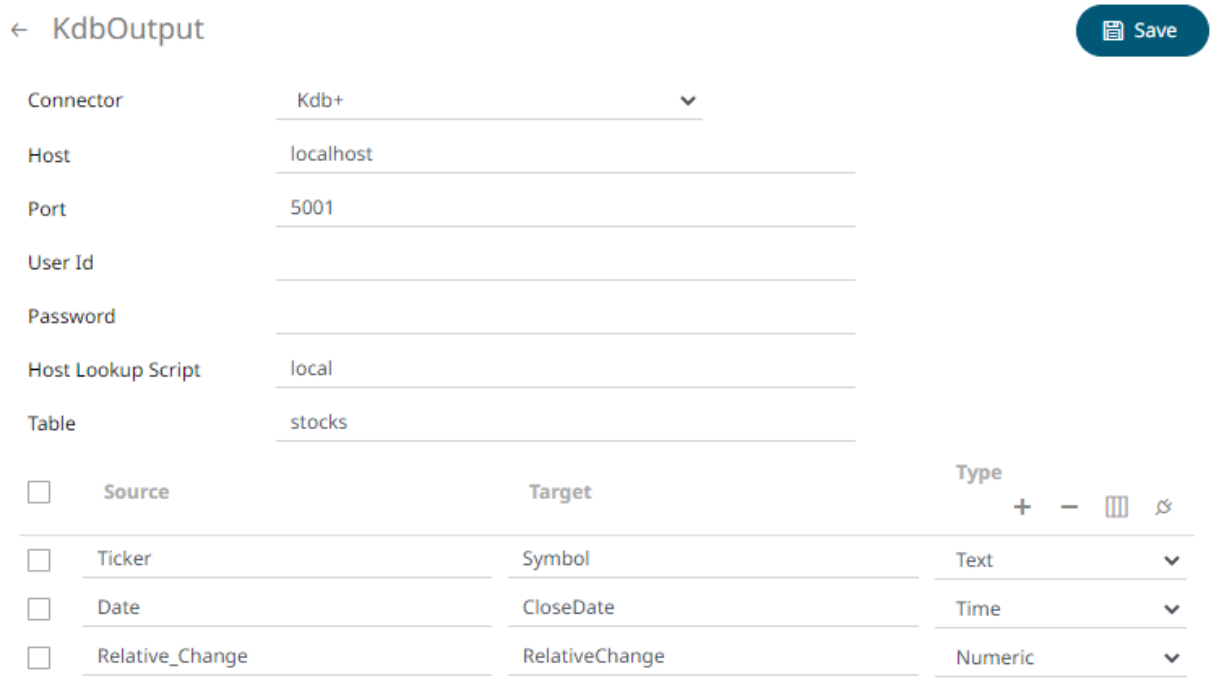
8. Click **Save** . The new data source is added in the *Data Sources* list.

MODIFYING DATA SOURCES

Steps:

1. On the **Data Sources** tab, click the link of a data source you want to modify.

   The corresponding data source page is displayed.



   All of the controls that are editable can be modified.

2. Make the necessary changes then click or the icon. The context menu displays with two saving options:

   -

     Click to save the changes made in the data source.

   -

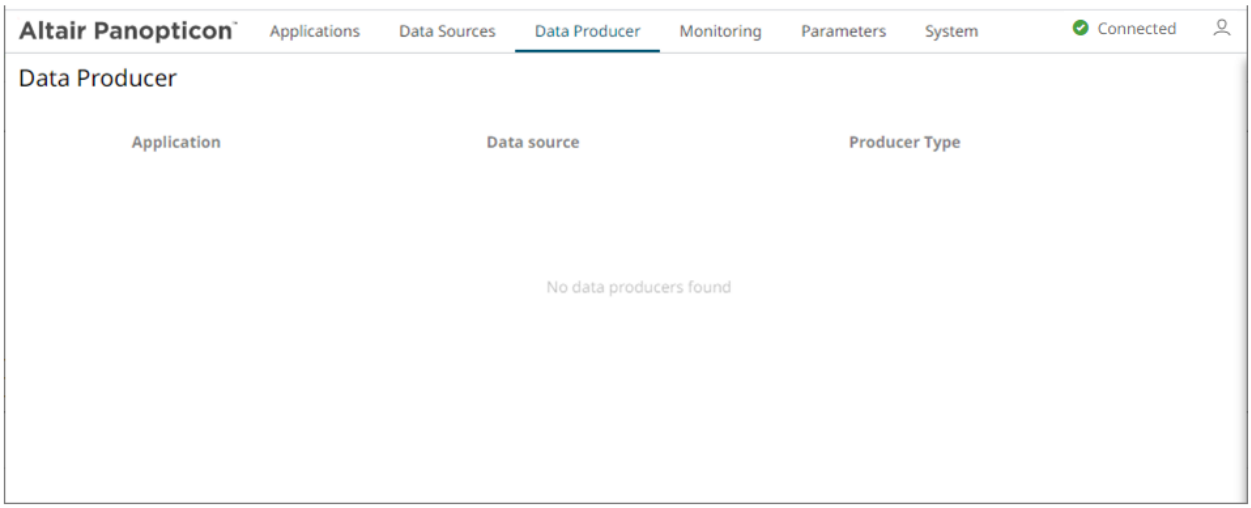
     Click to make a duplicate of the data source. The original name is appended with **_Copy**.

     To change the *Data Source Name*, click on it to make it editable, then enter a new one and click ✓ .

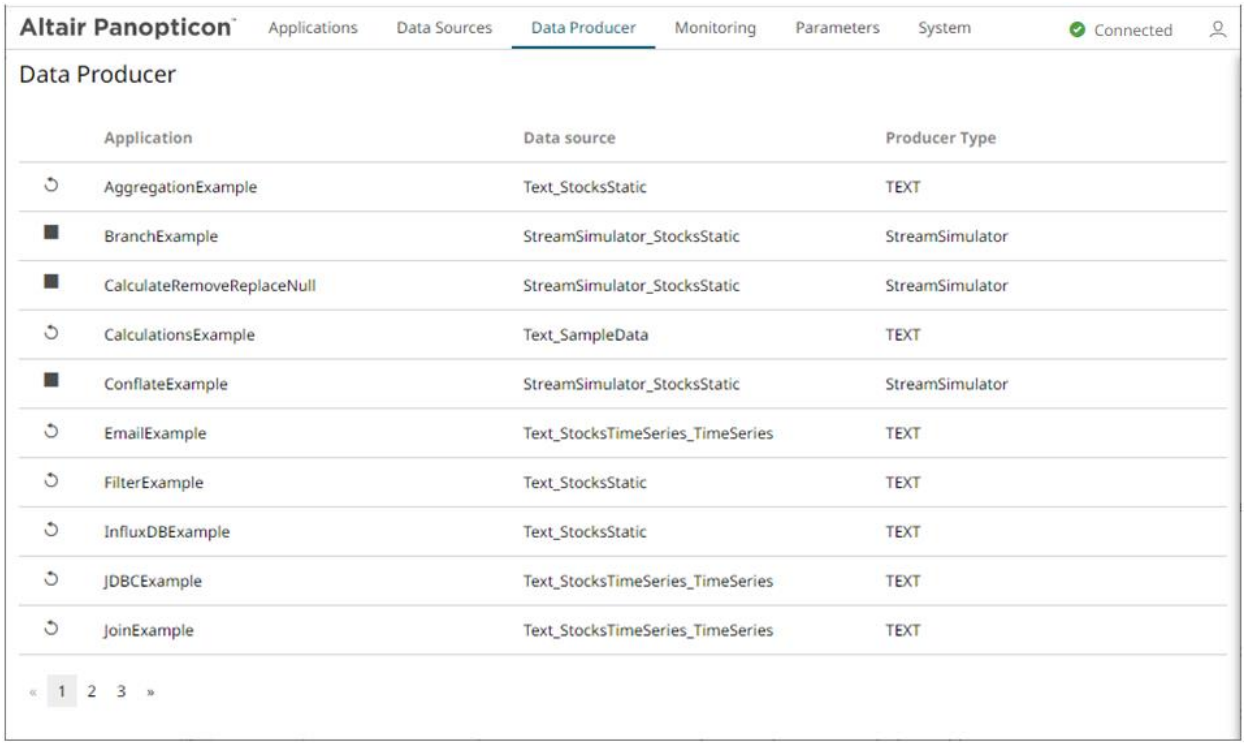
[10] MANAGING DATA PRODUCERS

When an application has been started, the data producers used to generate data from the data sources will be displayed on the **Data Producer** tab where you can:

- Refresh data producers
- Start or Stop data producers



Data Producer tab. Initially, no records are displayed when there are no running applications or the applications that are running have no data producers



Data Producer tab with data producers currently started

Refresh Data Producers

Steps:

1. On the **Data Producers** tab, click the Refresh icon of a data producer.

   A confirmation message displays.

2. Click Yes.

Starting or Stopping Data Producers

To start a Data Producer:

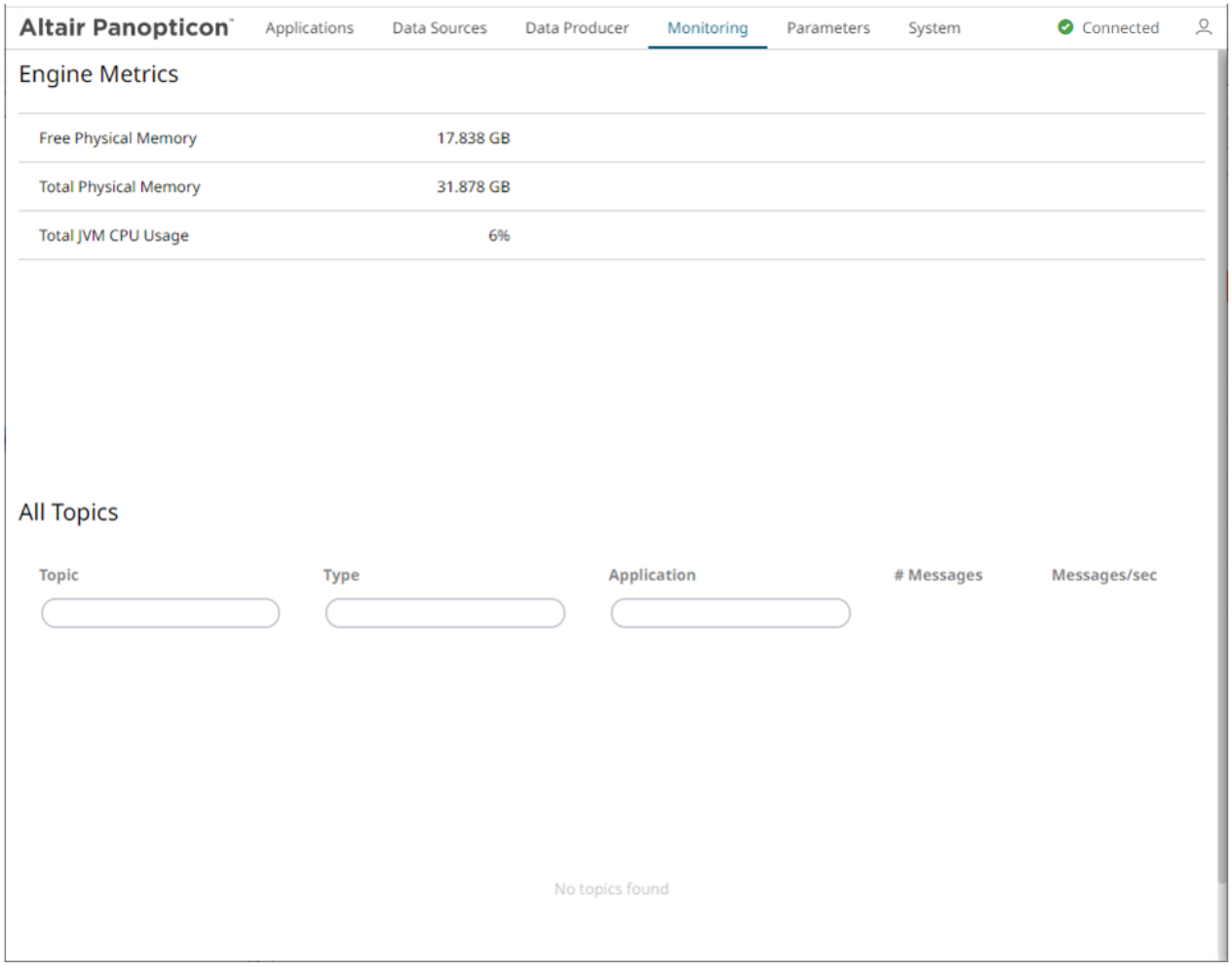
1. Click . A confirmation message displays.
2. Click **Yes**. The icon changes to .

To stop the Data Producer:

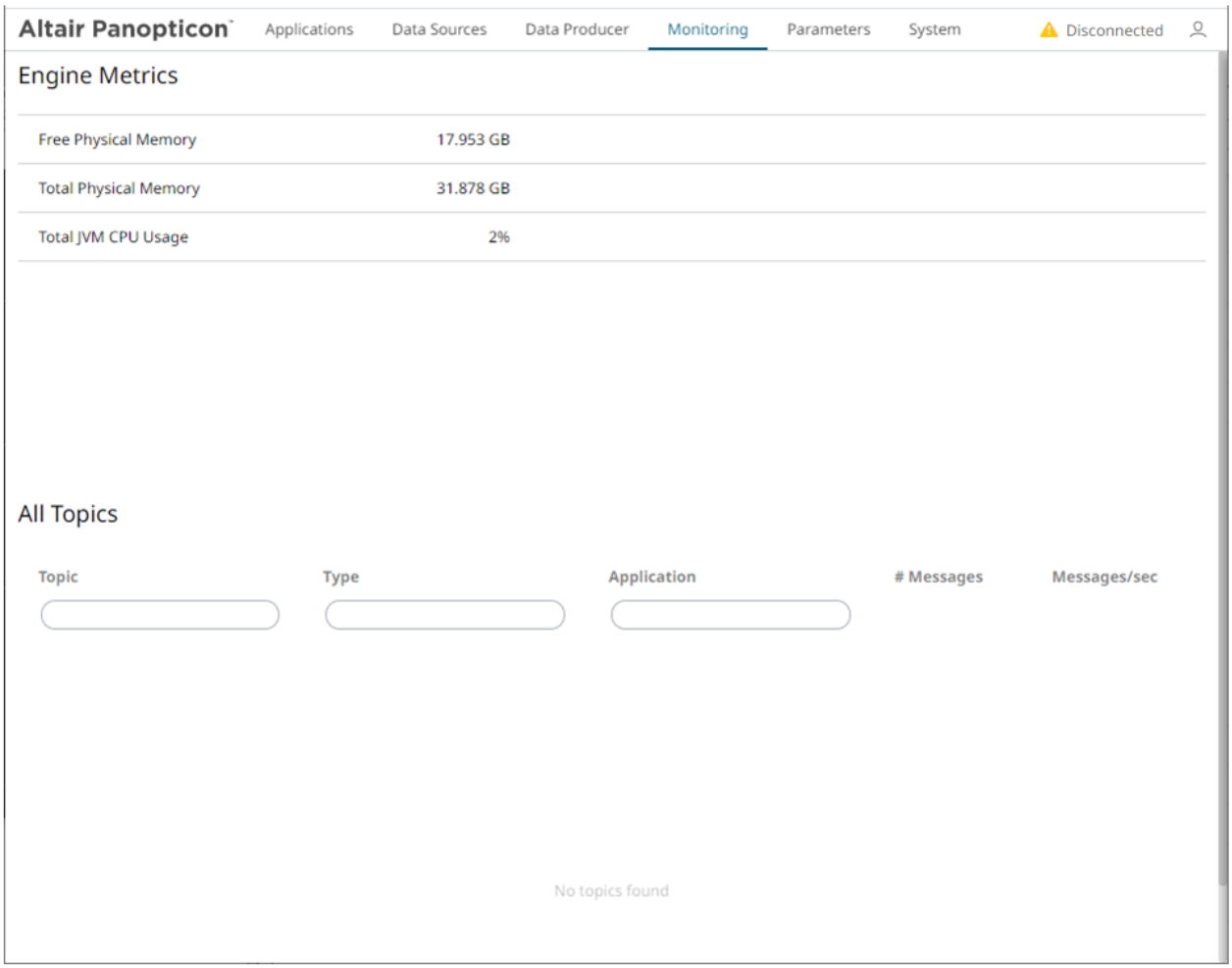
1. Click . A confirmation message displays.
2. Click **Yes**. The icon changes to .

[11] MONITORING ENGINE METRICS AND APPLICATION TOPICS

The **Monitoring** tab provides the ability to monitor the engine metrics that can help determine which part of the application is causing data bottlenecks, among others.

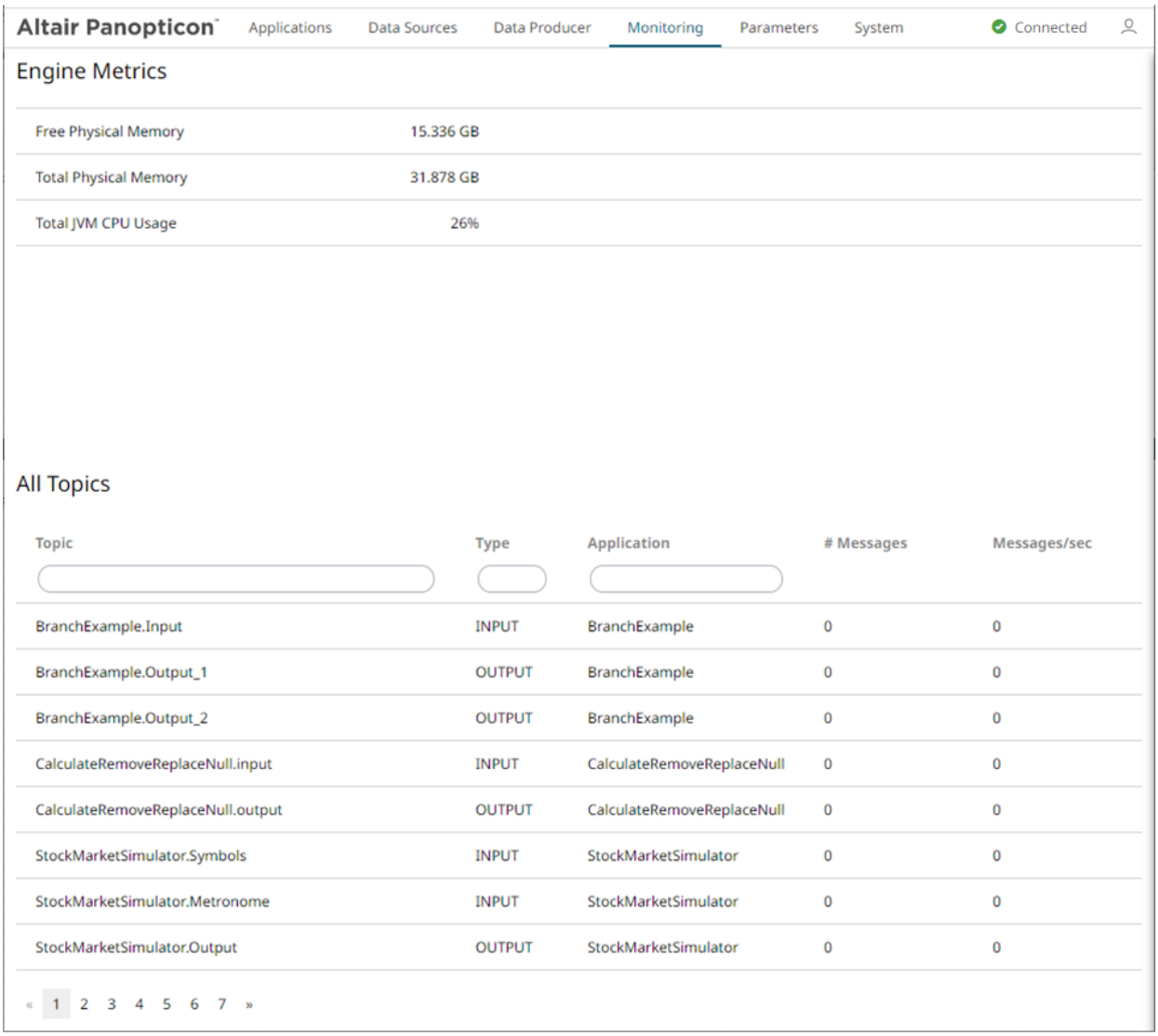


Engine Metric	Description
Free Physical Memory	The amount of free physical memory available to the Panopticon Streams server.
Total Physical Memory	The total amount of physical memory.
Total JVM CPU Usage	The recent CPU usage for the Java Virtual Machine process.



Monitoring tab when disconnected to the engine

It also displays the list of input and output topics currently running.



Monitoring tab when the engine has been started along with some applications. The list of input and output topics is displayed.

MANAGING TOPICS

While running or executing an application, input and output topics are retrieved and displayed on the **Monitoring** tab.

You can perform the following:

- View and monitor the number of retrieved messages and the number of retrieved messages per second
- Define a filter among the topics
- Sort the list of topics

Filter Topics

The topics can be filtered by entering letters, numbers, or underscores in the *Topic* or *Application* text box.

For the *Type* of application, enter a text (either **Output** or **Input**) into the text box above the listing.

Sorting the List of Topics

Modify the sorting of the list by clicking the ↓ or ↑ button of the *Topic*, *Type*, *Application*, *#Messages*, or *#Messages/sec* column. The icon beside the column that was used for the sorting will indicate if it was in an ascending or descending order.

Moving to Other Topics List Pages

Go to the other topics pages by clicking:

- any link of a page number
- « . This displays the previous page
- » . This displays the next page

[12] MANAGING PARAMETERS

The **Parameters** tab supports adding, modifying, and deleting global parameters that will pull and enter specific data into the different components of an application model.

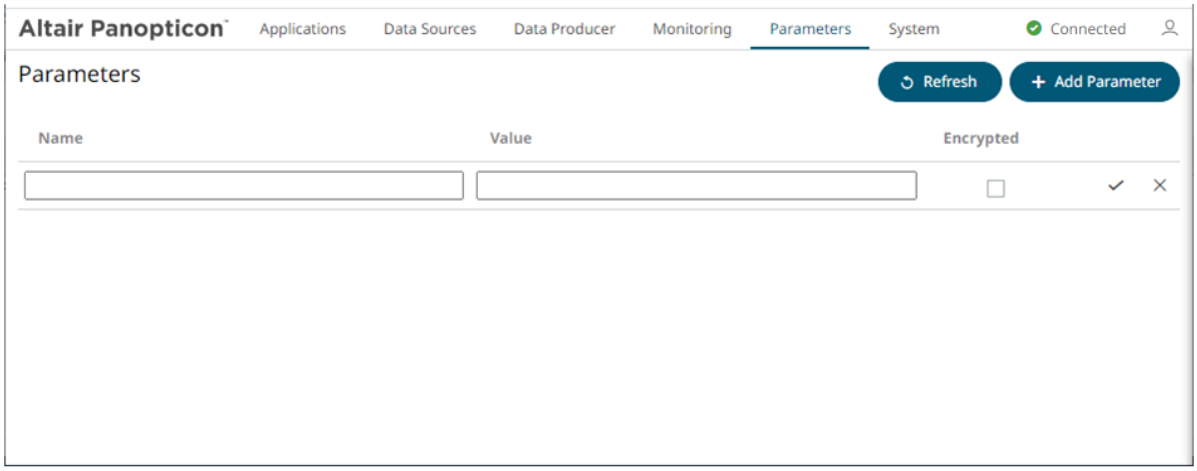
ADDING PARAMETERS

Steps:

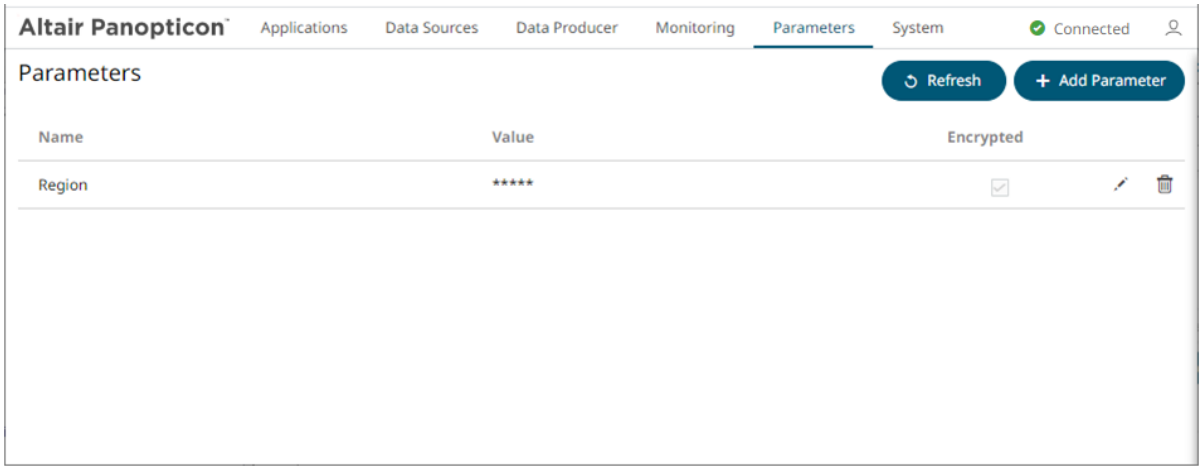
1. On the Parameters tab, click

.

   A new parameter entry displays.



2. Enter a *Name* for the new parameter and the *Value*.
3. Check the *Encrypted* box to encrypt the value.
4. Click ✓. The new parameter is added in the list.



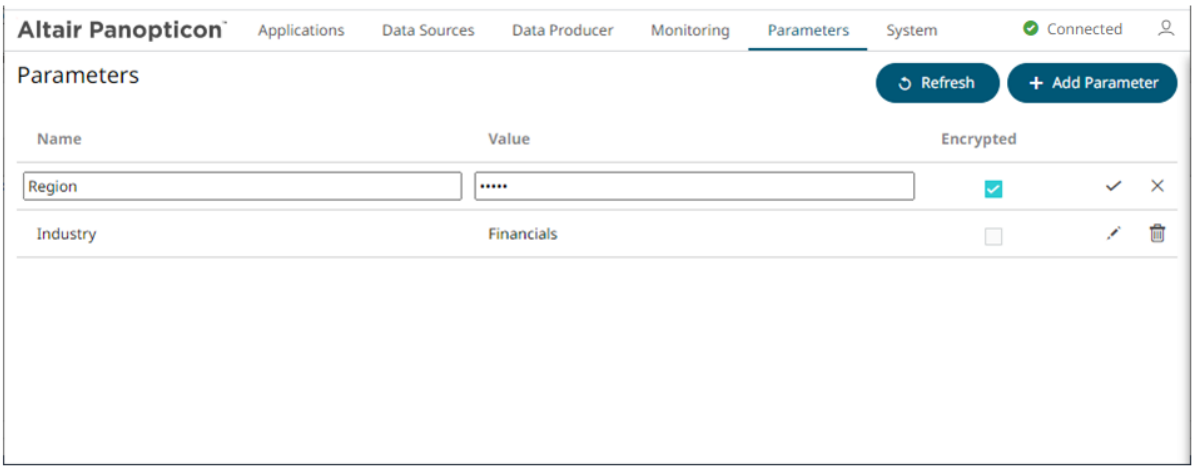
New parameters are added in the `Parameters.json` file located in the `AppData` folder (i.e., `c:\streamsseverdata`).

Modifying Parameters

Steps:

1. On the **Parameters** tab, click the **Edit** icon of a parameter you want to modify.

   The *Name, Value, and Encrypted* controls are enabled.



2. Make the necessary changes then click .

Deleting Parameters

Steps:

1. On the **Parameters** tab, click of a parameter you want to delete.

   A confirmation message displays.

2. Click Yes to delete.

Refresh Parameters

Click Refresh to refresh the values that are being pulled by the application models.

Sorting the List of Parameters

By default, the parameters are listed based on the sequence that they were added. Modify the sorting of the list by clicking the ↓ or ↑ button of the *Name, Value,* or *Encrypted* columns. The icon beside the column that was used for the sorting will indicate if it was in an ascending or descending order.

[13] EXAMPLE APPLICATIONS

Panopticon Streams is installed with a series of example applications:

- AggregationExample – Demonstrates how to aggregate data based on a grouping key and a set of aggregated fields.

  Includes simple aggregations such as avg, count, first, last, max, min, samples, sum, sdevp, sdevs, Sum, varp, and vars.

- BranchExample – Demonstrates how to split a stream into one or more branches.
- CalculateRemoveReplaceNull – Demonstrates how to:
  - remove and replace fields from output schemas
  - set a field value to null
  - set a field value to the current timestamp
- CalculationExample – Includes the SquareRoot calculation.
- CalculationsExample – Includes the following calculations:
  - Numeric calculations such as Abs, SquareRoot, Subtract, Multiply, Divide, Truncate, IF
  - Text calculations such as Upper, Lower, Proper, Left, Right, Mid, Concat, Find
  - Time Period calculations such as DateDiff

  In addition, data type casting between Text, Number, and Date/Time

- ConflateExample – Demonstrates how to lower the frequency of updates by setting a fixed interval.
- EmailExample – Shows how to send an email via SMTP where the SMPT and email settings can be parameterized. Each record passed to the connector results in an email which can be primarily used as an output for alerting, having a conditional expression that would need to be fulfilled for a record to be forwarded to the output.

  Requires the EmailWriter plugin.

- ExternalInputExample – Demonstrates how to directly source data from a Kafka topic (defined in the schema registry with the message format set to Avro).
- ExternalInputJsonParserExample – Demonstrates how to directly use a parsed input Json data.
- ExternalInputXMLParserExample - Demonstrates how to directly use a parsed input XML data.
- FilterExample – Demonstrates how to filter a data source based on a predicate.
- InfluxDBExample - Allows periodical dumping of records from a Kafka topic into an InfluxDB output connector. Requires the InfluxDBWriter plugin.
- JDBCExample – Allows periodical dumping of records from a Kafka topic into a JDBC database output connector. Requires the JDBCWriter plugin.
- JoinExample – Demonstrates how to join a stream to a global table.
- KdbExample - Allows periodical dumping of records from a Kafka topic into a Kx kdb+ output connector. Requires the KdbWriter plugin.
- MetronomeExample – Demonstrates how the metronome operator works in generating a timestamp field schema. A static metronome has a defined frequency while a dynamic metronome takes frequency as an input which determines the speed of the simulation.
- RetentionTimeExample – Demonstrates how to define the different retention time periods set for tables, input streams, output streams, and topics in an application.

This helps minimize memory utilization and the amount of data retrieved when subscribing from the beginning to the latest messages.

NOTE **Setting these properties in the application level overrides the defaults set in the `Streams.properties` file.**

For example, if the following properties are defined in the `streams.properties` file:

```
cep.kafka.table.retention.ms=86400000
cep.kafka.input.retention.ms=60000
cep.kafka.output.retention.ms=900000
```

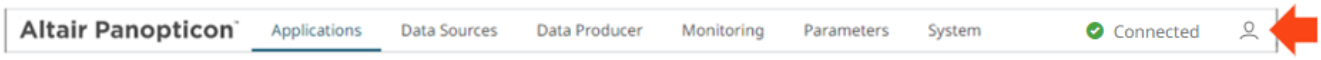
In the application level, the input retention period will be 1,000 milliseconds instead of 60,000 and the output retention period will be 1,000 milliseconds instead of 900,000. Also, a custom topic retention period has been added using the following pattern: `TopicName.retention.ms` (i.e., `TimeSeries.retention.ms)`.

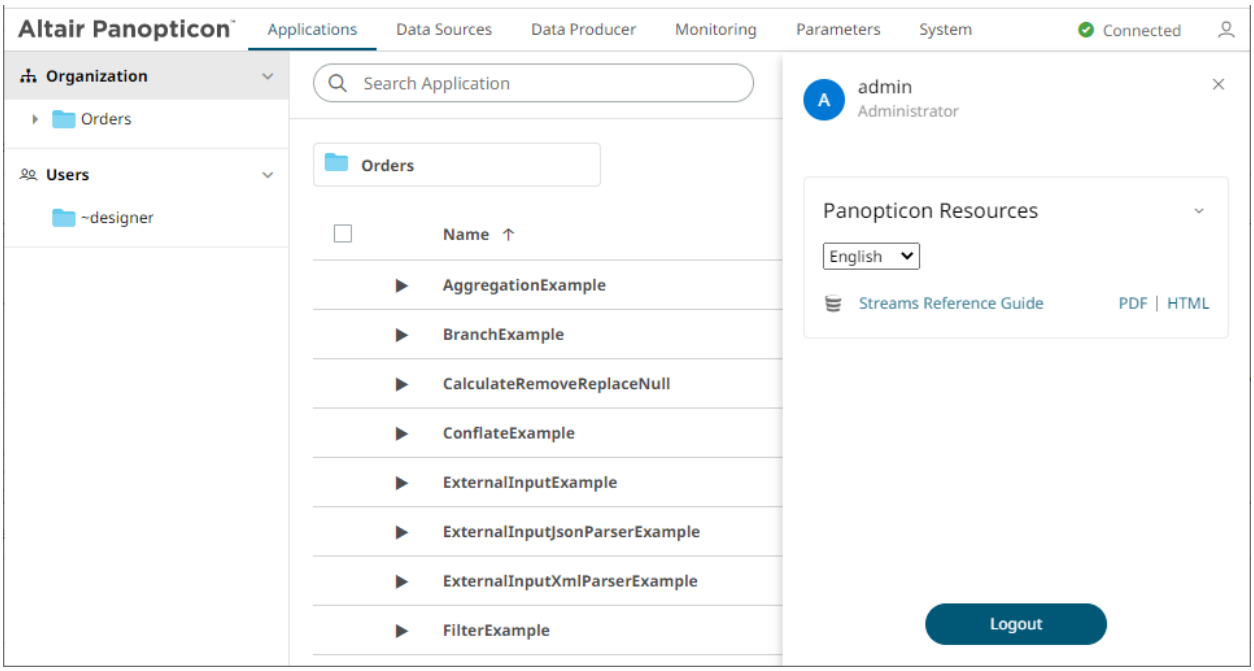
```
<properties>
    <!-- Keep tables alive one day -->
    <entry>
        <key>table.retention.ms</key>
        <value>86400000</value>
    </entry>
    <!-- Keep input and output streams for 1 second -->
    <entry>
        <key>input.retention.ms</key>
        <value>1000</value>
    </entry>
    <entry>
        <key>output.retention.ms</key>
        <value>1000</value>
    </entry>
    <!-- Custom retention time for InputStream topic -->
    <entry>
        <key>TimeSeries.retention.ms</key>
        <value>1111</value>
    </entry>
</properties>
```

- StockMarketSimulator – Shows a stock market simulation using a streaming data with join, calculations, and metronome operators.
- StockStaticTimeSeriesApp – Joins a static and a time series data sources using common keys. Also demonstrates adding a sum aggregation.
- StreamtoGlobalTableJoinExample – Joins stream and global table inputs using common keys.
- StreamToTableJoinExample - Joins stream and table inputs using common keys.
- TextExample - Allows periodical dumping of records from a stream Kafka topic into a Text connector. Requires the TextWriter plugin.
- UnionExample- Unioning of two streams.
- WindowedStreamExample – Demonstrates aggregation across a windowed stream.

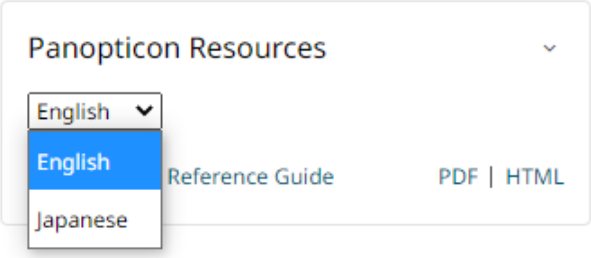
[14] PANOPTICON RESOURCES

Clicking on the top right section of the toolbar displays the available Panopticon online resources.





Select the *Language* on the drop-down list: **English** or **Japanese**.



Then click *Panopticon Streams Reference Guide* either on a PDF or HTML Help format. This guide is also available upon installation.

[APPENDIX]

PROPERTIES: STREAMS

The `Streams.properties` file located in the `AppData` folder (i.e., `c:\streamsserverdata`), contains majority of properties for controlling the configuration of Panopticon Streams. The following properties can be overridden by updating the file.

Property	Access
Attribute	`access.administrator.groups`
Description	The role that is mapped to the administrator group.
Default Value	**admin**
Property	Access
Attribute	`access.default.roles`
Description	The default roles applied to all users of the server. For example, if access.default.roles=DESIGNER,ADMINISTRATOR and a user with a VIEWER role logs on to the server, then the user will simultaneously have a VIEWER, DESIGNER, and ADMINISTRATOR roles. However, if no default roles are wanted, then leave the property blank. **NOTE:** The roles that can be assigned in this property can only be ADMINISTRATOR, VIEWER, ANONYMOUS, and/or DESIGNER. This property is case sensitive.
Default Value	**VIEWER**
Property	Access
Attribute	`access.designer.groups`
Description	The role that is mapped to the designer group.
Default Value	**designer**
Property	Access
Attribute	`access.viewer.groups`
Description	The role that is assigned to the viewer group. **NOTE:** Currently not in use. Development ongoing.
Default Value	
Property	Access
Attribute	`access.list.delimiter`
Description	The value delimiter to use when parsing access groups. Examples: access.list.delimiter=, access.administrator.groups=group1,group2

	The groups are mapped to {'group1', 'group2'} access.list.delimiter=, access.administrator.groups=group1;group2,group3 The groups are mapped to {'group1;group2', 'group3'} access.list.delimiter=; access.administrator.groups=group1;group2,group3 The groups are mapped to {'group1', 'group2,group3'}
Default Value	**',' (comma)**
Property	Authentication: Header
Attribute	`authentication.header.role.delimiter`
Description	The delimiter used to separate the roles. Example: role1, role2,role3
Default Value	**, (Comma)**
Property	Authentication: Header
Attribute	`authentication.header.roles`
Description	The name of the header that contains all the roles.
Default Value	
Property	Authentication: Header
Attribute	`authentication.header.rolesdynamic`
Description	Supports the ability to create dynamic roles using free form patterns or string replacement. To create dynamic roles, use '{header value to be used}'. Example: authentication.header.rolesdynamic={HEADER_ROLES},financials,role_for_company_{HEADER_COMPANY} Given this table: **KEY** / **VALUE** HEADER_ROLES / designer, watcher HEADER_COMPANY / industrials, consumers Then the roles to create the authentication token will be the following: • designer • watcher • financials • role_for_company_industrials

	• role_for_company_consumers
Default Value	
Property	Authentication: Header
Attribute	`authentication.header.username`
Description	The name of the header that contains the username
Default Value	
Property	Authentication: Logout
Attribute	`authentication.logout.redirect.url`
Description	Takes a URL as a parameter. Clicking the logout button redirects the user to the specified URL. If this property is not set, user will be returned to the start page of Panopticon.
Default Value	
Property	Authentication: OAuth 2.0
Attribute	`authentication.oauth2.client.ID`
Description	The ID of the OAuth 2.0 client.
Default Value	
Property	Authentication: OAuth 2.0
Attribute	`authentication.oauth2.client.secret`
Description	The secret used by the OAuth 2.0 client.
Default Value	
Property	Authentication: OAuth 2.0
Attribute	`authentication.oauth2.identity.attribute.roles`
Description	The attribute that will be extracted from the identity response and used as the role. There can be multiple assigned roles for a user.
Default Value	
Property	Authentication: OAuth 2.0
Attribute	`authentication.oauth2.identity.attribute.roles.pattern`
Description	Takes regex used to extract the roles from the OAuth 2.0 server identity response. For example, the returned string: `cn=admin,ou=groups,dc=openam,dc=openidentityplatform,dc=org,cn=designer,ou=groups,dc=openam,dc=openidentityplatform,dc=org` contains two roles, **admin** and **designer** The regex to extract the roles is **cn=([^,]+)**.
Default Value	
Property	Authentication: OAuth 2.0
Attribute	`authentication.oauth2.identity.attribute.username`

Description	The attribute that will be extracted from the identity response and used as the username.
Default Value	
Property	Authentication: OAuth 2.0
Attribute	`authentication.oauth2.identity.url`
Description	The URL to the REST service that provides details about the authenticated user.
Default Value	
Property	Authentication: OAuth 2.0
Attribute	`authentication.oauth2.login.callback.url`
Description	The callback URL. The URL should be the same as one of the specified callback URLs used by the client. The URL should refer to Panopticon Streams.
Default Value	
Property	Authentication: OAuth 2.0
Attribute	`authentication.oauth2.login.redirect.url`
Description	Redirects the user to the specified URL after successfully logging in. This property can be left blank, in which case the user is redirected to the URL they requested to access.
Default Value	
Property	Authentication: OAuth 2.0
Attribute	`authentication.oauth2.login.response.type`
Description	The response type. The only response type that is currently supported is **CODE**. The value can also be left blank.
Default Value	
Property	Authentication: OAuth 2.0
Attribute	`authentication.oauth2.login.scope`
Description	The requested scope. The field can be left blank.
Default Value	
Property	Authentication: OAuth 2.0
Attribute	`authentication.oauth2.login.state`
Description	The requested state. The field can be left blank.
Default Value	
Property	Authentication: OAuth 2.0
Attribute	`authentication.oauth2.login.url`
Description	The URL to the OAuth 2.0 login resource.
Default Value	
Property	Authentication: OAuth 2.0

Attribute	`authentication.oauth2.logout.redirect.url`
Description	Logging out revokes the token from the authentication server if the property authentication.oauth2.logout.url is set to the revocation URL. If this property is not set, the server will only remove its own token. If none of these properties are set, the server will attempt to redirect to the start page of Panopticon when logging out.
Default Value	
Property	Authentication: OAuth 2.0
Attribute	`authentication.oauth2.logout.url`
Description	The URL to the OAuth 2.0 logout resource. This field can be left blank.
Default Value	
Property	Authentication: OAuth 2.0
Attribute	`authentication.oauth2.token.method`
Description	The method on how the token should be retrieved. Supported values are **QUERY**, **BODY,** and **HEADER.**
Default Value	
Property	Authentication: OAuth 2.0
Attribute	`authentication.oauth2.token.url`
Description	The URL to the OAuth 2.0 token resource.
Default Value	
Property	Service authentication level
Attribute	`authentication.role`
Description	The authentication role.
Default Value	
Property	Service authentication level
Attribute	`authentication.required`
Description	The property that will make the authentication required. It will force the user to login in order to use any of the services provided by the server.
Default Value	**true**
Property	Authentication: SAML
Attribute	`authentication.saml.serviceprovider.id`
Description	The ID of the service provider configured in the IdP.
Default Value	
Property	Authentication: SAML
Attribute	`authentication.saml.assertionconsumerservice.url`
Description	The URL to the Panopticon assertion consumer service. URL: [Protocol]://[Host]:[Port]/[Context]/server/rest/auth/login

Default Value	
Property	Authentication: SAML
Attribute	`authentication.saml.identityprovider.url`
Description	The URL to the IdP login service.
Default Value	
Property	Authentication: SAML
Attribute	`authentication.saml.assertion.username`
Description	User attribute for username configured in the IdP.
Default Value	
Property	Authentication: SAML
Attribute	`authentication.saml.assertion.roles`
Description	User attribute for roles configured in the IdP.
Default Value	
Property	Authentication: SAML
Attribute	`authentication.saml.certificate.name`
Description	The name of the certificate used to validate signature and/or sign outgoing SAML messages
Default Value	
Property	Authentication: SAML
Attribute	`authentication.saml.certificate.password`
Description	The password of the certificate used to validate signature and/or sign outgoing SAML messages.
Default Value	
Property	Authentication: SAML
Attribute	`authentication.saml.challenge.required`
Description	This property determines whether the IdP-first authentication with SAML is enabled or not. To enable, set this property to **false**.
Default Value	**true**
Property	Authentication: SAML
Attribute	`authentication.saml.identityprovider.logout.url`
Description	The URL to the IdP logout service.
Default Value	
Property	Authentication: SAML
Attribute	`authentication.saml.keystore.file`
Description	The location of the Keystore file that contains the certificate.

Default Value	
Property	Authentication: SAML
Attribute	`authentication.saml.keystore.password`
Description	The password to the Keystore file.
Default Value	
Property	Authentication: SAML
Attribute	`authentication.saml.identityprovider.certificate.file`
Description	Takes a file path to a certificate file that contains the IdP's public key.
Default Value	
Property	Authentication: SAML
Attribute	`authentication.saml.identityprovider.signature.validation.required`
Description	Specifies whether to require a valid IdP signature to be present on the SAML response. Default value is **false**.
Default Value	**false**
Property	Authentication: SAML
Attribute	`authentication.saml.provider`
Description	The IdP provider. Possible values are **OPENSAML**, **OPENAM**.
Default Value	**OPENSAML**
Property	Authentication: SAML
Attribute	`authentication.saml.keystore.type`
Description	The key store type. Possible values are **JKS, JCEKS**, **PKCS12**.
Default Value	**JKS**
Property	Authentication: SAML
Attribute	`authentication.saml.login.redirect.url`
Description	Redirects the user to the specified URL after successfully logging in. This property can be left blank, in which case the user is redirected to the URL they requested to access.
Default Value	
Property	Authentication: SAML
Attribute	`authentication.saml.logout.redirect.url`
Description	Redirects the user back to the specified URL after logging out. This is mainly used with a proxy. In which case, Panopticon Real Time does not know the endpoint which the user is going towards to, and therefore cannot redirect the user back to the Overview page. If you are using OpenAM this is required, otherwise this property can be left blank.
Default Value	

Property	Authentication: SAML
Attribute	`authentication.saml.openam.meta.alias`
Description	The meta alias for the IdP if you are using OpenAM.
Default Value	
Property	Authentication: SAML
Attribute	`authentication.saml.protocolbinding`
Description	Protocol binding for the use of SAML authentication. Possible values are **HTTP-Redirect**, **HTTP-POST**, **HTTP-Artifact**, **HTTP-POST-SimpleSign**, or **SOAP**.
Default Value	**HTTP-Redirect**
Property	Service authentication login request
Attribute	`authentication.timeout.callback`
Description	The timeout (in milliseconds) for the user between initiated login and callback. The default value is five minutes.
Default Value	**300000**
Property	Authentication: Token
Attribute	`authentication.token.cookie`
Description	Used when sticky load balancer is using cookies.
Default Value	**stoken**
Property	Authentication: Token
Attribute	`authentication.token.domain`
Description	The domain in which the token cookie should be registered under.
Default Value	
Property	Authentication: Token
Attribute	`authentication.token.persistence`
Description	This property is used to determine if the token should persist if the browser is closed or if it should only last while the browser is open. There are two possible values: **PERSISTENT** and **SESSION**. **PERSISTENT** will persist the token in the browser even if the browser has been closed and reopened. **SESSION** will remove the token from the browser if it is shutdown. **IMPORTANT:** After modifying the property value to SESSION, ensure to clear the `AppData/Token` folder before starting the server.
Default Value	**PERSISTENT**
Property	Authentication: Token
Attribute	`authentication.token.refreshable`
Description	This property determines if the token can refresh itself. The web client can identify if the token is about to expire and then request a new token with the existing token. A token is refreshable if the property is set to true. The token will expire and invalidate the user session if the property is set to false.

Default Value	**true**
Property	Authentication: Token
Attribute	`authentication.token.secret`
Description	The secret is used to sign the token. The secret will be auto-generated when the server starts for the first time. **NOTE:** This value should be kept a secret.
Default Value	**Auto-generated**
Property	Authentication: Token
Attribute	`authentication.token.validity.seconds`
Description	The number of seconds that the token should be valid.
Default Value	**604800**
Property	Authentication
Attribute	`authentication.type`
Description	The type of the authentication mechanism that will be used on Panopticon Streams.
Default Value	**BASIC**
Property	Cache
Attribute	`cache.plugin.ID`
Description	The ID of the cache plugin that will be used. Possible value: BinaryTableFile-Cache
Default Value	**BinaryTableFile-Cache**
Property	Cache
Attribute	`cache.purge.condition`
Description	The condition for determining when the cache should be purged or cleared. Possible values: **NONE, MEMORY**.
Default Value	**MEMORY**
Property	Cache
Attribute	`cache.purge.condition.memory.threshold`
Description	The memory threshold used to determine if the cache should be purged or not. The values are presented in percent, 0-100. 80 means that the cache will be purged if the memory consumption reaches 80 % or more.
Default Value	**80**
Property	Cache
Attribute	`cache.purge.enabled`
Description	Enable or disable the purge functionality. Possible values: true, false
Default Value	**true**
Property	Cache
Attribute	`cache.schedule.clear.enabled`

Description	Enable the cache clearing schedule. This is scheduling the clear cache operation which will remove all the expired cache entries.
Default Value	**true**
Property	Cache
Attribute	`cache.service.enabled`
Description	Enables and disable the service cache
Default Value	**true**
Property	Cache
Attribute	`cache.service.type`
Description	The service cache mechanism being used.
Default Value	**IN_MEMORY**
Property	CEP: Application
Attribute	`cep.application.autostart`
Description	Determines whether all of the stored applications in the Streams server should auto start when the Streams server starts.
Default Value	**false**
Property	CEP: Application
Attribute	`cep.kafka.application.state.path`
Description	Where the tmp folder of Panopticon Streams data are created.
Default Value	**C:/PanopticonStreams/Data/tmp/kafka-streams**
Property	CEP: Kafka
Attribute	`cep.kafka.connection.timeout`
Description	The connection timeout towards Kafka. The value is presented in milliseconds.
Default Value	**10000**
Property	CEP: Kafka
Attribute	`cep.kafka.input.retention.ms`
Description	Specifies the retention period of input streams.
Default Value	**60000**
Property	CEP Kafka
Attribute	`cep.kafka.monitoring.consumer.interceptor`
Description	Names of classes that will be used to monitor data consumed from topics in a Streams application. In addition, these are hooks that will allow an external application to inspect this traffic. **NOTE:** The default value enables the Confluent Control Center to show metrics for a Streams application.
Default Value	**io.confluent.monitoring.clients.interceptor.MonitoringConsumerInterceptor**

Property	CEP Kafka
Attribute	`cep.kafka.monitoring.producer.interceptor`
Description	Names of classes that will be used to monitor data produced to topics in a Streams application. In addition, these are hooks that will allow an external application to inspect this traffic. **NOTE:** The default value enables the Confluent Control Center to show metrics for a Streams application.
Default Value	**io.confluent.monitoring.clients.interceptor.MonitoringProducerInterceptor**
Property	CEP: Kafka
Attribute	`cep.kafka.output.retention.ms`
Description	Specifies the retention period of output streams.
Default Value	**900000**
Property	CEP: Kafka
Attribute	`cep.kafka.properties`
Description	The user-defined file that contains the properties for controlling the Kafka configurations in Panopticon Streams.
Default Value	**kafka.properties**
Property	CEP: Kafka
Attribute	`cep.kafka.schemaregistry.url`
Description	The URL to the Schema Registry.
Default Value	**http://localhost:8081**
Property	CEP: Kafka
Attribute	`cep.kafka.servers`
Description	The URL to all the Kafka servers.
Default Value	localhost:9092
Property	CEP: Kafka
Attribute	`cep.kafka.session.timeout`
Description	The timeout for the Kafka session. The value is presented in milliseconds.
Default Value	**15000**
Property	CEP: Kafka
Attribute	`cep.kafka.table.retention.ms`
Description	Specifies the retention period for tables.
Default Value	**86400000**
Property	CEP: Kafka
Attribute	`cep.kafka.topic.partitions`
Description	Propagates the server-wide default for topic partitions.

	NOTES: • The *Partition Count* priority is applied in the following sequence (top to bottom): 1. Topic level 2. Application level 3. Property level • The event processor create topic gets the partition count for that topic. If the topic exists, it checks for an existing partition count and deletes the topic if it has a different value, and then creates it with the provided partition count. • Kafka server with auto topic creation on connect will cause issues due to preemptive metric collector component. This can be fixed with Kafka-client version 2.3 onward upgrade and adding "`allow.auto.create.topics=false`" in KafkaConsumer properties for TopicMetricsThread: ○ PreviewSubscriptionThread ○ TopicInputSchemaRepository
Default Value	**1**
Property	CEP: Kafka
Attribute	`cep.kafka.watcher.wait`
Description	The interval (in milliseconds) at which Streams will check the status of the ZooKeeper, Kafka Broker, and Schema Registry services.
Default Value	**5000**
Property	CEP: Kafka
Attribute	`cep.kafka.zookeeper.servers`
Description	The URL to the ZooKeeper servers.
Default Value	**localhost:2181**
Property	CEP: Kafka
Attribute	`cep.type`
Description	The CEP type. For now, the available value is **KAFKA**.
Default Value	**KAFKA**
Property	Server Cluster
Attribute	`cluster.bully.bind`
Description	The URL of the server in bully mode. This should be the URL to the Panopticon server web application on the server itself, by which is reachable from the other servers.
Default Value	
Property	Server Cluster
Attribute	`cluster.bully.boot`
Description	Comma-separated list of server URLs in bully mode. At least one of these servers should be running at all time for the bully mode to work correctly. The URLs should be the same as the cluster.bully.bind value on each boot server.

Default Value	
Property	Server Cluster
Attribute	`cluster.bully.id`
Description	The unique server ID in bully mode. Can be any string, but do not change it after the server has participated in a cluster -- the other servers will store it and expect it to identify the same server in the future. The running server with the lowest ID lexicographically will be leader.
Default Value	
Property	Server Cluster
Attribute	`cluster.fixed.leader`
Description	The leader URL in fixed mode. This should be the URL to the Panopticon server web application on the preset leader server, by which it is reachable from the follower servers. Leave blank on the leader server itself.
Default Value	
Property	Server Cluster
Attribute	`cluster.kubernetes.id`
Description	Set to the name of the pod that runs the container.
Default Value	(blank)
Property	Server Cluster
Attribute	`cluster.kubernetes.label_selector`
Description	Standard Kubernetes label selector that should only match the pods that are running the server.
Default Value	
Property	Server Cluster
Attribute	`cluster.kubernetes.peer_path`
Description	Path to the web application on each server. For example, "panopticon/", or "/" if you have deployed to Tomcat's root.
Default Value	
Property	Server Cluster
Attribute	`cluster.mode`
Description	**NONE** (default), **FIXED, BULLY,** or **KUBERNETES** Controls how multiple servers connect to each other. This needs to be the same on all connected servers.
Default Value	**NONE**
Property	Host Lookup
Attribute	`connector.kdb.host.lookup.script`

Description	Full path of the shell script file that is accessible on the server. When set, before making a new kdb+ connection, this script is executed to get the host info. This property helps in overriding connection details entered inside the kdb+ connector UI centrally, and may help when different authentications are set at kdb+ like Kerberos/Custom etc. The output of this script is expected to be a JSON object like below. `{ "host": "localhost", "port": 5001, "username": "", "password": "" }`
Default Value	
Property	Host Lookup
Attribute	`connector.kdb.host.lookup.script.arguments`
Description	Delimited set of arguments to be passed to the script when it is executed. '`{host},{port},{userid},{password}`' is the default value, and these parameters are mapped to respective settings in the connector UI i.e., the value entered against these settings in the connector UI are passed as arguments to the script. This property can be extended or updated if you want to pass other datatable parameters as arguments. System parameter like `{_user_id}` or `{_workbook_folder}`, if added to the data table, can also be used. If the value of some parameter is null or empty at the time of execution of the script, two single quotes are passed (‘’) against that parameter, this is to make sure that arguments count matches the arguments set at this property.
Default Value	**{host},{port},{userid},{password}**
Property	Host Lookup
Attribute	`connector.kdb.host.lookup.script.arguments.delimiter`
Description	Used to split the arguments set at above property.
Default Value	**,**
Property	Host Lookup
Attribute	`connector.kdb.host.lookup.script.timeout`
Description	The timeout (in milliseconds) to wait for the host lookup script to run and return the host info.
Default Value	**5000**
Property	Amazon Kinesis – Data Streams connector
Attribute	`connector.kinesis.datastreams.accesskeyid`
Description	The Access Key ID from the AWS account.
Default Value	
Property	Amazon Kinesis – Data Streams connector
Attribute	`connector.kinesis.datastreams.secretaccesskey`
Description	The Secret Access Key ID from the AWS account.
Default Value	

Property	Python connector
Attribute	`connector.python.host`
Description	The default Python Pyro instance host address. **NOTES:** For `connector.python.host`, `connector.python.password`, `connector.python.port`, and `connector.python.serializertype` properties: • If set in the `Streams.properties` file, these fields will be hidden in the Python connector and will be applied to the Python transform as well. • These default Streams Server connection properties will be applied at runtime. • These default Streams Server connection properties will override old Python connection settings.
Default Value	
Property	Python connector
Attribute	`connector.python.password`
Description	The default HMAC Key.
Default Value	
Property	Python connector
Attribute	`connector.python.port`
Description	The default Python Pyro host port.
Default Value	
Property	Python connector
Attribute	`connector.python.serializertype`
Description	The default Python serialization type. Possible values are **serpent** or **pickle**.
Default Value	
Property	REST Documentation
Attribute	`documentation.enabled`
Description	Enable or disable the OpenAPI Specification documentation for the REST interface.
Default Value	**false**
Property	REST
Attribute	`error.default.message`
Description	The error message that will be displayed instead of the actual error message. This is used to mask or hide error messages that may contain internal or sensitive details.
Default Value	
Property	File Upload
Attribute	`file.upload.size.max.bytes`

Description	Limit for files size to be uploaded through the web browser (i.e., workbooks, streams applications, streams data sources).
Default Value	**30000000**
Property	Log level
Attribute	`logger.level.file`
Description	Controls the level that is logged to file.
Default Value	**WARNING**
Property	Server Metrics
Attribute	`metrics.authorization.level`
Description	Specifies the required authorization level to get server metrics. Available values are **ANONYMOUS**, **VIEWER**, **DESIGNER**, **ADMINISTRATOR**. **NOTE:** This property is case sensitive.
Default Value	**ADMINISTRATOR**
Property	Server Metrics
Attribute	`metrics.collection.rate`
Description	Specifies the rate at which metrics are collected in milliseconds.
Default Value	**1000**
Property	Server Metrics
Attribute	`metrics.file.flush.rate`
Description	Specifies how often metrics should be saved to disk in milliseconds. Only used if the `metrics.publisher.type` is set to **FILE**.
Default Value	**10000**
Property	Server Metrics
Attribute	`metrics.memory.queue.size`
Description	Specifies how many metric entries are stored in memory. When the number of metrics goes above the specifies value, the oldest value is removed to make room for the newest one (FIFO). Only used if the `metrics.publisher.type` is set to **MEMORY**.
Default Value	**100**
Property	Server Metrics
Attribute	`metrics.publisher.type`
Description	Specifies the current metric publisher that is used. Available values are **NONE, MEMORY, FILE, EMAIL, INFLUX_DB, JDBC, KAFKA, KDB, MQTT, REST, TEXT**.
Default Value	**MEMORY**
Property	Server Metrics
Attribute	`metrics.publisher.configuration`
Description	Specifies the id for which metric publisher configuration to use.

Default Value	
Property	Repository
Attribute	`repository.import.archived.applications`
Description	Allows to import all application backups from the `<appdata>/CEP/Archive/`. Refer to step 4 in the Migration to Streams Server 2021.0 from an Older Version section for more information.
Default Value	**true**
Property	Repository
Attribute	`repository.pack.enabled`
Description	The repository tracks all changes to all applications. If you have a very large number of applications, or have kept the repository for a very long time, the sheer number of files inside the `.streams-repository` subdirectory could cause the repository to become slower. Set this property to **true** to have the repository pack all the files into fewer larger ones for faster access.
Default Value	**false**
Property	Repository
Attribute	`repository.startup.filesystemcheck`
Description	If set to **true,** server runs on startup to verify the repository integrity and reports any of the following issues: • a deleted `/HEAD` file, • a modified `/HEAD`, • a modified `/refs/heads/master` file, • any file deleted inside `/objects/` (e.g., /objects/94/443eec118fb8bb2021071896ff7d386a9c9518), • any file modified inside `/objects/`. **NOTE:** There may be dangling files in the `/objects/` directory or those that are not in use. These files are typically results of failed saves and/or sync conflicts. The check may or may not detect deleted or modified dangling files, but that is not critical.
Default Value	**false**
Property	REST
Attribute	`rest.response.error.stacktrace.included`
Description	Include the error stackrace in REST responses.
Default Value	**false**
Property	Server Downgrade
Attribute	`server.force_downgrade`
Description	The server normally refuses to start if it detects that the `AppData` directory has been used by a server with a newer version. This is because downgrading content and other `AppData` files is not supported and can cause irreversable issues. You can set this property to true to force the server to start anyway, but it is strongly recommended that you do not.
Default Value	**false**

Property	Server
Attribute	`server.id`
Description	Specifies an id for the current server. The value of this property will be part of each metric entry so that it can be tied to a specific server if a server cluster is used. If no value is specified, the MAC address of the localhost network will be attempted to be used to identify the server. If this is not possible, a UUID will be generated.
Default Value	
Property	SOAP
Attribute	`soap.enabled`
Description	Enable or disable the SOAP interface
Default Value	**True**
Property	Licensing
Attribute	`license.hwu.hosted`
Description	Boolean stating if you wish to use Hosted or Local Altair Units licensing. Set to **true** if you wish to use hosted licensing.
Default Value	**false**
Property	Licensing
Attribute	`license.hwu.hosted.authorization.username`
Description	Username to the Altair One account.
Default Value	
Property	Licensing
Attribute	`license.hwu.hosted.authorization.password`
Description	Password to the Altair One account.
Default Value	
Property	Licensing
Attribute	`license.hwu.hosted.authorization.token`
Description	An authorization token generated through the Altair One admin portal. Used to authorize a machine to the Hosted Altair Units system.
Default Value	
Property	Licensing
Attribute	`license.hwu.operating.system`
Description	The operating system where Panopticon Streams is installed. Possible values are: **WIN_X86, WIN_X64, MAC, LINUX_X64,** or **LINUX_ARM64** **NOTE:** If the Java bitness (e.g., 32-bit) is different from the operating system (e.g., 64-bit), it is recommended to add the Java bitness in this property (e.g., **WIN_X86**).
Default Value	
Property	Licensing

Attribute	`license.hwu.uri`
Description	The path where the License Server is running e.g., **6200@191.255.255.0** where the syntax is `PORTNUMBER@HOST`. If multiple servers are specified, use the '**;**' semicolon separator sign for Windows and the '**:**' colon separator sign for Linux. **NOTE:** If value is not set in the `Streams.properties`, the environment variable **ALTAIR_LICENSE_PATH** serves as the backup path and will be used.

Example	For Windows:
	`license.hwu.uri=6200@192.168.5.51;6200@192.168.5.52`
	For Linux:
	`license.hwu.uri=6200@192.168.5.51:6200@192.168.5.52`

Default Value	
Property	Licensing
Attribute	`license.hwu.version`
Description	Value must match the license version found in the Altair Units license file.
Default Value	**19.0**
Property	Licensing
Attribute	`license.mode`
Description	The license mode. Possible values are: **FILE** or **HWU**. To use the Altair Units license, set this property to **HWU**.
Default Value	**FILE**
Property	Timeout Session
Attribute	`timeout.session.enabled`
Description	Boolean value stating if timeout functionality should be used or not.
Default Value	**false**
Property	Timeout Session
Attribute	`timeout.session.exception.delimiter`
Description	The delimiter to use for the usernames stated in the timeout.session.exception.usernames property.
Default Value	**, (comma)**
Property	Timeout Session
Attribute	`timeout.session.exception.usernames`
Description	Usernames that should be excluded from the timeout functionality. Separated by the delimiter stated in the `timeout.session.exception.delimiter` property.
Default Value	
Property	Timeout Session
Attribute	`timeout.session.minutes`

Description	Minutes of inactivity before a user session is terminated by logging out the user.
Default Value	**480**

07.2022

CONTACT US

GET IN TOUCH

We'd love to hear from you. Here's how you can reach us.

SALES CONTACT INFORMATION

US: + 1.800.445.3311
International: + 1.978.441.2200

Sales Email

US: sales@datawatch.com
Europe: sales_euro@datawatch.com
Asia Pacific: sales_apac@datawatch.com

SUPPORT CONTACT INFORMATION

Customer Portal: https://www.altair.com/data-analytics-support/
Email: **mailto:dasupport@altair.com**
US: +1 800.988.4739
Canada: +1 978.275.8350
Europe, Middle East, Africa: +44 (0) 8081 892481