insightsoftware

Hubble Accelerator

User Guide

Version 5.0

November 2025

Contents

Document Information

Notices

Copyright

Disclaimer

This guide is designed to help you to use the Hubble applications effectively and efficiently. All data shown in graphics are provided as examples only. The example companies and calculations herein are fictitious. No association with any real company or organization is intended or should be inferred.

Version History

Date	Revision	Software Version	Comments
17th November, 2025	1.0	5.0	Initial issue for 5.0.

Customer Support

For more information regarding our products, please contact us at https://insightsoftware.com/hubble/.

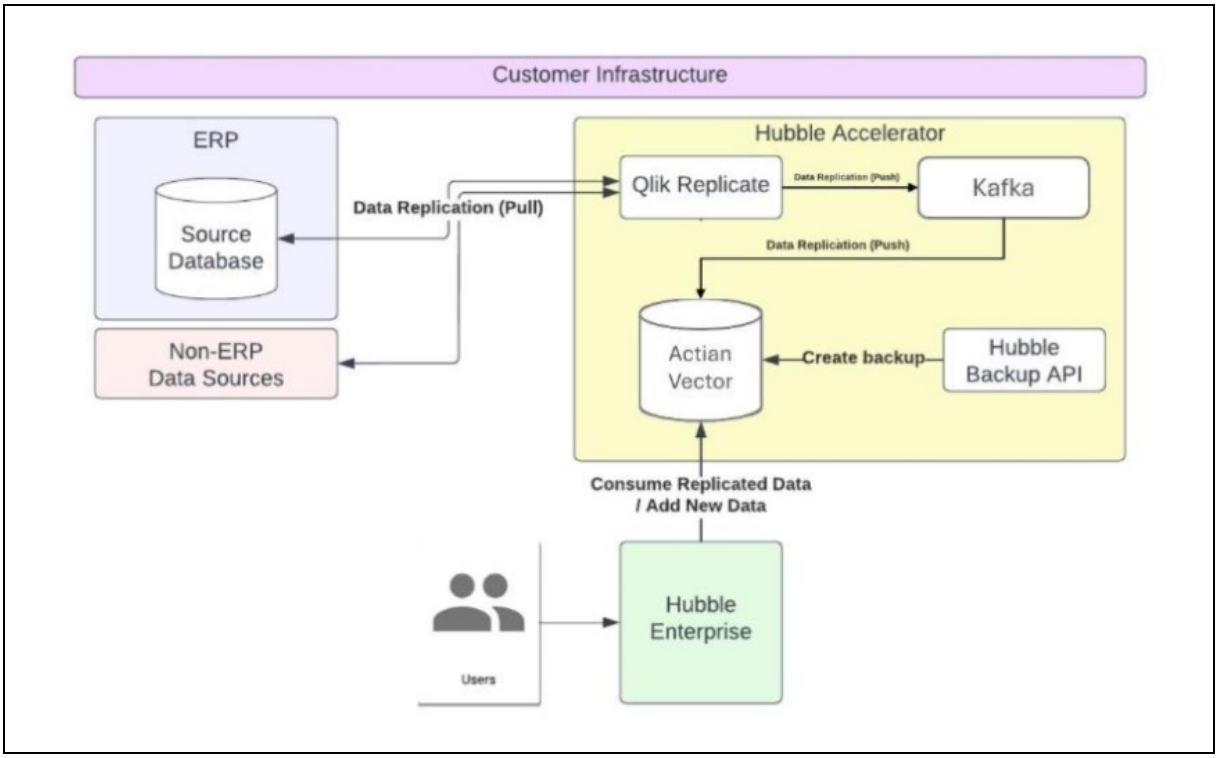
For upgrade questions, parallel version installations, product support, training, and documentation, contact Professional Services at http://central.insightsoftware.com/ or email HubbleServices@insightsoftware.com.

Architecture

Note: In version 5.0, a new enterprise connector that uses Kafka has been added to stream changes into the Actian Vector database. This enhancement ensures continued seamless data replication, complementing the existing Qlik connector.

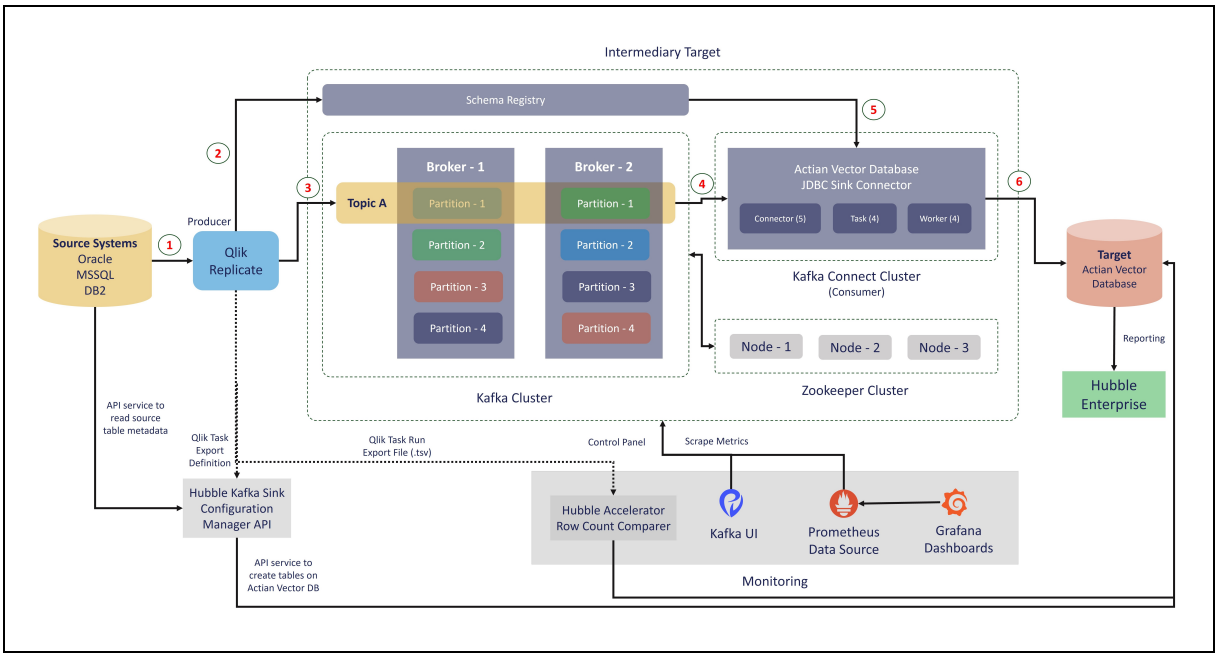
System Architecture

Here is a diagram with the standard System Architecture of a Hubble Enterprise Deployment.



Technical Architecture

Here is a diagram with the standard Technical Architecture of a Hubble Accelerator Deployment.



The Hubble Accelerator techinical architecture integrates various components to ensure efficient data flow from source systems to the target database, addressing challenges encountered during migration. Here are the key components:

- **Source Systems:** Oracle, SQL Server, and DB2 databases supported by the product.
- **Qlik Replicate:** Primary replication component that pulls data from source ERP systems and sends it to Kafka.
- **Kafka Component:**
  - **Partitions:** Kafka stores data in partitions, managed by brokers.
  - **Brokers:** Multiple brokers ensure high availability; if one broker fails, another takes over.
  - **Topics:** Similar to table rows in RDBMS, topics organize data within partitions.
- **JDBC Sink Connector:** Key engine responsible for reading data from Kafka partitions and pushing it to the target Actian Vector database.
- **Schema Registry:** Holds schema information for messages, ensuring data consistency between source and target.
- **Zookeeper:** Control center for Kafka components, managing coordination and high availability.
- **Hubble Kafka Sink Configuration Manager API:**
  - Retrieves the DDL of the source table and constructs an equivalent DDL on the target database.
  - Runs the DDL on the target database, deleting the table if it already exists.
  - Monitors the current status of Kafka, indicating whether Kafka is online or if a restart is needed.

- **Monitoring:**
  - **Qlik Replicate UI:** Provides real-time data replication monitoring.
  - **Kafka UI:** Included in the software bundle, allowing users to monitor Kafka to an extent.
  - **Hubble Accelerator Row Count Comparer Utility:** Utility for consultants and support engineers to view row counts on the source and target.
- Reads Qlik Replicate execution task details and row counts.
- Runs SELECT COUNT(*) queries on the target database and generates a CSV file with table name, source row count, and target row count.

- **Prometheus Data Source and Grafana Dashboards:**
  - Optional components for monitoring real-time data replication and identifying performance issues.
  - Not included in the default package to avoid additional resource consumption.
  - Can be installed by consultants if performance issues are observed.

Components

Hubble Accelerator installs:

- **Actian Vector:** SQL RDBMS designed for high performance in analytical database applications.
- **Qlik Replicate (formerly Attunity Replicate):** Data replication service.
- **Confluent Kafka:** Kafka is a distributed streaming system used for real-time data pipelines and data integration at scale.
- **Kafka UI:** Tool that makes your data flows observable, helps find and troubleshoot issues faster, and delivers optimal performance. The lightweight dashboard makes it easy to track key metrics of your Kafka clusters, including Brokers, Topics, Partitions, Production, and Consumption.
- **Hubble Kafka Sink Configuration Manager API**
- **Hubble Accelerator Row Count Comparer:** This utility compares row counts between a source file (.tsv) and a target database.
- **Hubble Backup API:** Facilitates backups for the Actian Vector data in case of migrations and upgrades.

Network Access

Here are the ports that need to be configured in the firewall to ensure the normal functioning of Hubble Accelerator within a Hubble deployment.

Port	Protocol	Service	Software	Sources
3552	TCP	Replication Admin UI	Attunity Replicate	Replication Admin PC
27832	TCP	Net server IIGCC (Communication)	Actian Vector	Hubble desktop users on company network, web server
27839	TCP	Data Access Server	Actian Vector	Hubble desktop users on company network, web server
3550	TCP	Replication	Attunity Replicate	Attunity front end running on the accelerator
5000	TCP	Hubble Backup API	hubble-applianceapi	Admin tool backup budgeting data
50001	TCP	Hubble Alert Services	hubble-alerts	Application server, Web Servers
9092 - 9094	TCP	Kafka Brokers	Kafka	Qlik Replicate, Kafka Consumer
2181-2183	TCP	Zookeeper Client	Zookeeper	Kafka Brokers
8081	TCP	Schema Registry API	Confluent Schema Registry	Kafka Producer/Consumer
8083	TCP	Kafka Connect REST API	Kafka Connect	Kafka Consumer
5002	TCP	Hubble Kafka Sink Configuration Manager API	hubble-kafka-sink-api	Admin tool for table creation on Actian Vector, post connector config files to Kafka Connect
8080	TCP	Kafka UI	Kafka UI	Admin, Monitoring Tools

Recommended Hardware Configuration

To ensure optimal performance and scalability of Hubble Accelerator, the following recommended hardware configurations are provided:

- Scalability Guidelines Based On Customer Data
- Hubble Accelerator Host Server Hardware
- Hubble Accelerator as a Virtual Machine

Scalability Guidelines Based On Customer Data

The following table provides guidelines for expected user and data densities based on testing and production results from various customers. Results may vary due to numerous factors, and conducting your own testing will help determine your specific scalability.

	Performance Variables	Minimum	Recommended	Advanced
1	Data volumes in Vector – Low (~20GB)	X	X	
2	Data volumes in Vector – Medium (~150GB)		X	
3	Data volume in Vector– High (~500GB)			X
4	Data complexity [1] – Low (<100k transactions per day)	X	X	
5	Data complexity [1] – Medium (100k - <1M transactions per day)		X	
6	Data complexity [1] – High (>1M transactions per day)			X
7	Anticipated User Count [2]	20-50	30-100	100-1000+

[1] Complexity in this context is volume and frequency of change via Change Data Capture (CDC).

[2] Precise user counts will vary subject to the complexity of application usage, workspaces, reports, etc.

Tip: We recommend early discussions with the Hubble Support on appropriate hardware configuration for significantly higher user counts (and/or multiple data sources).

Analysis Summary

The combination of Data Volume and Data Complexity require significant CPU resources, so investing in higher CPU speeds and more cores will pay off in both performance and scalability. In the table above, the combination of #1 and #5 may require use of the Advanced specifications as shown in the last column.

Hubble Accelerator Host Server Hardware

Note: The following hardware requirements are based on internal testing. Actual performance may vary depending on system configuration, and adjustments may be necessary accordingly

Aspect	Minimum	Recommended	Advanced [1]
CPU (example)	DualIntel® Xeon® Gold 6426Y Processor [32 cores total]	Dual Intel® Xeon® Gold 6448H Processor [48 cores total]	Dual Intel® Xeon® Platinum 8461V Processor [64 cores total]
Memory	256 GB	512 GB	512-720 GB
Networking [2]	1 Gbps	10 Gbps	10 Gbps
Storage [3]	Sufficient for hypervisor plus hosted systems. High speed (1+ GB/sec read) SSDs[4]recommended. Additional Disk Space: 500+ GB (varies based on the data volume)		

[1] Use of dual 4th Generation Intel® Xeon® Scalable Processors may be appropriate in some cases (beyond 250 concurrent users with multiple complex data sources, for example). Please discuss exact requirements with Hubble Support at early stages of your planning

[2] 1 GB/sec equates to read IOPS of 266,000 or greater.

[3]The primary storage for the Vector database is disk storage. Your storage solution must satisfy performance and space requirements. To achieve good consistent performance, you should choose smaller rather than bigger disks.
For example, choose 146 GB disks over 500 GB (or larger) disks. Faster spinning disks at 15k RPM have higher throughput rates than slower 10k RPM or 7.2k RPM disks. In an ideal case, a single spinning 15k RPM disk can sustain up to 150 MB/s data transfer.

[4]Vector is optimized to work with both memory and disk-resident datasets, allowing it to efficiently process large amounts of data (hundreds of gigabytes). Vector can process data at more than 1.5 GB/sec per CPU core. To achieve this rate, the CPU cores must be fed at a rate fast enough to keep them busy. Consider making SSD technology a viable storage consideration for your system. It is recommended to choose SSDs for temporary database storage to improve performance for spill-to-disk operations i.e. Vector work area (II_WORK).

Hubble Accelerator Virtual Machine

Note: For optimal performance, it is recommended to run the Hubble

Accelerator
in a Virtual Machine on dedicated hardware. It should not be hosted on the same physical device as other Virtual Machines or business applications that could impact system resources.

Aspect	Minimum	Recommended	Advanced
Memory [1]	256 GB	512 GB+	512-720 GB
vCPUs	48 cores	64 cores	96 cores
Disk Space [2]	Data [2] - 500 GB	Data [2] - 1 TB	Data [2] - 1-2 TB
Additional Disk Space	500+ GB (varies based on the data volume)		

[1]This is the amount of RAM available to the applications and is based on a single replication data source. Additional RAM may be required for particularly complex or big data environments. RAM size may have to be increased post-implementation.

[2] Storage requirements will vary considerably by customer, depending upon the quantity of source ERP data and how much needs to be replicated to the Accelerator to service your reporting and analytics needs. For guidance, the typical compression ratio between source data in the ERP database and the accelerator is 4:3.

Recommended Software Configuration

This section describes how to prepare your system for Hubble Accelerator and how to install it.

Installation Prerequisites

This section describes how to prepare your system to use Hubble Accelerator. The requirements differ according to the platform on which you want to install Hubble Accelerator.

Software Requirements

Hubble Accelerator is distributed in RPM format and can be installed on the following Linux platforms:

- Red Hat Enterprise Linux (64-bit) 7.x
- CentOS 7.x

Dependencies

All dependencies and its dependencies are required. If an offline installation is required, please ensure all dependencies are installed before installing Hubble Accelerator rpm.

- libaio
- perl
- libX11
- libXext
- libXi
- libXrender
- libXtst
- alsa-lib
- perl
- perl-Data-Dumper
- unixODBC-utf16 (For version 4.1.x or higher - shipped with the RPM package)
- initscripts
- atk
- cairo
- gdk-pixbuf2
- gtk2
- pango
- sudo
- which

- mono-complete-6.12.0.107-0.xamarin.9.epel8 (for version 4.2.x or higher)
- bc
- e2fsprogs
- openssl

To install **mono-core-6.12.0.107-0.xamarin.9.epel7.x86_64** on CentOS/RHEL 8:

- Refer to the detailed instructions for installing Mono on CentOS/RHEL 8 available at https://www.mono-project.com/download/stable/#download-lin-centos
- For additional information, you can check the release notes for Mono 6.12.0 at https://www.mono-project.com/docs/about-mono/releases/6.12.0/.

If some dependencies for **mono-complete-6.12.0.107-0.xamarin.9.epel7** have been moved from the **centos8-stable** repository, you can find them at this URL. You can manually download and install them using the following commands:

- Download the package:

```
wget <url>
```

- Install the package:

```
yum localinstall <package.rpm>
```

To uninstall **unixODBC-utf16** for an older version of Accelerator (for example, Version 3.x), you can use the following command:

```
yum remove -y unixODBC-utf16
```

To uninstall **unixODBC** for version 4.2.x or higher, you can use the following command:

```
yum remove -y unixODBC
```

To uninstall **mono** for version 4.2.x or higher, you can use the following command:

```
As root user

# yum list mono-complete --showduplicate

Remove older versions
```

```
# yum remove -y mono

# cd /etc/yum.repos.d/
# ls -la centos7-stable.repo
# rm centos7-stable.repo

Download centos8-stable repo
# wget https://download.mono-project.com/repo/centos8-stable.re
# yum install mono-complete
```

Software Components Used in Hubble Accelerator

The following table lists the software components required for the Hubble Accelerator environment, including their versions, update history, and relevant notes for deployment and compatibility.

	Version
Hubble Accelerator	5.0.0
Actian Vector	6.3.0-146
Patch	14621
Attunity (Qlik)	2025.5.0-247
Oracle ODBC	12.2-basiclite-12.2.0.1.0-1
MS ODBC	18.1.2.1-1
IBM i Access ODBC	1.1.0.27-1.0
Hubble Appliance API	3.0.0-3
Kafka	7.8

Supported Replication Source Endpoints

The following table outlines the supported source databases and their configurations for use with the Hubble Accelerator. The target database is always Actian Vector.

Source Database	Database Version	Application Version
Oracle DB for EBS R12	12c, 19c	EBS 12.2.6
Oracle DB for JDE	19c	
SQL Server for JDE	2012, 2022	JDE 9.0

Oracle

For supported Oracle DB versions and configurations, refer to Qlik Replicate's documentation.

Oracle Instant Client version 12.1.0.2.0-1.x86_64 will be installed with Hubble Accelerator installation.

Microsoft SQL Server

For supported Microsoft SQL Server DB versions and configurations, refer to Qlik Replicate's documentation.

Microsoft ODBC Driver 18.1 for Windows will be installed on the Qlik Replicate Server machine with Hubble Accelerator installation (for version 4.2.x or higher).

Important: Installing MSODBC 18.1 will uninstall previous versions such as 13.1 and 17.1.

TLS 1.2 prerequisites

If your environment includes:

- Microsoft SQL Server 2014
- Microsoft ODBC Driver 18 installed on Red Hat 8.1 or later

To enable TLS 1.2, you must install a specific Service Pack on the Windows machine where Microsoft SQL Server is installed. For more details, visit https://support.microsoft.com/en-us/help/3135244/tls-1-2-support-for-microsoftsql-server.

Recommended Host Configuration

Provide a separate disk to hold the Vector database and Qlik replication Data. For example, `/etc/fstab`:

```
LABEL=hubble    /opt    auto    defaults    00
```

Users and Groups

Hubble Accelerator installer will create users and groups in Linux.

Users:

- **attunity:** Owner of all Attunity related files and directories. This user runs Attunity Replicate on the Accelerator.
- **actian:** Owner of all Actian related files and directories. This user runs Actian Vector and its related executable on the Accelerator.
- **hubble-applianceapi:** Owner of Hubble Appliance API related files and directories.

Groups:

- **actian**
  - members: actian
- **attunity**
  - members: attunity

- **hubble**
  - members: hubble-applianceapi

Main Components of Hubble Accelerator

This section describes the procedures used when working with the following main components of Accelerator.

- "Actian Vector" below
- Qlik (Formerly Attunity) Replicate
- Confluent Kafka
- Kafka Sink Configuration Manager API
- Hubble Accelerator Row Count Comparer
- Replication Monitoring using Kafka -UI
- "Hubble Backup API" on page 49

Actian Vector

This section is divided into the following key areas for Actian Vector:

- Installation Path
- Basic Commands

◦ Startup Vector

◦ Shutdown Vector

Installation Path

Actian Vector will be installed by default on:

```
/opt/Actian
```

Basic Commands

- Start up Vector:

```
systemctl start actian-vectorVW
```

- Shut down Vector:

```
systemctl stop actian-vectorVW
```

Qlik (Formerly Attunity) Replicate

- Qlik (Formerly Attunity) Replicate Installation Path
- Qlik (Formerly Attunity) Replicate Basic Commands
- Qlik (Formerly Attunity) Replicate Maintenance Guide

Qlik (Formerly Attunity) Replicate Installation Path

On the Hubble Accelerator server, Qlik Replicate is installed under the following directory:

```
/opt/attunity
```

The replication logs are created in the following directory:

```
/opt/attunity/replicate/data/logs
```

Qlik (Formerly Attunity) Replicate Basic Commands

- Start up Qlik Replicate:

```
/opt/attunity/replicate/bin/areplicate start
```

- Shut down Qlik Replicate:

```
/opt/attunity/replicate/bin/areplicate stop
```

If the above commands fail, try the following commands:

```
/opt/attunity/replicate/bin/repctl -d
/opt/attunity/replicate/data service stop port=3550 rest_
port=3552
```

```
/opt/attunity/replicate/bin/repctl -d
/opt/attunity/replicate/data service start port=3550 rest_
port=355
```

Note: If there are tasks running on the system and you are unsure if they are still active follow the Qlik Shutdown Replication procedure.

- **Accessing Qlik Replicate Console:**

  Open a web browser and enter the following URL.

  `https://{IP Address}:3552/attunityreplicate/`

  User: admin

  Password: As detailed in the **Installing Hubble Accelerator** section, the password is generated on install.

Qlik (Formerly Attunity) Replicate Maintenance Guide

Contents:

- Shut down Qlik Replication
- Importing Replication Tasks
- Exporting Replication Tasks
- Using add-on to Remove Invalid Characters during Replication

Shut down Qlik (Formerly Attunity) Replication

Propagating the transaction log to disk will fail if processes are making database changes. As such the replication service must be stopped before attempting to propagate the log.

Propagating Accelerator In-Memory Changes and Condensing the Log.

1. Connect to Hubble accelerator server.
2. Switch user to the attunity replication user.

   # su attunity
3. Get a list of all the running replication tasks.

   # ps -ef | grep repctl

```
[root@yb~]# ps -ef | grep repctl
attunity  2349     1  0 May24 ?        00:00:00 /opt/attunity/replicate/bin/repctl service start
attunity  2350  2349  0 May24 ?        00:43:51 /opt/attunity/replicate/bin/repctl service start
attunity  6660  2350  1 Nov20 ?        05:57:29 repctl reptasksrv PRODDTA 127.0.0.1:3552 2350
root     12585 12579  0 14:44 pts/0    00:00:00 grep repctl
attunity 22588  2350  0 Nov06 ?        00:18:24 repctl reptasksrv JDE812 127.0.0.1:3552 2350
attunity 23203  2350  0 Nov06 ?        01:44:48 repctl reptasksrv CUSTOM TABLES 127.0.0.1:3552 2350
```

The running replication tasks are the processes that start repctl reptasksrv <TASK_NAME>. In the example above there are 3 running tasks:

- PRODDTA
- JDE812
- CUSTOM_TABLES

4. For each task returned by the above, run the following command to get the status of the task and if it is running to stop the replication task. (You can also stop the tasks in the Qlik Console, but you should check the status as instructed below)

   # ./repctl connect\; gettaskstatus <TASK_NAME>\; disconnect # ./repctl connect\; stoptask <TASK_NAME>\; disconnect

   The stoptask command used above will signal the task to stop, if the task is currently showing latency, then the task will not stop immediately but will stop after processing the changes that are causing the latency. This will allow you to restart the task with the resume feature, this means you do not need to reload the data. If you run the stoptask command and manually kill the associated replication process, then you will not be able to use the resume command to restart the task, you will have to restart the task using the reload functionality in order to ensure data consistency. All the tasks must be stopped before moving on to Step 5 and 6.

5. Shutdown Qlik replication service. This will prevent users form connecting remotely using Qlik Console and starting new tasks.

   # ./S65areplicate stop

6. Confirm that the services has stopped by checking that there are no repctl processes running:

   # ps -ef | grep repctl

Import Replication Tasks

If you are importing a backup file, the backup file will also contain the connection details (encrypted). After the import process has been completed, the password used for each of the database connections is removed. The password will need to be re-entered in the manage database connection dialog for each connection.

1. Copy the backup file (json) onto the accelerator server in the Linux directory:

   /opt/attunity/replicate/data/imports

2. Log onto the accelerator server and change the owner of the backup file to the user “attunity”, if necessary:

   chown attunity:attunity <FileName.json>

3. Log in or change the accelerator user account that you are logged in as to the “attunity” user:

   su - attunity

4. To import the backup file (json) by entering the following:

   repctl importrepository json_file=<Name_of_Json_File>

For example, for a json file named Ora_Rep_F0911.json, run the following command:

repctl importrepository json_file=Ora_REP_F0911

The name of the file is case sensitive.

You do not need to enter the .json extension.

Export Replication Tasks

1. Log onto the Accelerator server and change directory to /opt/attunity/replicate/data/imports:

   cd /opt/attunity/replicate/data/imports

2. Change the accelerator user account to use the attunity user:

   su - attunity

3. To export an individual task, enter the following:

   repctl exportrepository task=<name_of_task>

   For example, if the task is called PRODDTA, enter (note the task name is case sensitive):

   repctl exportrepository task=PRODDTA

4. To export all tasks, enter the following:

   repctl exportrepository

   A json file will be created in the /opt/attunity/replicate/data/imports directory. This can now be copied from the Accelerator (SCP) and to your backup location.

```
[root@yb-banfi-1 bin]# ./repctl exportrepository
command exportrepository response:
{
        "message":      "Export succeeded, export location: /opt/attunity/replicate/data/imports"
}
[exportrepository command] Succeeded
```

Using add-on to Remove Invalid Characters during Replication

Summary

This article describes a solution dealing with characters in the ERP source tables that are considered invalid during the replication to Vector and cause the replication task to fail with an error similar to the following:

VWLOAD output: loading Error in file '/opt/attunity/replicate/data/tasks/badora/vectorwise/1/LOAD00000001.csv', record 1: Illegal character encountered in input Input record: '"@@?@"' processed 1 records, loaded 0 records, 1 errors

The specific case addressed is when text columns representing descriptions in the ERP system were containing characters that are not UTF-8. A user-defined function called fix_latin was added to the

transformation functions. This function accepts a string of characters with mixed encodings, for example it may contain predominantly UTF-8 characters but may also contain characters encoded with Windows-1252 or ISO 8859-1. It will create a UTF-8 output that is based on the following conversions:

0x00 - 0x7F ASCII - passed through unchanged

0x80 - 0x9F Converted to UTF8 using Windows-1252 (aka CP1252) mappings 0xA0 - 0xFF Converted to UTF8 using ISO/IEC 8859-1 (aka Latin-1) mappings

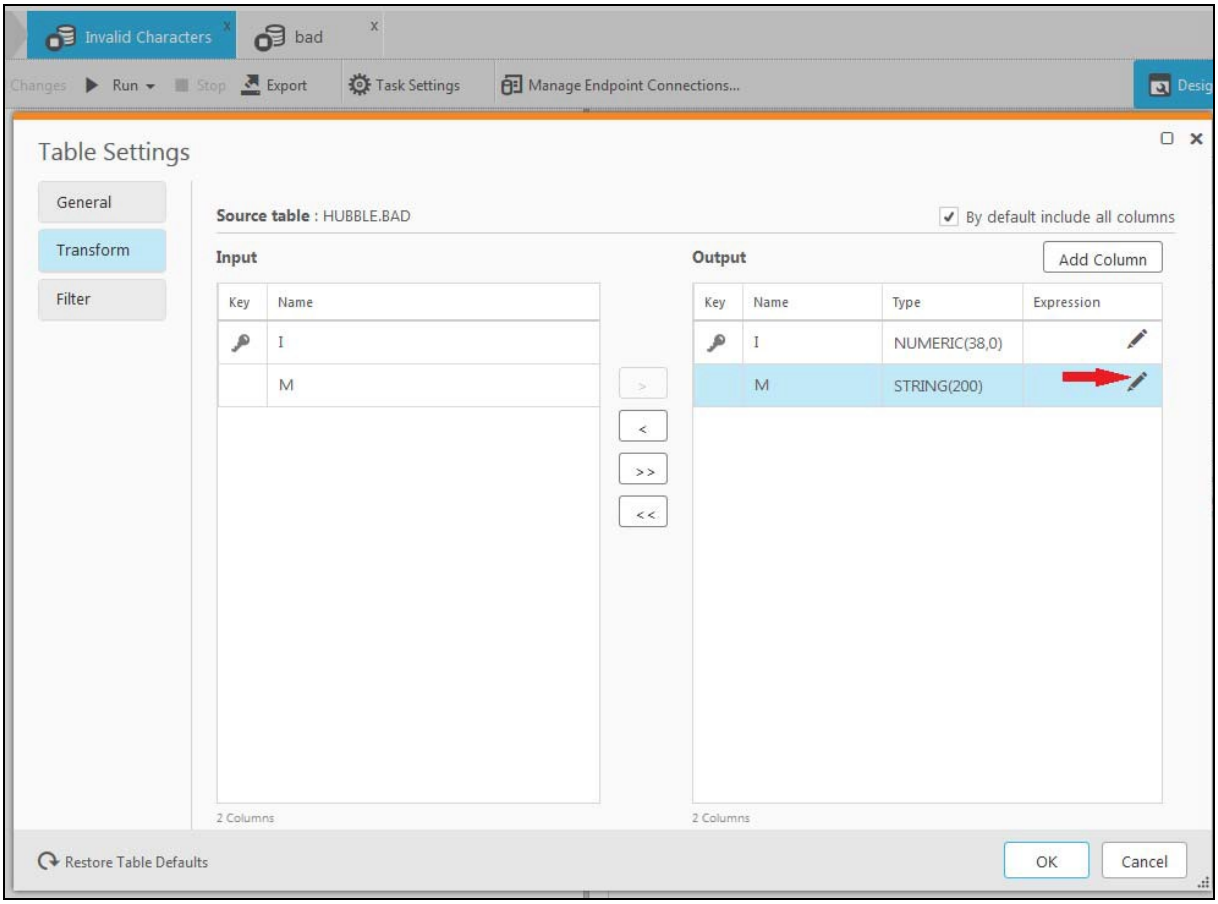
Five characters that do not have a mapping are converted to a string %xx with xx being the hexadecimal value, for example 0x81 is mapped to %81. Multi-byte UTF8 characters will be passed through unchanged, although over-long UTF8 byte sequences will be converted to the shortest normal form (see the original documents for the perl module).

Note that this solution has been designed and tested using Oracle as the source database. The fix_latin function is also available when other source database types are being used but it is untested and unsupported in that scenario.

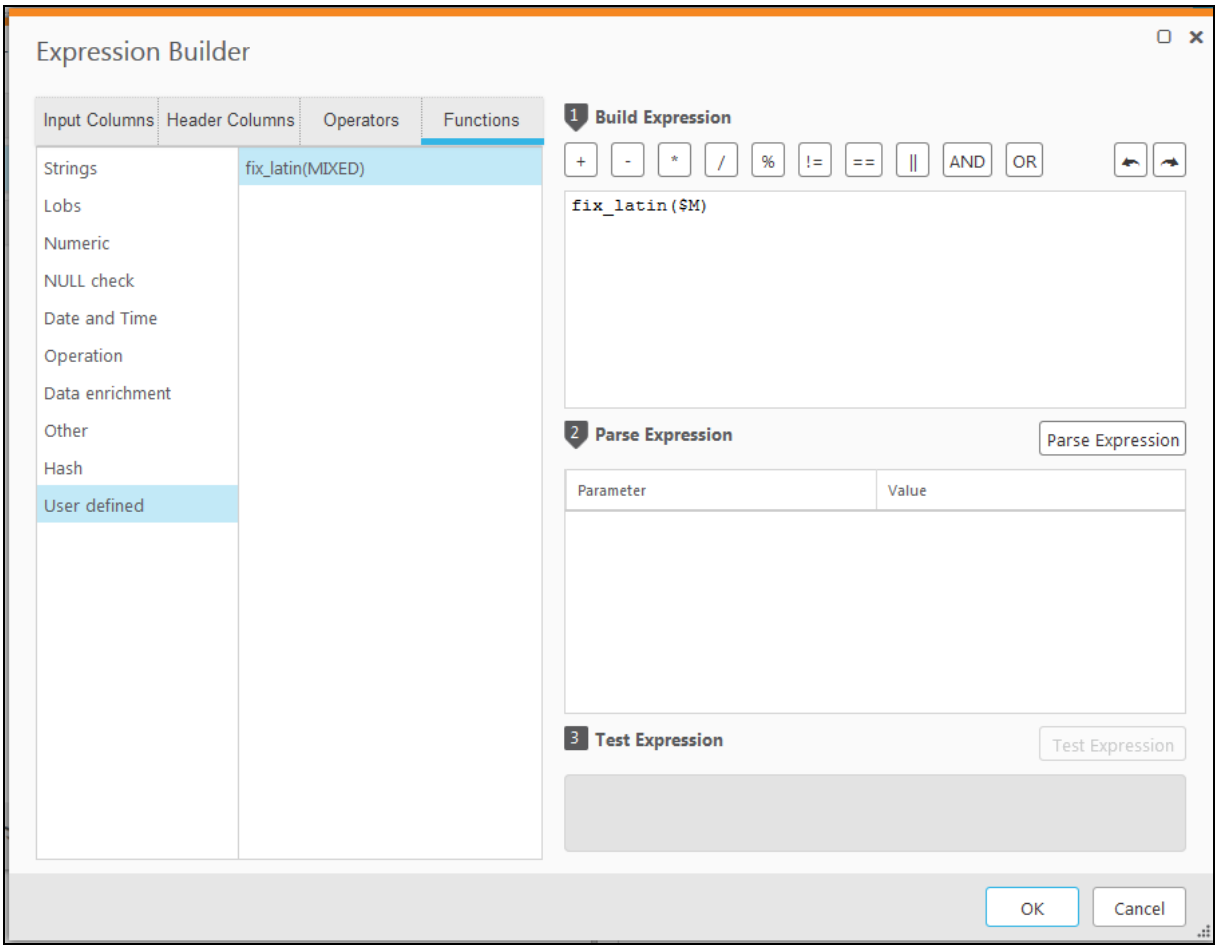
Manual Configuration Steps for Qlik Replicate

For any column that has invalid characters, a custom transformation can be set up in Qlik Replicate to replace these characters.

1. In the Table Settings, on the Transform tab, select the pen symbol to edit the transformation expression for the column in question:



2. Use the fix_latin function available on the Functions tab under User defined. Reference the source column that needs to be transformed as input to the function (M in the example below).



3. Repeat above steps for all text columns that have the same issue.

Confluent Kafka

Overview

Here's an overview of the data transfer process from Qlik to Actian Vector using Kafka:

- **Data Origin:** Data originates in Qlik and is sent to Kafka topics.
- **Intermediary Role:** Kafka acts as an intermediary to facilitate the transfer of data between Qlik and Actian Vector.
- **Kafka Server:** The Kafka server serves as the backbone of this setup, managing topics.
- **Kafka Connect:** A tool within the Kafka ecosystem, used to integrate Kafka with external systems.

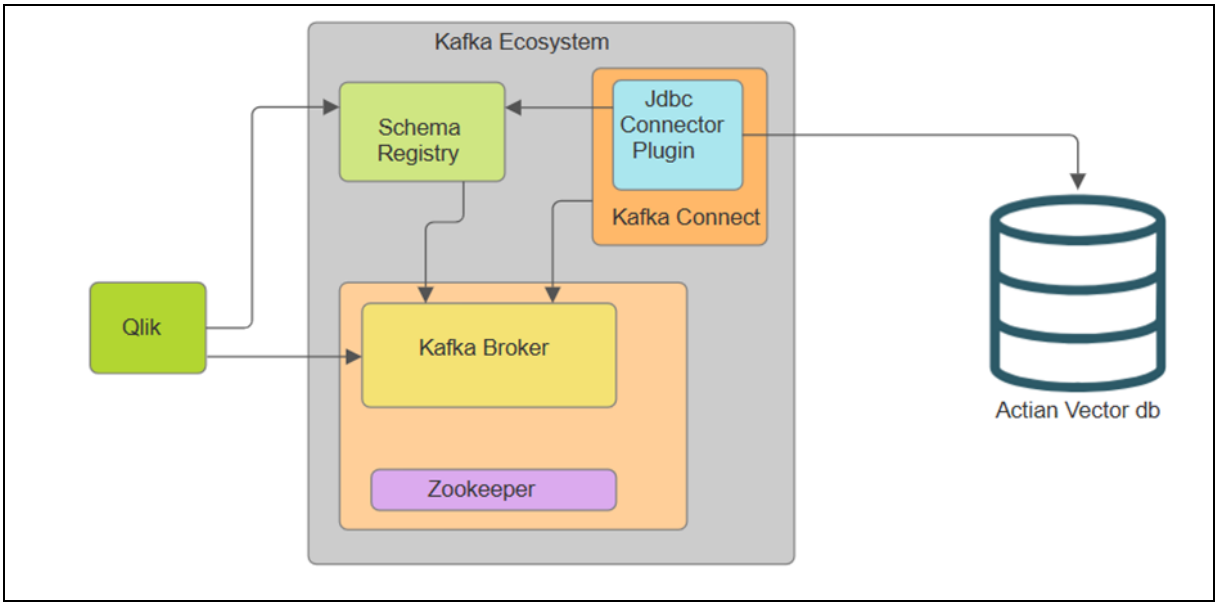
- **JDBC Sink Connector:** Connects Kafka topics with Actian Vector, translating topic data into database records.

  - **Actian Vector:** The target database that stores data from Kafka topics.
  - **Data Insertion:** The JDBC Sink Connector ensures proper data insertion into Vector tables.
  - **Schema Registry:** If in use, it manages data schemas across Kafka topics to ensure compatibility between data producers (Qlik) and consumers (Actian Vector) via the JDBC connector.

This structured approach ensures efficient and accurate data transfer and storage.

Architecture

The main components for this approach are Zookeeper, Kafka Brokers, Schema Registry, Kafka Connect, and Connector Plugins.



- **Qlik:** Serves as the source for the Kafka ecosystem components, where tasks are configured to push data into the intermediary Kafka component.
- **Zookeeper:** Manages multiple Kafka Broker nodes within the Kafka Cluster.
- **Kafka Broker:** Stores messages into topics and provides the means to store data in different partitions.
- **Schema Registry:** Provides REST API endpoints to post/get schema details and uses Kafka Broker to store schema details into specific schema topics.
- **Kafka Connect:** Connects Kafka to different target systems and provides REST API endpoints to get/post connectors.

- **Connectors:** Plugins added to Kafka Connect Worker that create tasks and handle the insertion of data into target systems.

Starting Kafka Cluster

To start the Kafka Cluster, follow these steps:

1. Zookeeper

Zookeeper maintains different Kafka Broker nodes and helps ensure high availability of the data.

a. Change the configurations as required in the `/etc/kafka/zookeeper.properties` file.

b. Start Zookeeper by running:

```
sudo systemctl start confluent-zookeeper
```

2. Kafka Broker

Kafka Broker is the Kafka server that stores all the topics and handles messages.

a. Change the configurations as required in the `/etc/kafka/server.properties` file.

b. Start Kafka Broker by running:

```
sudo systemctl start confluent-kafka
```

3. Schema Registry

Schema Registry is a component in the Kafka ecosystem that maintains message schemas. It provides REST API endpoints, allowing Kafka producers (Qlik) and Kafka Connect (consumers) to access the message schemas.

a. Configure the schema-registry properties in the `/etc/schema-registry/schema-registry.properties` file.

b. Start the schema-registry by running:

```
sudo systemctl start confluent-schema-registry;
```

4. Kafka Connect

Kafka Connect is a tool used to connect Kafka to target systems. It requires connectors as plugins to handle the actual message processing and handling.

a. Kafka Connect can be run in distributed and standalone modes, with separate properties files at `/etc/kafka/connect-distributed.properties`.

b. Start Kafka Connect by running:

```
sudo systemctl start confluent-kafka-connect
```

After starting all the components, initiate the full load from the Qlik Replicate side.

Note: Schema Registry supports only Avro format, so select the appropriate options in the Qlik Replicate Task.

Kafka Sink Configuration Manager API

Overview

The Kafka Sink Configuration Manager API allows you to:

- Monitor the health of Kafka services.
- Start, stop, and restart Kafka services.
- Add and delete sink connectors that replicate data from Kafka to Actian Vector.

Installation Path

By default, the Kafka Sink Configuration Manager API is installed at:

```
/opt/insightsoftware/hubble-accelerator/5.0.x/hubble-sinkconnector
```

Authentication

The system supports nginx basic authentication for the Kafka Sink Configuration Manager API and Kafka UI.

Password File Location

The password file is located at `/etc/nginx/.kafkaui_swagger_pass`.

View Current Credentials

To view the current authentication credentials, use the following command:

```
cat /etc/nginx/.kafkaui_swagger_pass
```

The file contains the username and encrypted password in the format:

```
admin:$apr1$B2ZSKTyJ$r/nbs1X23J9WVsNBxZR1f/
```

Change Password

To change the password for the admin user, run the following command:

```
sudo htpasswd /etc/nginx/.kafkaui_swagger_pass admin
```

After changing the password, restart the nginx service:

```
sudo systemctl restart nginx
```

The following are the default authentication credentials:

- **Username:** admin
- **Password:** hubble

Caution: For security, it is recommended to change the default password after initial setup.

Restart the Hubble Sink Connector

After updating the password, restart the service to apply the changes:

```
sudo systemctl stop hubble-sinkconnector
```

```
sudo systemctl start hubble-sinkconnector
```

Basic Commands

Startup

To start the `hubble-sinkconnector` service, use the following command:

```
systemctl start hubble-sinkconnector
```

Shutdown

To stop the hubble-sinkconnector service, use the following command:

```
systemctl stop hubble-sinkconnector
```

Note: The `hubble-sinkconnector` service must be run as root.

Usage Guide

- "Using Kafka Services API" below
- "Using SinkConnector API" on page 35

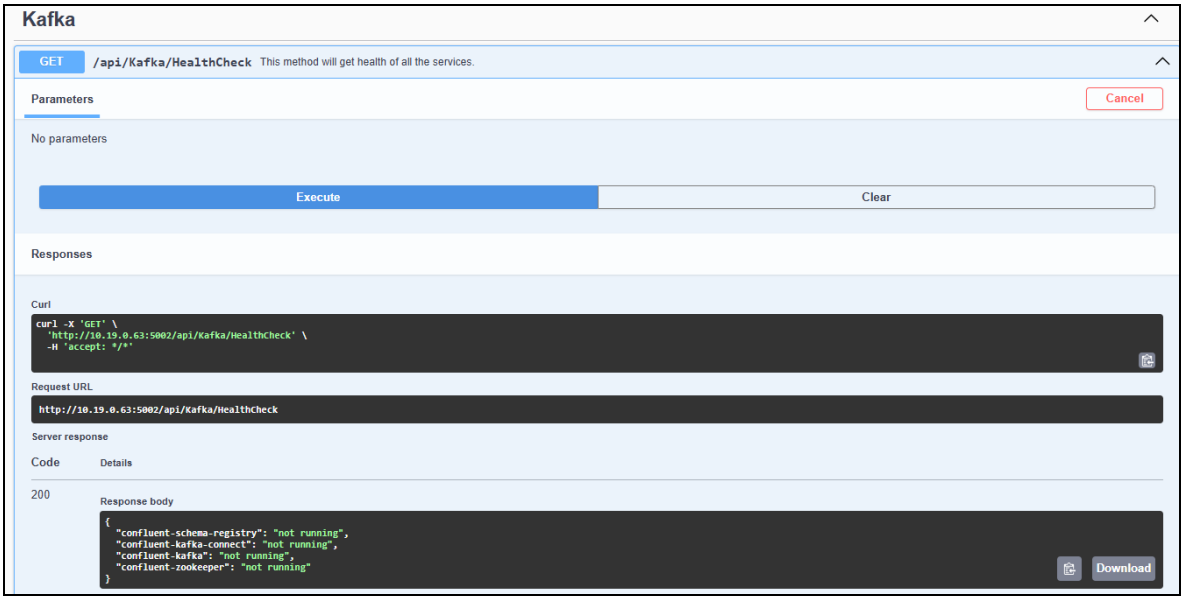
Using Kafka Services API

Two APIs are provided to check the health of Kafka services and change their status:

- HealthCheck
- KafkaService

HealthCheck

- This API provides the health status of all services related to Kafka.
- As part of the accelerator, we bundle Kafka, Zookeeper, Kafka Connect, and Schema Registry.
- If any of these services are not running, replication stops working.



HealthCheck API Access

To access the Healthcheck API, open the Swagger UI in your browser using the SinkConnector service URL:

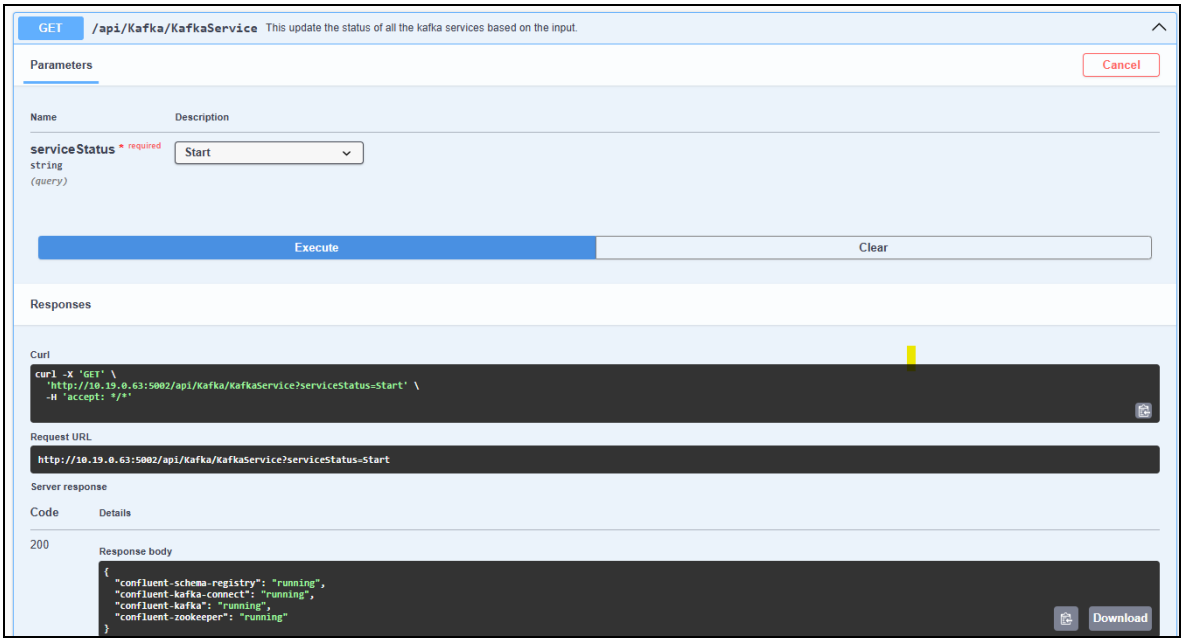
http://{IP_Address}/Swagger/index.html

Replace {IP_Address} with the actual IP address of your SinkConnector service.

KafkaService

- This API is used to start, stop, or restart all Kafka services.

- If you encounter issues with any Kafka component or if any component is not running, you can use this API to manage the services.



Kafka UI Access

To monitor Kafka services visually, use the Kafka UI:

Kafka UI: http://{IP Address}/kafkaui

Replace {IP Address} with the actual IP address of your Kafka host.

Troubleshooting

1. **Check Log Files:**
   - Navigate to the folder: **/opt/insightsoftware/hubble-accelerator/5.0.x/hubble-sinkconnector/logs/**
   - Review the log files for any issues.
2. **Identify Errors:**
   - If any Kafka service fails to start, use the following commands to identify errors specific to the service:

```
journalctl -xefu confluent-kafka-connect
```

```
journalctl -xefu confluent-schema-registry
```

```
journalctl -xefu confluent-confluent-zookeeper
```

```
journalctl -xefu confluent-kafka
```

3. Restart the service.

Using SinkConnector API

Three APIs are provided to check the health of Kafka services and change their status:

- SinkConfiguration
- AddOnlySinkConfiguration
- RemoveSinkConnector

API Access

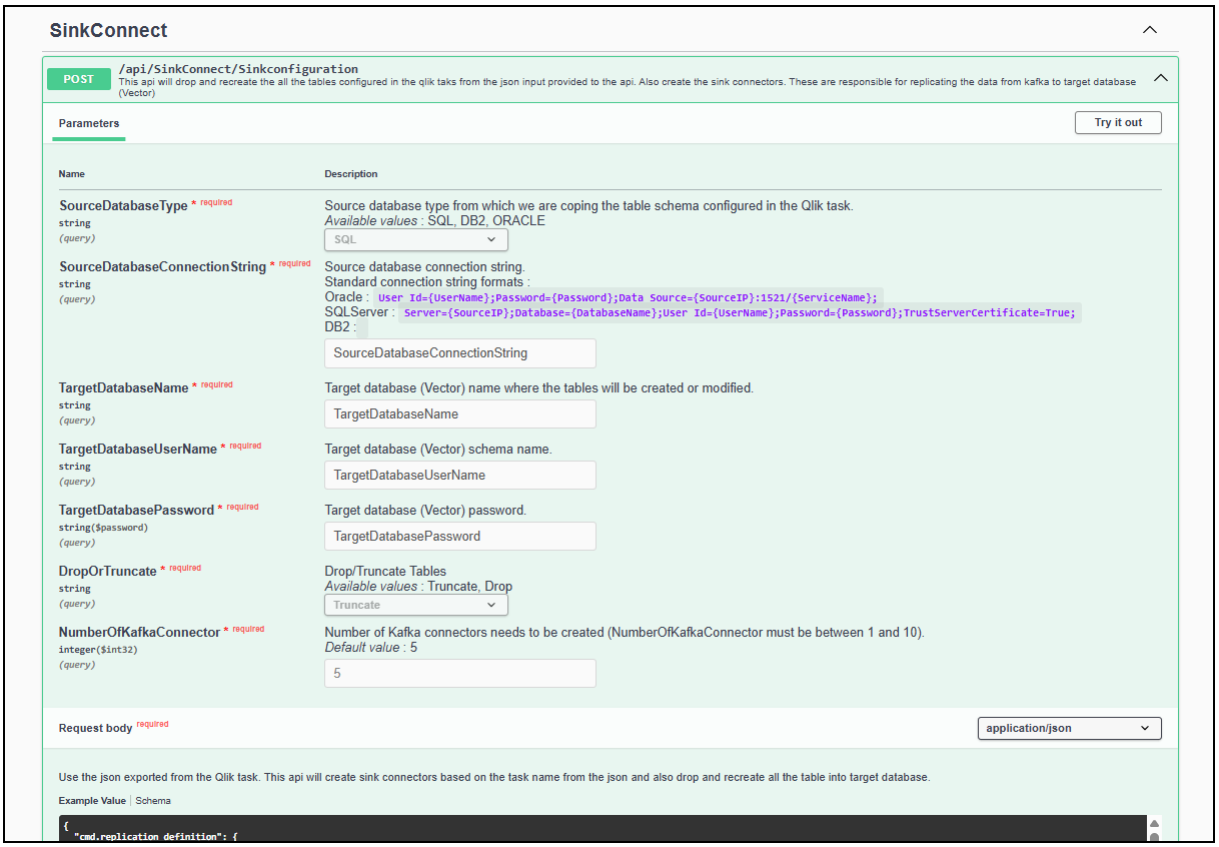
To use the SinkConnect APIs, open the Swagger UI in your browser:

Swagger UI: http://{IP Address}/Swagger/index.html

Replace {IP Address} with the actual IP address of your SinkConnector service.

SinkConfiguration

This API is used to create target tables (drop and re-create if they exist) and create the sink connector to replicate data from Kafka to the target.

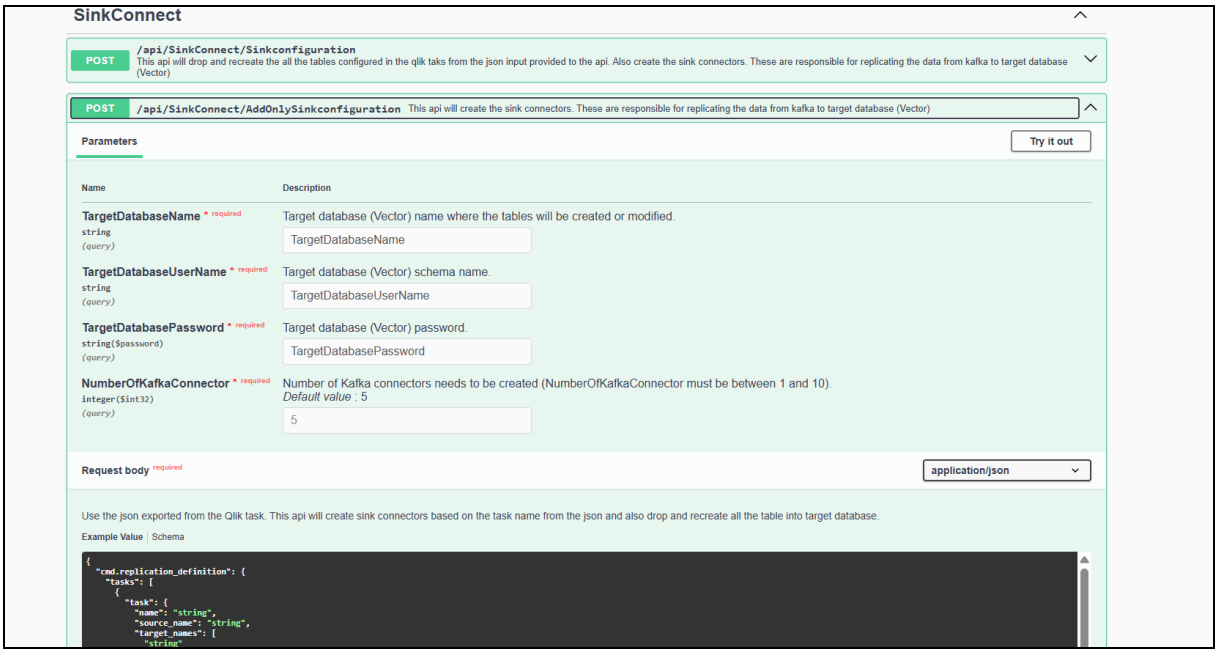


Steps:

1. Select the source database type (SQL, Oracle, DB2) from which data will be replicated.
2. Provide the connection string of the source database (a standard template is provided in the API).
3. Enter the target database username and password.
4. Specify the number of Kafka connectors to be created (default is 5). This allows connectors to run in parallel, similar to Qlik (5 tables at a time).
5. In the request body section, post the Qlik task JSON file. This file helps determine which tables need to be created in the target database.
6. Click **Execute**. This API will create all the tables in the target database and also create the sink connectors to replicate data from Kafka to the Actian Vector database.

AddOnlySinkConfiguration

This API is used to create the sink connector to replicate data from Kafka to the target (it does not create target tables).



Steps:

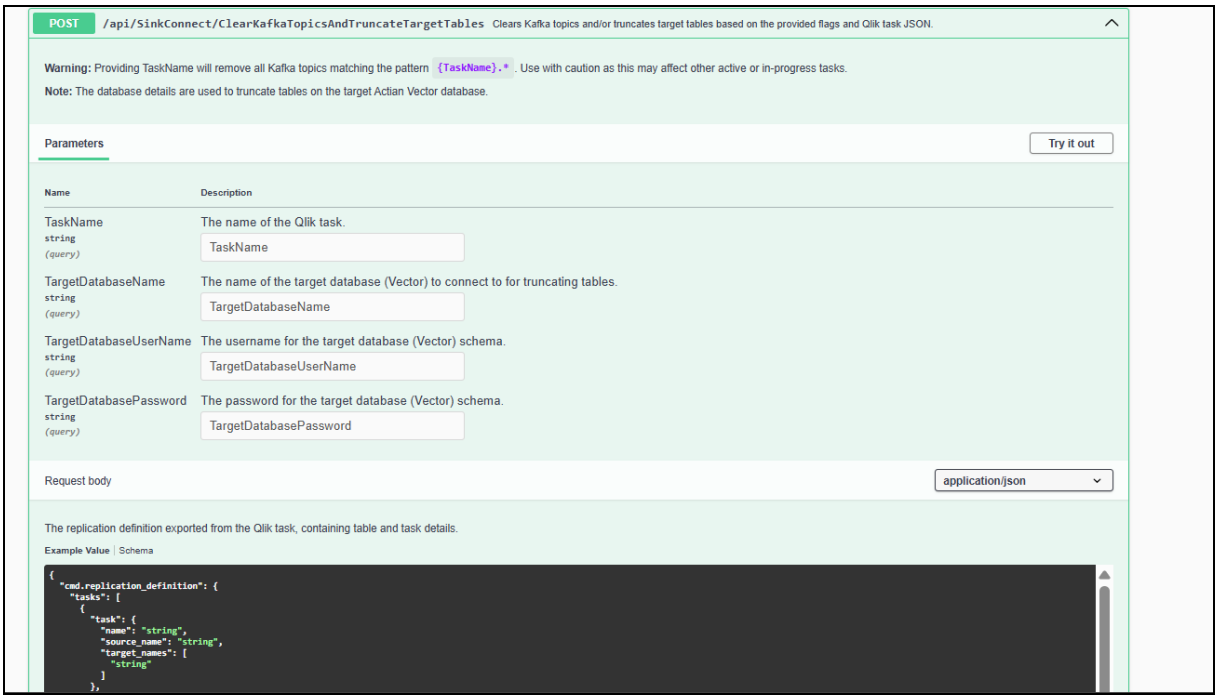
1. Enter the target database username and password.
2. Specify the number of Kafka connectors to be created (default is 5). This allows connectors to run in parallel, similar to Qlik (5 tables at a time).
3. In the request body section, post the Qlik task JSON file. This file helps distribute the tables into different Kafka connectors for data replication.
4. Click **Execute**. This API will create the sink connectors to replicate data from Kafka to the Actian Vector database.

ClearKafkaTopicsAndTruncateTargetTables

This API endpoint clears messages from specified Kafka topics and truncates associated target database tables.

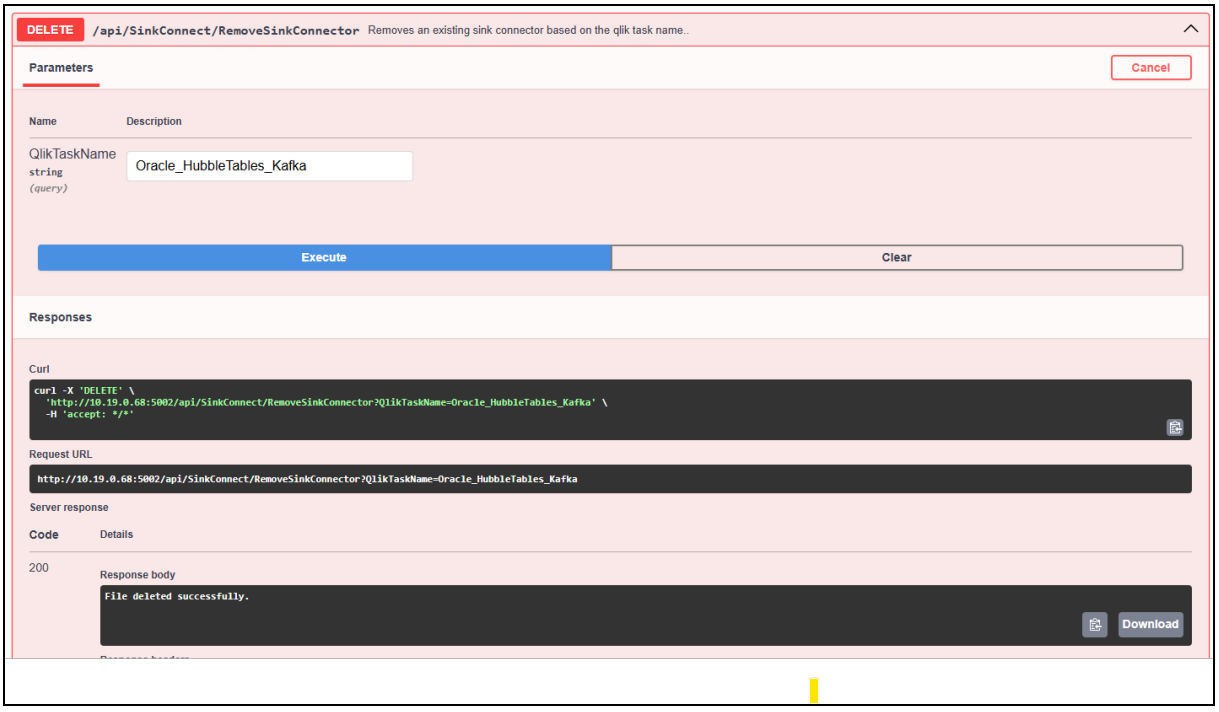
Caution: Providing `TaskName` will remove all Kafka topics matching the pattern `{TaskName}.*`. Use with caution as this may affect other active or in-progress tasks.

Note: The database details are used to truncate tables on the target Actian Vector database.



RemoveSinkConnector

This API is used to remove existing sink connectors.



Provide the Qlik task name used when creating the sink connector. This will delete all sink connectors associated with that task name.

Troubleshooting

1. **Check Log Files:**
   - Navigate to the folder: **/opt/insightsoftware/hubble-accelerator/5.0.x/hubble-sinkconnector/logs/**
   - Review the log files for any issues.
2. **Schema Reading Issues:** When reading schema details from the source database, connection failures may occur due to network issues. If this happens, rerun the execution.
3. **Table Creation Issues:** If there are issues creating tables in the target database, the sink connectors will not be created. Correct these errors and try again.
4. **Sink Connector Creation Issues:**
   - Issues may arise when creating the sink connector, especially if the existing connector is still processing messages from Kafka. When posting the sink connector, the existing connector is deleted and recreated.
   - Wait until all messages are processed or stop all Kafka services and restart them.
5. **Restart the service:** After addressing the issues, restart the service as needed.

Hubble Accelerator Row Count Comparer

Overview

The Hubble Accelerator Comparer utility compares row counts between a source file (.tsv) and a target database.

- It takes the table names and row counts from the `.tsv` file.
- Connects to the target database using a provided connection string.
- Retrieves the row counts for each table.
- Compares the row counts.

The results are generated in a CSV report, showing:

- Table Name
- Source Row Count

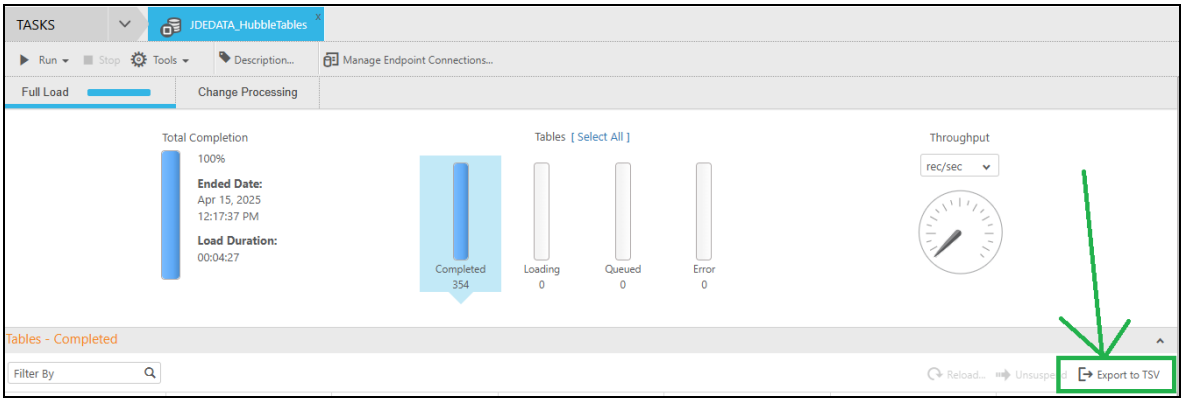
- Target Row Count
- Row Count Match

Execution

1. Navigate to the downloaded folder.
2. Extract the contents.
3. Open `Hubble.Accelerator.RowCountComparer.exe`.

Notes

- You can obtain the .tsv file from Qlik (see attached screenshot for reference).



- Contact insightsoftware Support for sample `.tsv` and generated `.csv` files.

Replication Monitoring using Kafka UI

Overview

Kafka UI is a web-based interface for managing and monitoring Apache Kafka clusters. This topic provides setup instructions, basic commands, and guides for common operations.

Install Path

- Installation Directory: `/opt/kafka-ui`
- Executable JAR File: `/opt/kafka-ui/kafka-ui.jar`
- Systemd Service Name: `kafka-ui`

Basic Commands

- Start Kafka UI: `sudo systemctl start kafka-ui`
- Stop Kafka UI: `sudo systemctl stop kafka-ui`
- Restart Kafka UI: `sudo systemctl restart kafka-ui`
- Check Kafka UI Status: `sudo systemctl status kafka-ui`
- Enable Kafka UI to Start on Boot: `sudo systemctl enable kafka-ui`
- Disable Kafka UI from Starting on Boot: `sudo systemctl disable kafka-ui`

Getting Started with Kafka UI

1. Open your browser and go to: http://{IP Address}/kafkaui
2. Login is required using nginx authentication.

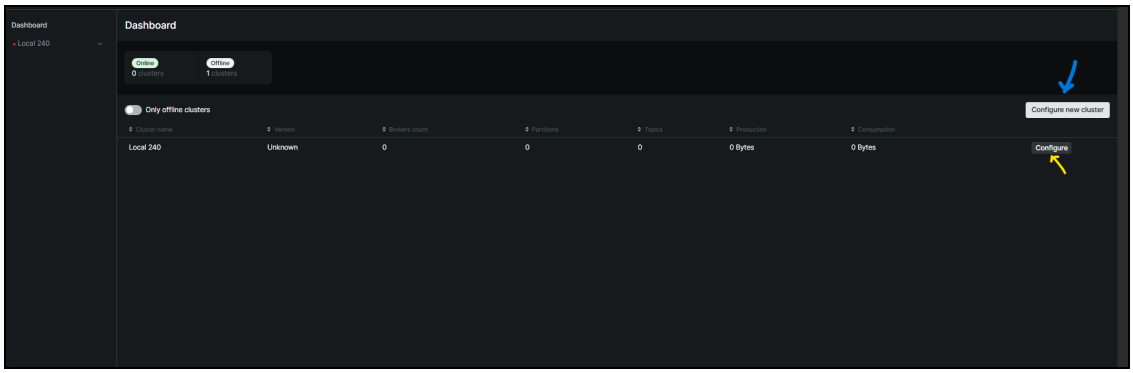
   Default values:

   - **Username:** admin
   - **Password:** hubble
3. You should see the dashboard listing all configured Kafka clusters.

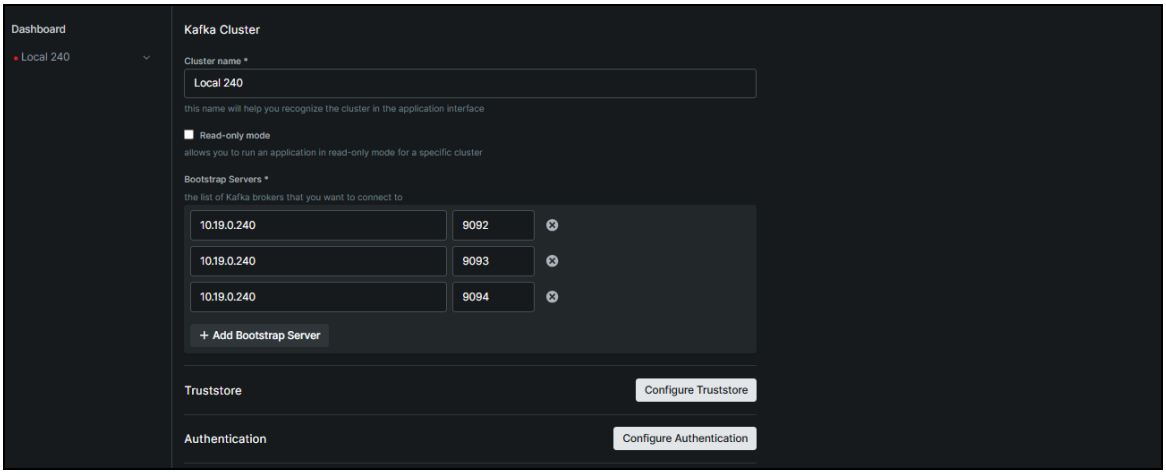
How to Configure or Set up a New Cluster

1. **Add the Cluster Details:** Create a new cluster in the Dashboard section by specifying:
   - Cluster Name
   - Environment

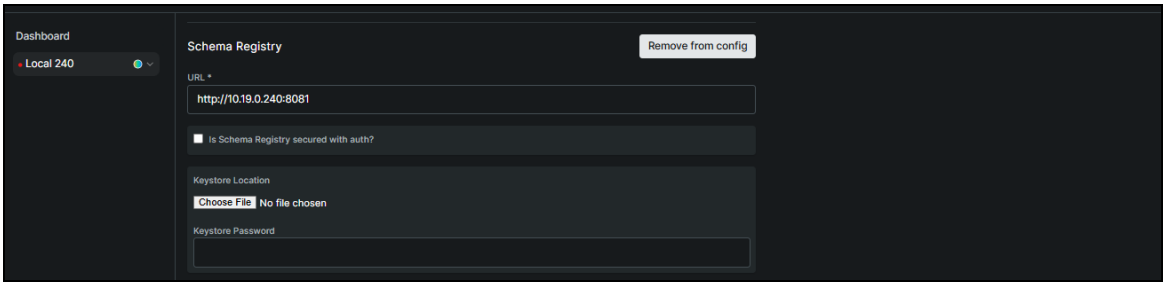
- Cluster ID or Endpoint



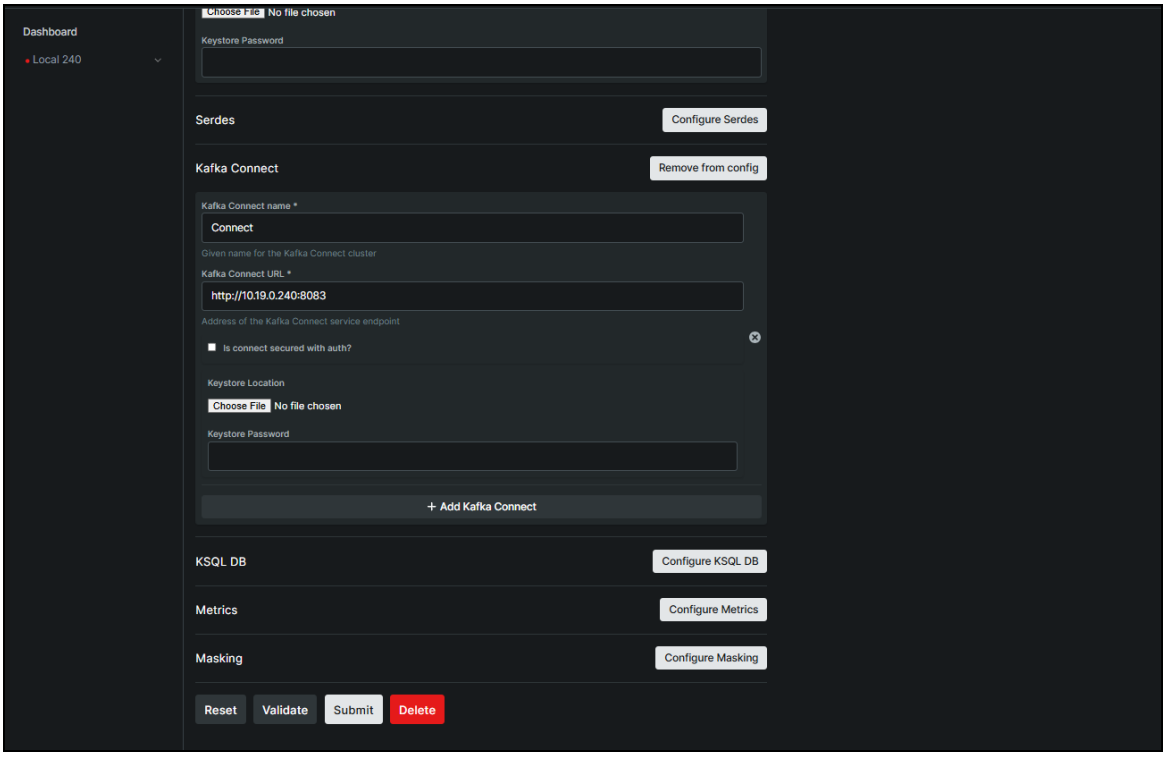
2. **Add Kafka Broker Details:** Under the **Bootstrap Server** section, provide the **Kafka broker URL(s)** (for example, broker1:9092,broker2:9092).



3. **Add Schema Registry Details:** In the **Schema Registry** section, enter the **Schema Registry URL** (example, http://localhost:8081) and any authentication credentials, if required.



4. **Add Kafka Connect Details:** In the **Kafka Connect** section, provide the **Kafka Connect REST URL** (for example, http://localhost:8083) along with any authentication details if required.



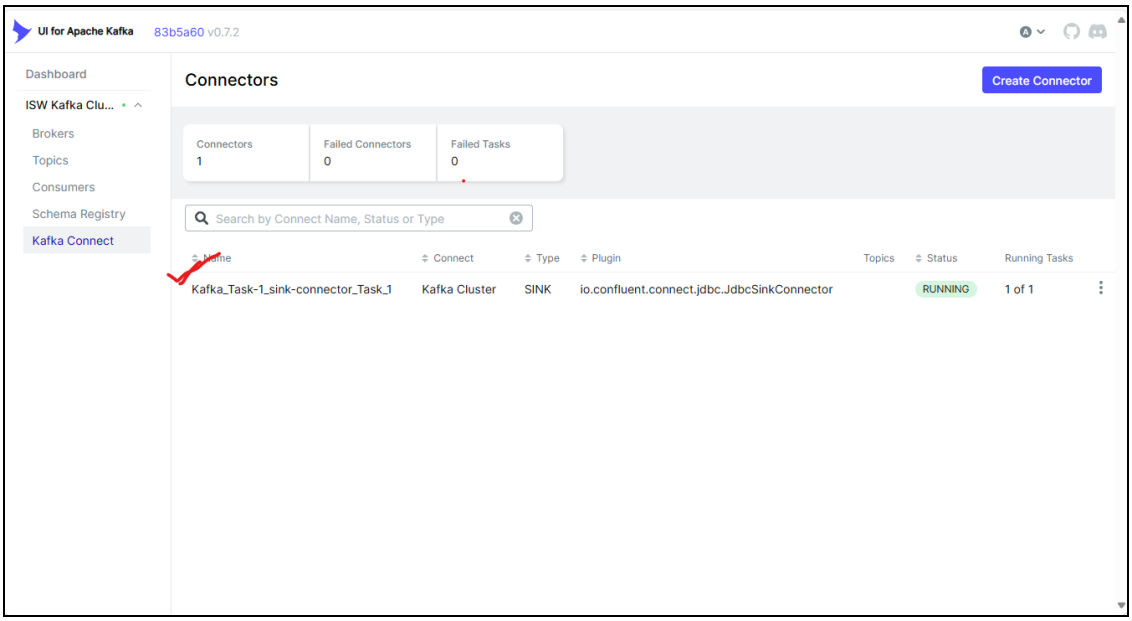
How to View the Replication Stats in Kafka UI

1. Open **Kafka UI** in your browser.
2. Navigate to the **Topics** section.
3. Select the topic related to your replication task.
4. View key metrics such as:
   - Message count
   - Partition offsets
   - Consumer lag
   - Throughput
5. Optionally, go to the **Consumers** or **Consumer Groups** section to monitor which consumers are reading the replicated data and their current offsets.

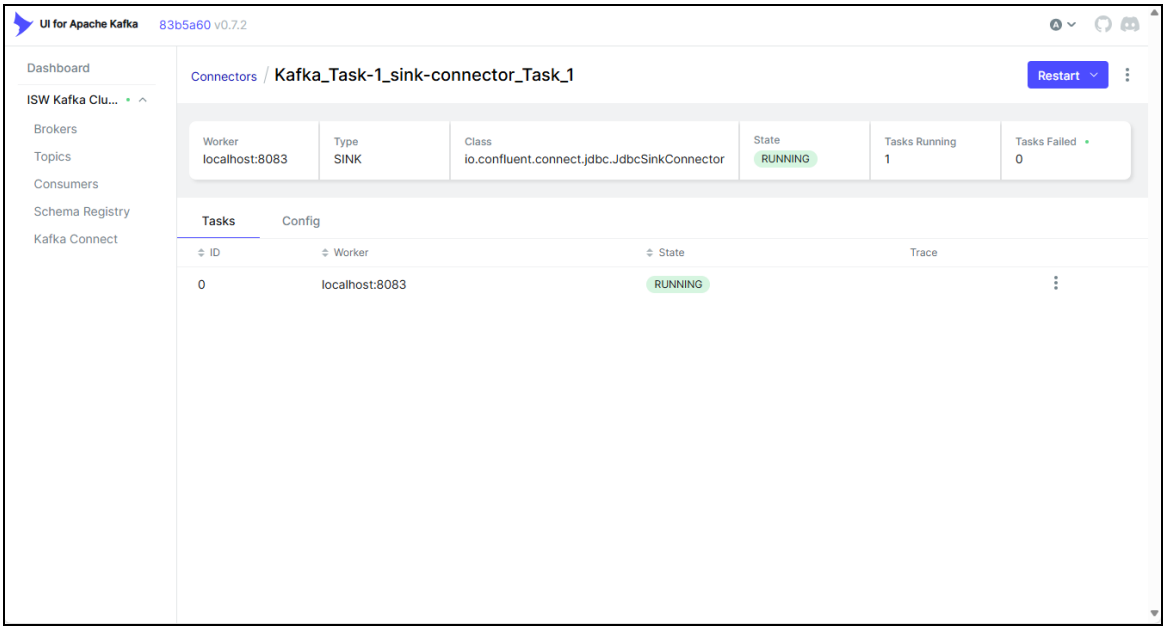
This allows you to track the health and progress of your replication jobs in real time.

How to View the Kafka Connect Stats?

1. From the Kafka Connect section:
   - Select the connector.
   - Go to the **Tasks** tab.
   - Check for failing tasks.
2. Errors will also appear in:
   - The Status of connectors or tasks (for example, FAILED)
   - Logs from the Kafka Connect server (see logs section)



3. Open the connector you want to inspect.

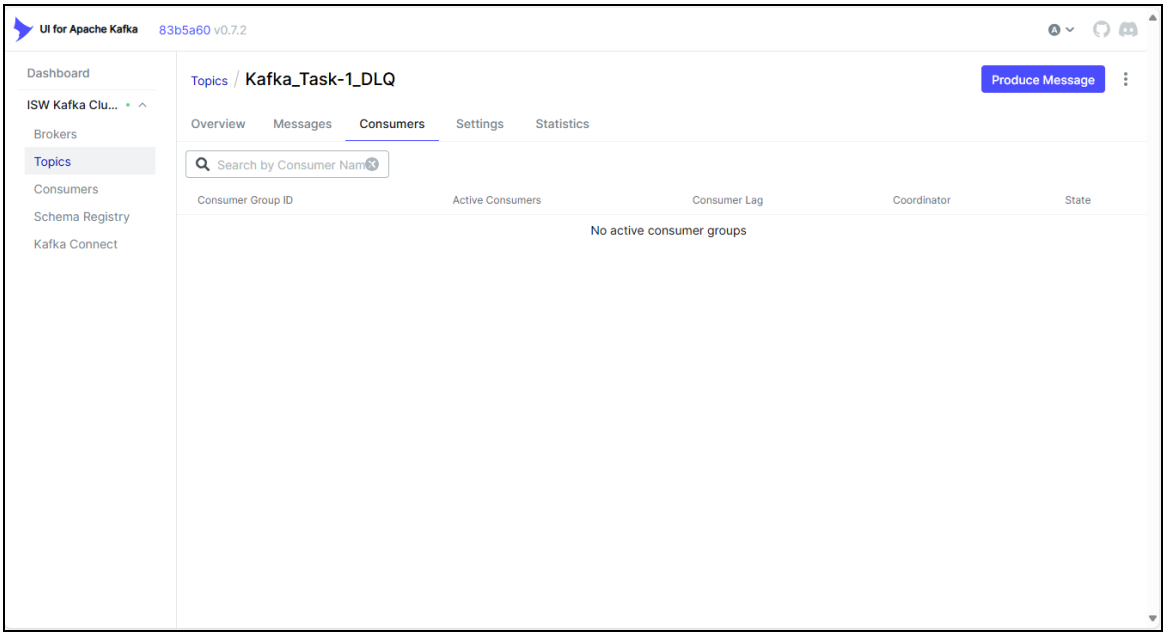


Replication stats such as task status, partitions, and offsets will be visible in the connector overview.

How to View the Failed Topic/Table Stats

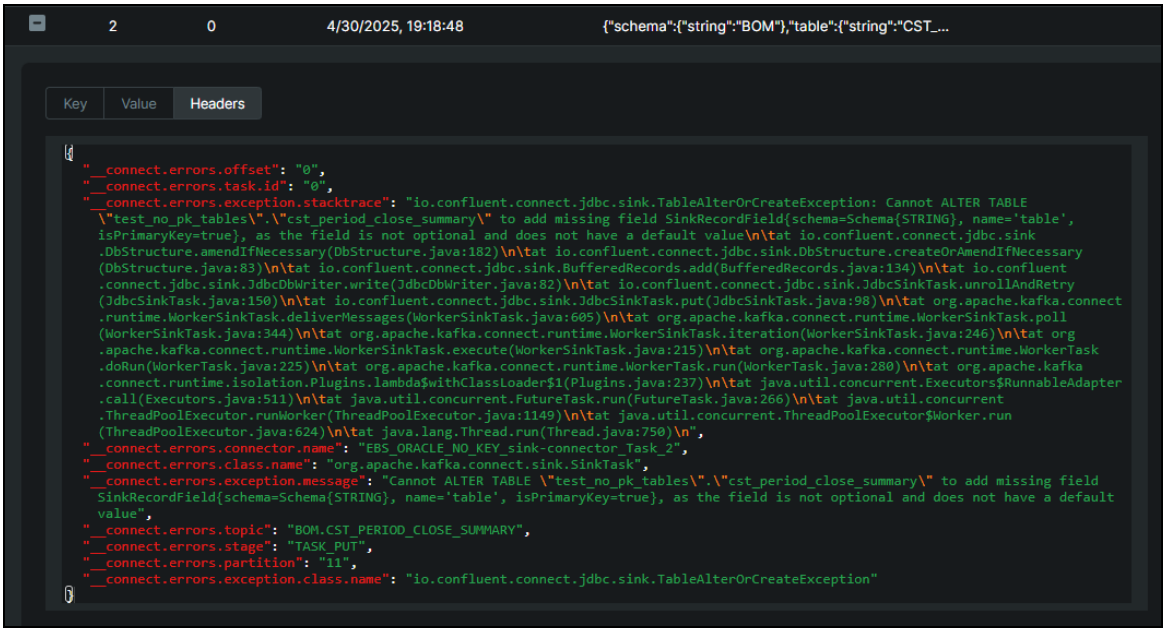
> **Note:** In this setup, if message processing fails, the messages are automatically routed to the **DLQ (Dead Letter Queue)** topic.

1. You can find the original topic name and error details in the headers section of each message in the DLQ. This helps trace back the source and diagnose the issue.



2. Errors typically surface in **Dead Letter Topics (DLTs)**, which you can inspect like normal topics.

> **Tip:** To troubleshoot failures, check the messages in the **DLQ (Dead Letter Queue)** topic. In the **Headers** section of each message, you can usually find the exception message or error details that caused the failure.



How to Clean up the Records in Brokers?

> **Important:** Use with caution. Cleanup is typically managed by Kafka retention policies, and manual deletion may lead to data loss if not handled carefully.

1. Open **Kafka UI** in your browser.
2. Go to the "**Topics**" section and select the topic you want to clean.
3. Click on the **Actions** or **Settings** menu next to the topic name.
4. Choose **Clear Messages** to remove all records from the topic.

When to Clean:

Manual cleanup is recommended in scenarios such as:

- Re-replicating tables without primary key columns in insert mode, where duplicate records may be generated.
- Resetting a topic before a fresh load to ensure no stale or inconsistent data remains.
- Clearing error topics after the issues have been resolved and logged.

Refer to Insert Mode Replication With Kafka (Oracle EBS) for details.

How to Clean up the Records in Brokers

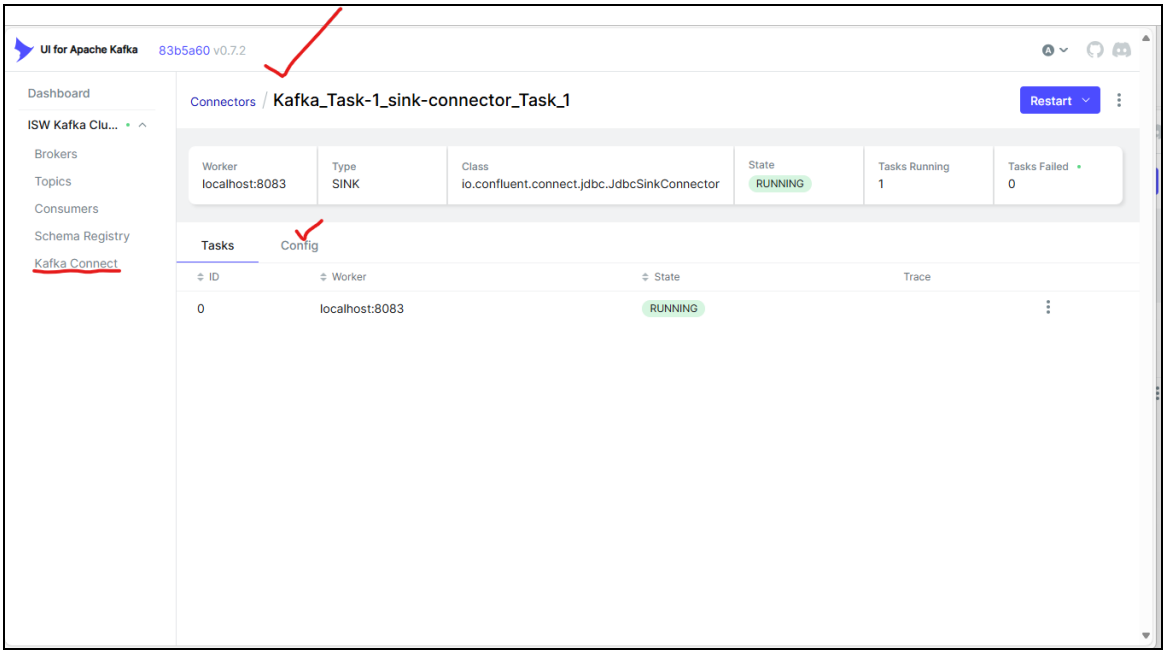
> **Important:** Use with caution. Cleanup is typically managed by Kafka retention policies.

- To manually delete a topic:
  1. Go to the **Topics** tab.
  2. Select the topic you want to delete.
  3. Click the **Delete** button (if enabled in Kafka broker and UI config).
- Alternatively, using Kafka CLI:

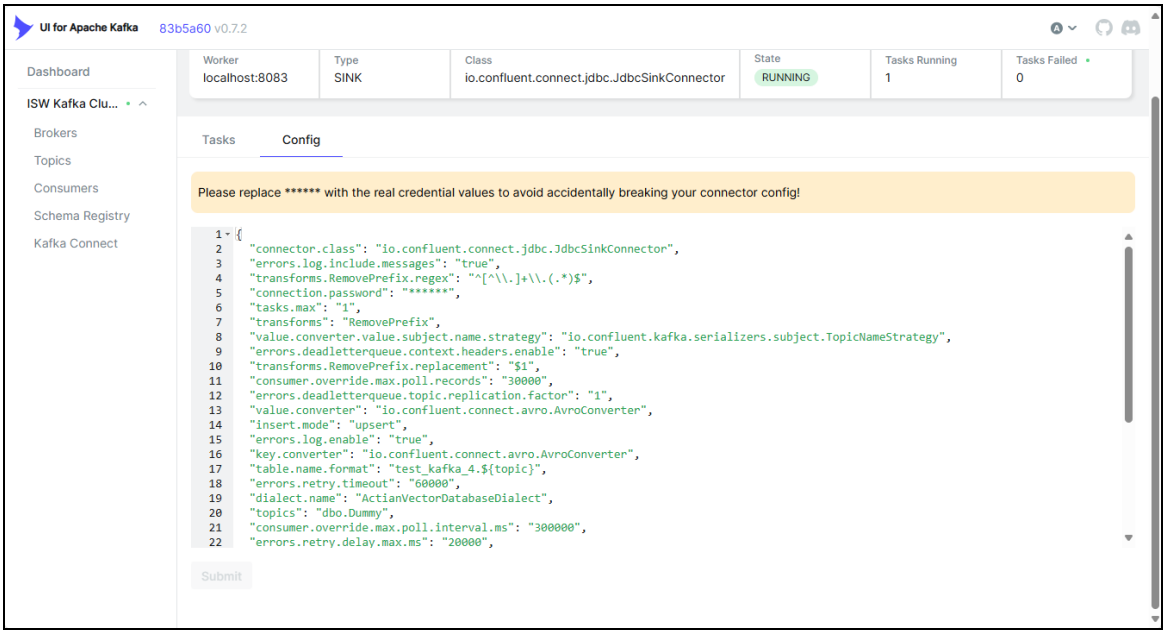
```
kafka-topics.sh --bootstrap-server localhost:9092 --delete --
topic your-topic-name
```

How to Update a Property in a Connector (for example, batch.size)

1. Go to the **Connect Clusters** section.
2. Open the connector you want to modify.



3. Click the **Edit Configuration** (pencil icon).



4. Update the desired property, for example:

```
{
```

```
"batch.size": "5000",
```

```
"connection.password": "******"
```

```
}
```

> **Note:** Ensure you update the `connection.password` field when making any property changes, as the characters **"******"** will be treated as a new password if not modified.

5. Save for the changes to apply.

How to View the Error Logs of Kafka UI

- Although Kafka UI does not directly expose backend logs, you can view them using `journalctl` with the following command:

  ```
  sudo journalctl -u kafka-ui -f
  ```

  This command displays real-time logs from the Kafka UI systemd service.

- For debugging Kafka Connect-related errors, check the logs of your Kafka Connect instance.

  These logs are typically found in `/logs/connect.log` or via Docker container logs if Kafka Connect is running in a container.

Hubble Backup API

- Hubble Backup API Installation Path
- Hubble Backup API Basic Commands
- Hubble Backup API Vector Maintenance Guide - Refer to the *Hubble Accelerator Maintenance Guide*.

Hubble Backup API Installation Path

Hubble Backup API will be installed by default on:

```
/usr/libexec/hubble-applianceapi
```

Hubble Backup API Basic Commands

- Start up:

  ```
  systemctl start hubble-applianceapi
  ```

- Shut down:

  ```
  systemctl stop hubble-applianceapi
  ```

Install Hubble Accelerator

Follow these steps to install Hubble Accelerator:

1. Download the Hubble Accelerator RPM from the SE URL provided by insightsoftware Support.
2. Install the local Hubble Accelerator rpm (replace **5.x.x-x** with the actual version of the rpm, example: **5.0.0-78**):

   ```
   yum localinstall hubble-accelerator-5.0.0-xx.exx.x86_64.rpm
   ```

   This step will create the Hubble Accelerator directories and copy the installation packages for all the Accelerator components.
3. Run the Hubble Accelerator installer that will install all the rpms for all the components, and configure them:

   > **Note:** Replace **5.x.x** with the actual version of the rpm, for example, **5.0.0**.

   ```
   /opt/insightsoftware/hubble-accelerator/5.0.0/install.sh
   ```
4. Stop Vector service.

   ```
   systemctl stop actian-vectorVW
   ```
5. For Accelerator build **4.3.x** with Vector version **6.3.x** or later, Actian has provided the flexibility for switching back to the old behavior of not padding spaces to strings using `CHAR / NCHAR` functions. When `CHAR` and `NCHAR` types are used with x100 tables, the correct behavior is to pad these datatypes with spaces to the explicit and/or implicit length of the datatype. This parameter `x100_nopad_char_compat` adds a backward compatibility configuration option to allow the user to fall back to the old behavior (Vector version 5.x) where these datatypes were not padded.

   To set `ii.localhost.dbms.*.x100_nopad_char_compat` to `ON` in `config.dat`:

   i. Go to the directory:

   ```
   cd $II_SYSTEM/ingres/files/
   ```

   ii. Open config.dat:

   ```
   vi config.dat
   ```

   iii. Find and change the line to:

```
ii.localhost.dbms.*.x100_nopad_char_compat: ON
```

iv. Save and exit:

```
:wq!
```

6. In Vector version **6.3**, the default configuration for string truncation is set to "fail," whereas in Vector version **5.0**, it was "ignore." To apply this change, follow these steps:

   i. Navigate to the directory containing the config.dat file:

   ```
   cd $II_SYSTEM/ingres/files/
   ```

   ii. Open config.dat in edit mode using vi:

   ```
   vi config.dat
   ```

   iii. Enter `i` to enter insert mode.

   iv. Change the value of `ii.<host>.config.string_truncation` to `ignore`.

   ```
   ii.<host>.config.string_truncation: ignore
   ```

   v. Save the updated `config.dat` file and exit the editor by entering `:wq!`.

   f. Restart the Vector database to apply the changes.

7. To improve performance, follow these steps to update the config.dat file:

   i. Navigate to the directory containing the `config.dat` file:

   ```
   cd $II_SYSTEM/ingres/files/
   ```

   ii. Open `config.dat` in edit mode using `vi`:

   ```
   vi config.dat
   ```

   iii. Enter insert mode by typing `i`.

   iv. Update the following values:

- Change ii.localhost.dbms.*.qef_dsh_memory to 2147483648:

```
ii.localhost.dbms.*.qef_dsh_memory: 2147483648
```

- Change ii.localhost.dbms.*.qef_memory to 8397044736:

```
ii.localhost.dbms.*.qef_memory: 8397044736
```

- Change ii.localhost.dbms.*.qef_sort_mem to 4194304:

```
ii.localhost.dbms.*.qef_sort_mem: 4194304
```

- Change ii.localhost.dbms.*.stack_size to 2048576:

```
ii.localhost.dbms.*.stack_size: 2048576
```

v. Exit insert mode by clicking **Esc**.

vi. Save the updated config.dat file and exit the editor with `:wq!`.

vii. Restart the Vector database to apply the changes.

8. To update the heap space, follow these steps:

i. Navigate to the directory containing the connect-distributed file:

```
cd /usr/bin/
```

ii. Open connect-distributed in edit mode using vi:

```
vi connect-distributed
```

iii. Locate the line:

```
export KAFKA_HEAP_OPTS="-Xms6M -Xmx8G"
```

iv. Enter insert mode by typing **i**.

v. Change the value from -Xms6M -Xmx8G to -Xms8G -Xmx12G:

```
export KAFKA_HEAP_OPTS="-Xms8G -Xmx12G"
```

vi. Exit insert mode by typing **Esc**.

vii. Save the updated `connect-distributed` file and exit the editor with **`:wq!`**.

9. To update the `vectorwise.db.conf` file, follow these steps:

i. Navigate to the directory containing the `vectorwise.db.conf` file:

ii. Open `vectorwise.db.conf` in edit mode using `vi`:

iii. Enter insert mode by typing **i**.

iv. Update the following values:

i. Change `max_memory_size` to `100133242368`:

```
max_memory_size=100133242368
```

ii. Change `max_transactions` to `30000`:

```
max_transactions=30000
```

iii. Change `max_parallelism_level` to `24`:

```
max_parallelism_level=24
```

iv. Change `max_table_update_ratio` to `0.05`:

```
max_table_update_ratio=0.05
```

v. Change `max_global_update_memory` to `0.25`:

```
max_global_update_memory=0.25
```

v. Exit insert mode by typing **Esc**.

vi. Save the updated `vectorwise.db.conf` file and exit the editor with `:wq!`.

10. Stop all services using sink connector

11. Start all services using sink connector

12. After updating the `config.dat` file, restart the vector instance and run the post-installation script to implement the non-default parameter. Use the following commands as the actian or root user:

    Restart the Vector instance (in case of upgrading an Accelerator (for implementing non-default Db parameter) after updating `config.dat` file using below commands:

    ```
    systemctl restart actian-vectorVW
    ```

    or

    Restart the Vector instance and run post installation script (in case of fresh installation of an Accelerator for implementing non-default parameter) after updating `config.dat` file using below command:

    ```
    systemctl restart actian-vectorVW
    ```

13. Reboot the machine.

14. Start all services using sink connector.

15. Check if `areplicate` is running if not start using below command.

    ```
    /etc/init.d/areplicate start
    ```

16. In case of upgrade, for version **3.3.x** or higher, the current vector database 'db' needs to be upgraded:

    ```
    upgradedb db
    ```

17. Test if you can access the Vector database:

    ```
    sql db
    ```

18. Confirm that replication is installed.

    ```
    repctl version
    ```

19. The password for the replication user, and the Hubble user for the database will be in:

    ```
    cat /opt/insightsoftware/hubble-accelerator/.vector
    ```

Feel free to remove the password from that file for security reasons, but leave the file empty.

20. The password for the replication user, in the replication console can be found in:

```
cat /opt/insightsoftware/hubble-accelerator/.attunity
```

Feel free to remove the password from that file for security reasons, but leave the file empty.

21. Enable and start the Hubble Appliance API service:

```
systemctl enable hubble-applianceapi
```

```
systemctl start hubble-applianceapi
```

Post Installation Steps for Hubble Accelerator

Insert Mode Replication With Kafka (Oracle EBS)

Handle Tables Without Primary Keys in Qlik Replication

Tables were segregated based on their unique indexes by referring to the table DDLs and the reference documentation available in ETRM 12.2.2. This approach ensures accurate identification of uniquely identifiable records, aligning with Oracle's structural definitions and best practices.

Categories

The tables have been segregated into three categories based on the presence of unique indexes:

- **Y:** Tables that have a unique index defined directly in the database.
- **N:** Tables that do not have any unique index defined in the database.
- **N*:** Tables that lack a unique index in the database, but Oracle E-Business Suite (EBS) documentation (as referenced from ETRM 12.2.2) specifies unique index details.

This classification ensures accurate handling during data replication and integrity checks.

Segregated List

Contact insightsoftware Support for the latest segregated list.

Replication Tasks

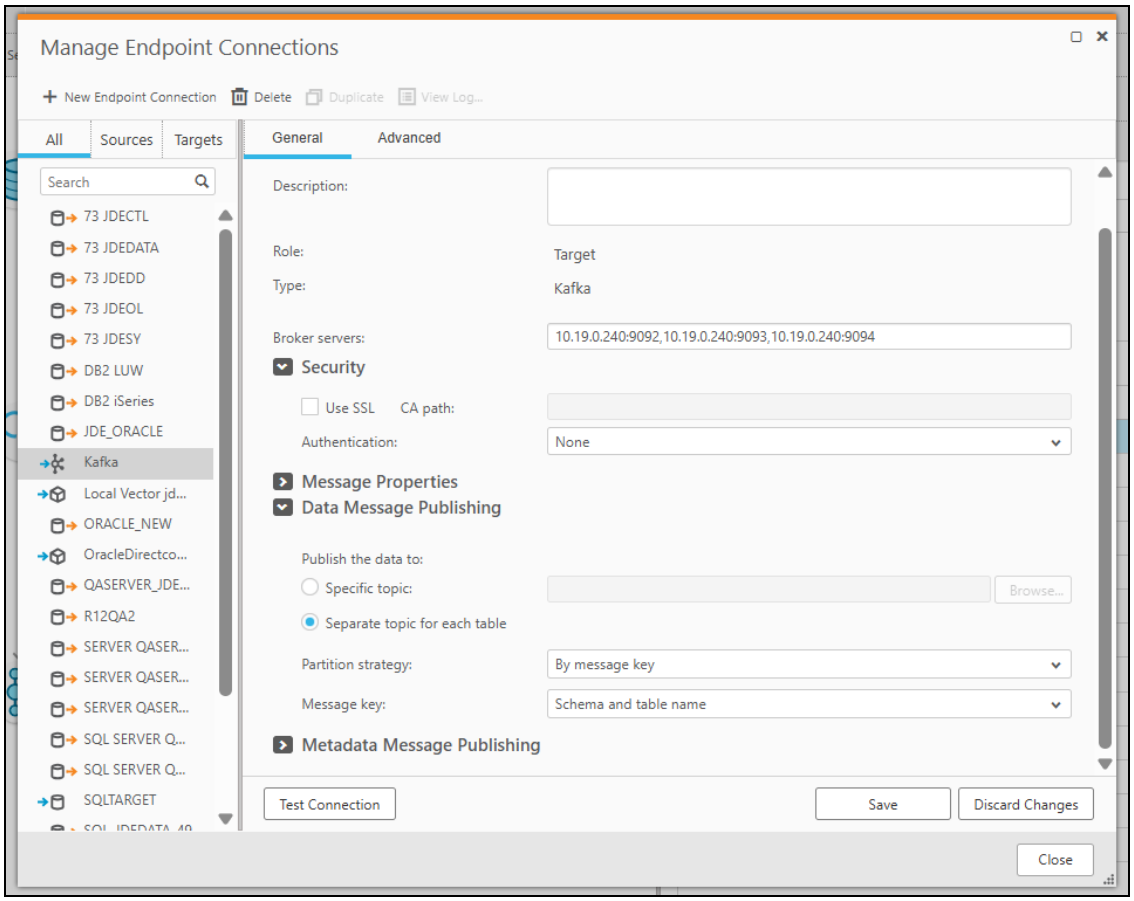
Dedicated Qlik tasks were created to replicate the segregated tables identified based on unique index analysis. These tasks are designed to ensure efficient and reliable data replication, aligning with the structure and integrity defined in the source systems.

Steps to follow for Fresh Insert Mode Run

1. **Create Qlik Task:**

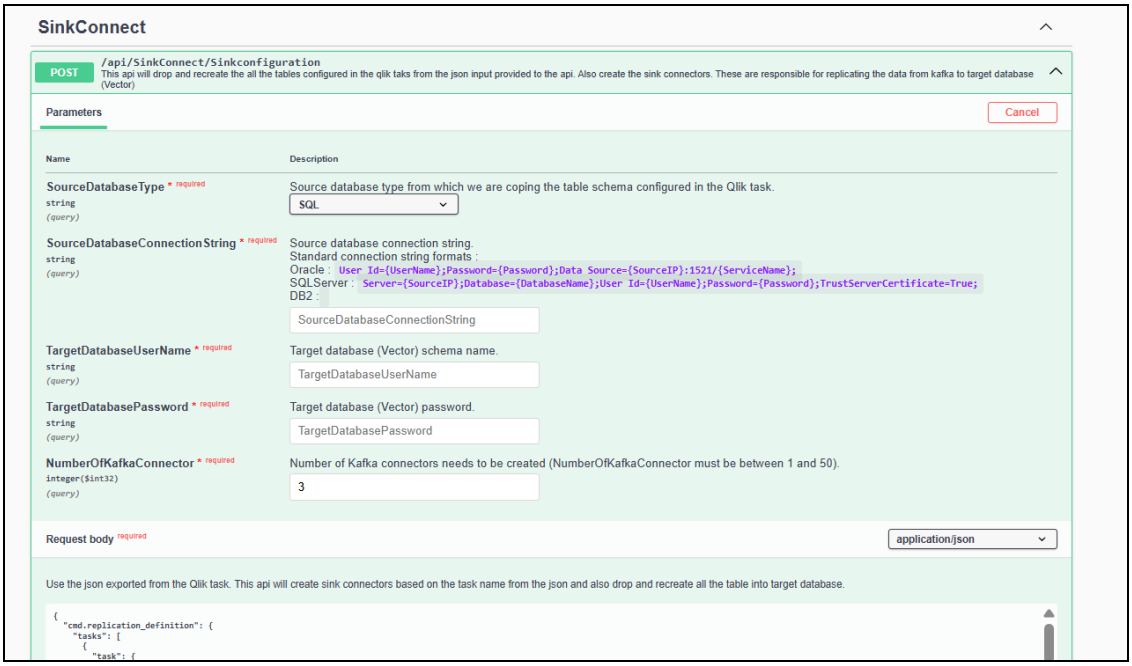
   - Initialize a new Qlik task to load data into the destination.

   - Create a new endpoint to ensure proper data flow and connectivity between Qlik and Kafka.



2. **Swagger API:**

   - Create the necessary tables and post the sink connector configurations. Use a maximum of three Kafka connectors for this process to ensure optimal performance and maintainability.



3. Navigate to Kafka UI, select the **Kafka Connect** tab, and open the newly created **Sink Connector** tasks.

   - Change the sink connector configuration values for all the newly created sink connectors:

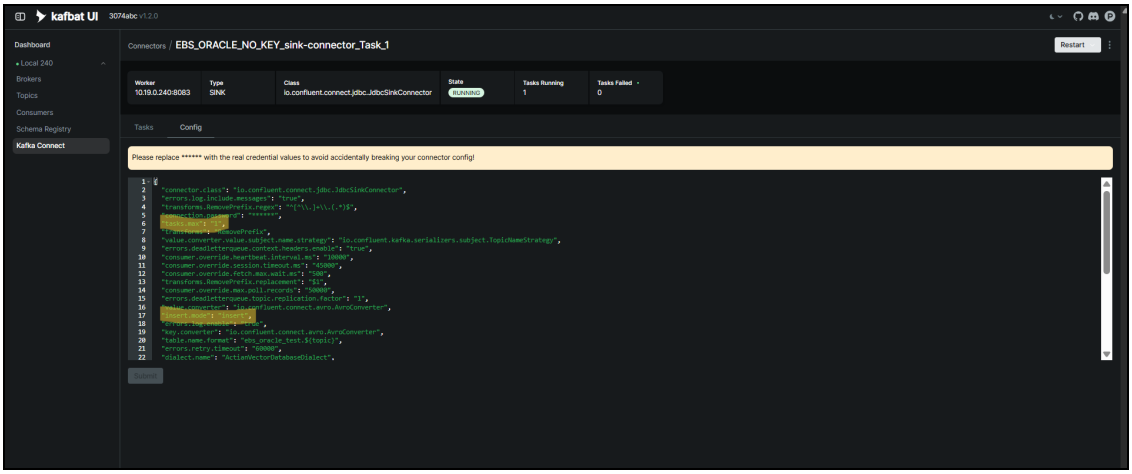
- Insert Mode
- Delete Enable Mode
- PK Mode
- tasks.max Replication

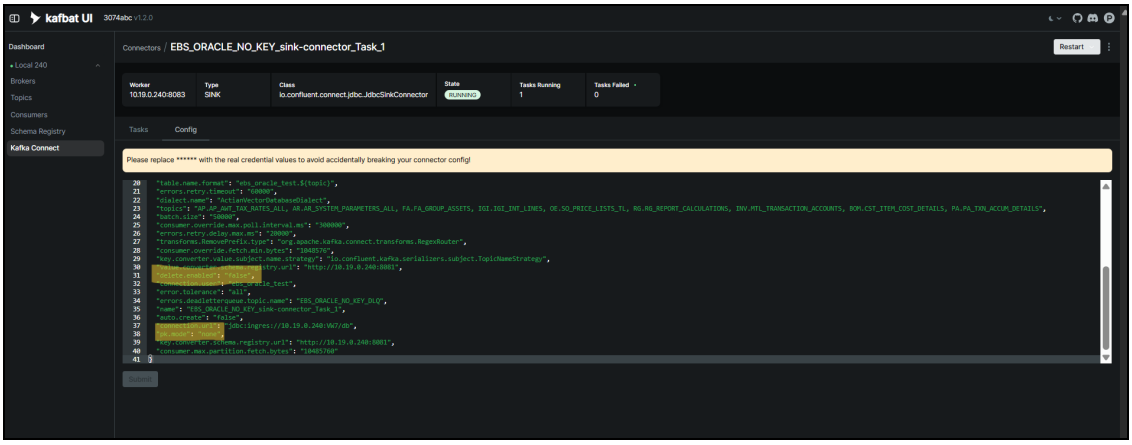
```
"tasks.max": "1",
```

```
"insert.mode": "insert",
```

```
"delete.enabled": "false",
```

```
"pk.mode": "none",
```

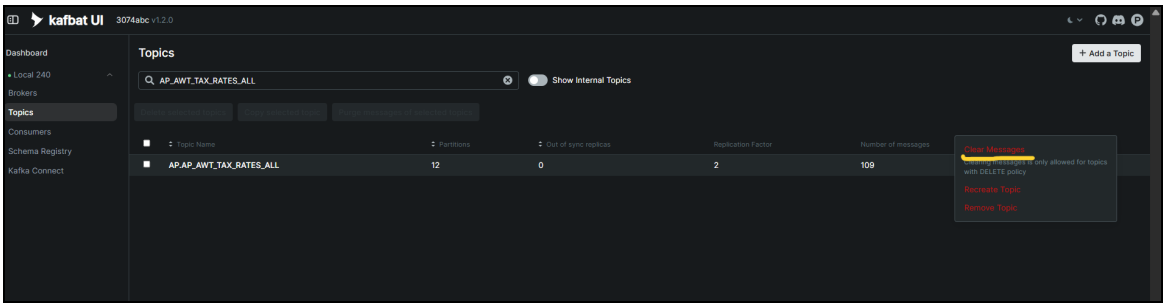




4. Run the Qlik task.

Re-run the Qlik Task in Insert Mode

1. Clear the messages in all the topics (i.e., the tables being replicated).



To ensure that there is no residual or duplicated data in the system, clear the messages in all the topics corresponding to the tables being replicated. This step involves:

- Identifying the relevant Kafka topics that hold the data for the tables being replicated.
- Using Kafka UI or Kafka command-line tools, delete or clear the messages from these topics to reset their state.
- This action is necessary because when running the replication in insert mode, it is necessary to ensure that:

◦ Any previously unprocessed or partially processed data does not get replicated again.

◦ Avoid duplication of data in the target system as we are performing a fresh replication.

By clearing the messages, it is ensured that the replication process only picks up new records and accurately processes the data from the source system into the destination without reprocessing old or redundant entries.

2. Truncate the tables in the target database.

   To ensure that the target database tables are cleared and ready for fresh data, truncate the relevant tables. This step is crucial when replicating data in insert mode because:

   - Truncating the tables removes all existing data from the target tables, ensuring that there is no conflict with previously replicated records.
   - This action ensures that only the newly replicated data from the source system will be inserted into the target tables, preventing any old or stale data from being part of the replication process.
   - Helps to identify the target tables that will receive the replicated data.
   - Use your preferred database management tool or SQL commands to truncate the tables. Sample SQL for truncating:

     ```
     TRUNCATE TABLE table_name;
     ```

3. Run the Qlik task to start the data loading process into the destination, ensuring that all configurations are properly applied.

Support Special Characters in Table and Column Names via Global Rules in Qlik Replication

This topic provides a step-by-step guide to support special characters using global rules in the Qlik Replication tool. This approach ensures that the transformation is applied universally across all tasks.

Issue Description

During the replication of tables, if the tables and columns contain special characters, Kafka Connect is unable to serialize the Kafka messages.

Note: Currently, only the special characters **#** and **$** have been identified and handled to support in table/column names. Any other special characters in the table/column names need to be handled separately.

Resolution

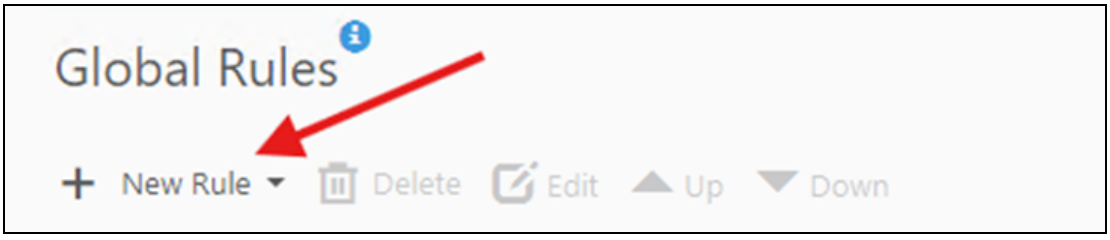
Using global transformations, special characters will be replaced with placeholders. These placeholders will then be replaced with the actual special characters in Kafka Connect.

Prerequisites

- Qlik Replication tool installed and configured.
- Access to the Qlik Replication console.
- Necessary permissions to modify global rules.

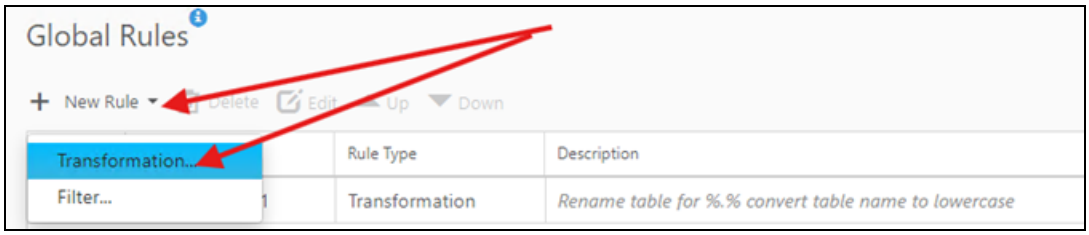
Steps

1. **Access the Qlik Replication Console**
   i. Open your web browser and navigate to the Qlik Replication console URL.
   ii. Log in with your credentials.
2. **Navigate to Global Rules**
   i. In the **Qlik Replication** console, go to the **Replication** task and designer then click ****Global Rules****.
   ii. Click ****Add Global Rule**** to create a new rule.



3. **Define the Global Rule:** Create two rules. One for table names and one for column names.
   - In the **Rule Name** field, enter a descriptive name for the rule (e.g., "Convert Table/Column Names to Support Special Characters").

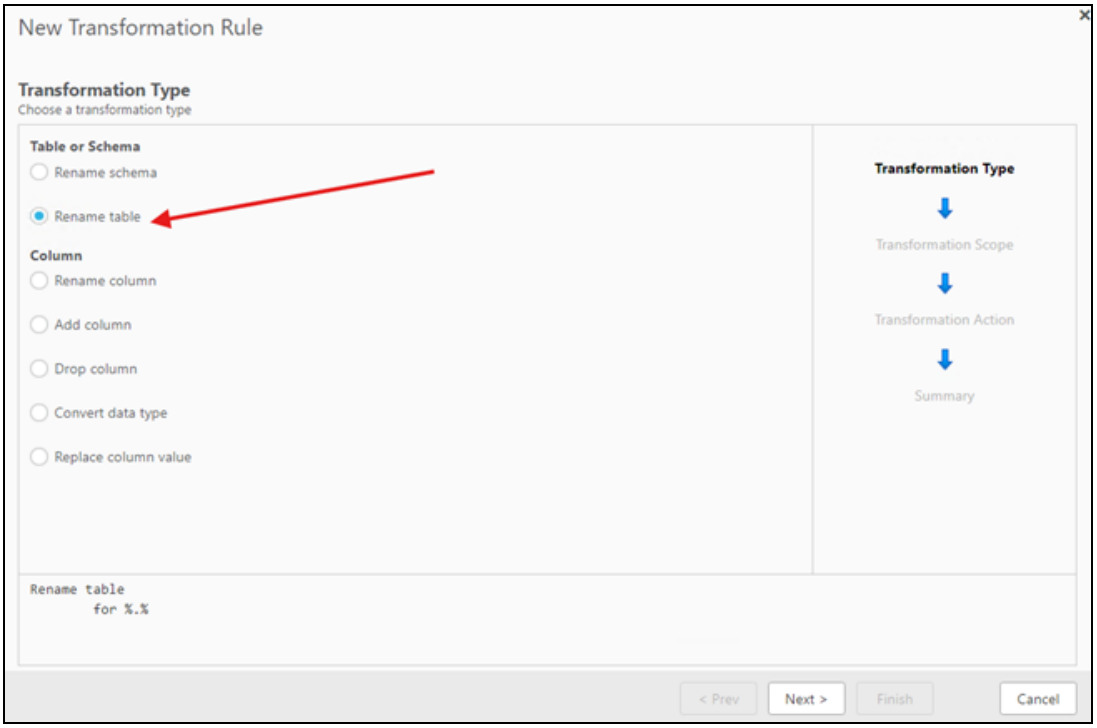
- In the **Rule Type** dropdown, select **Transformation**.



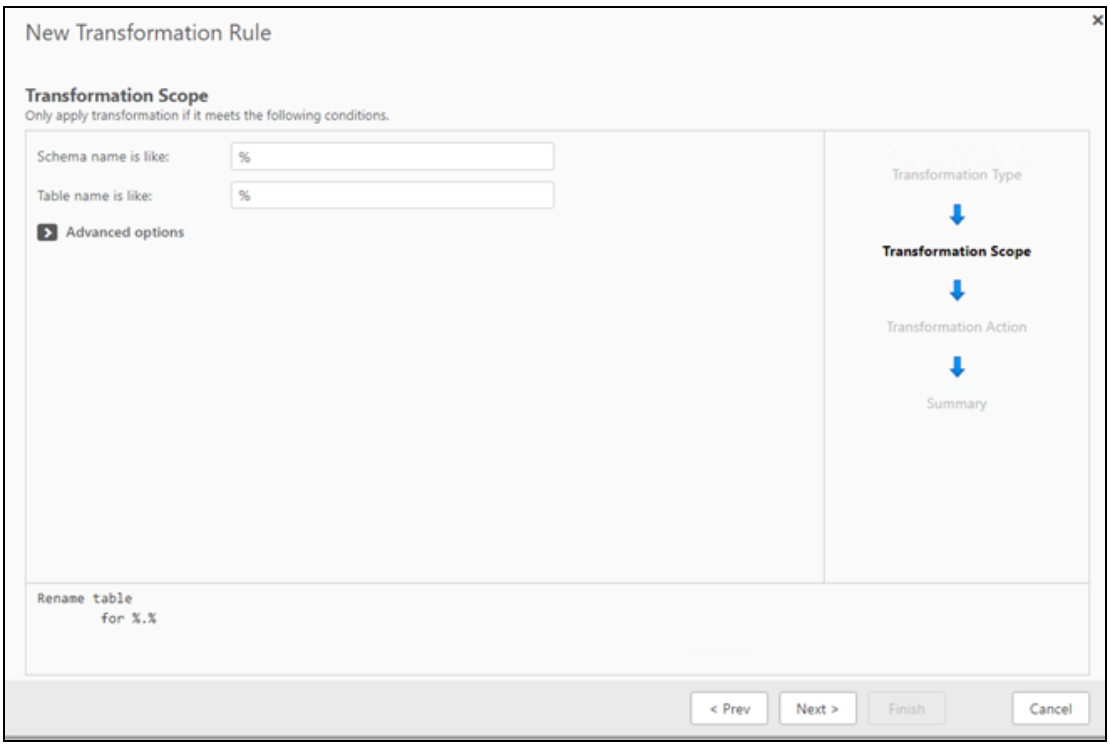
4. **Configure the Transformation:**

   i. In the transformation settings, add a new transformation rule.

   ii. Set the **Transformation Type** to **Rename table/column**.



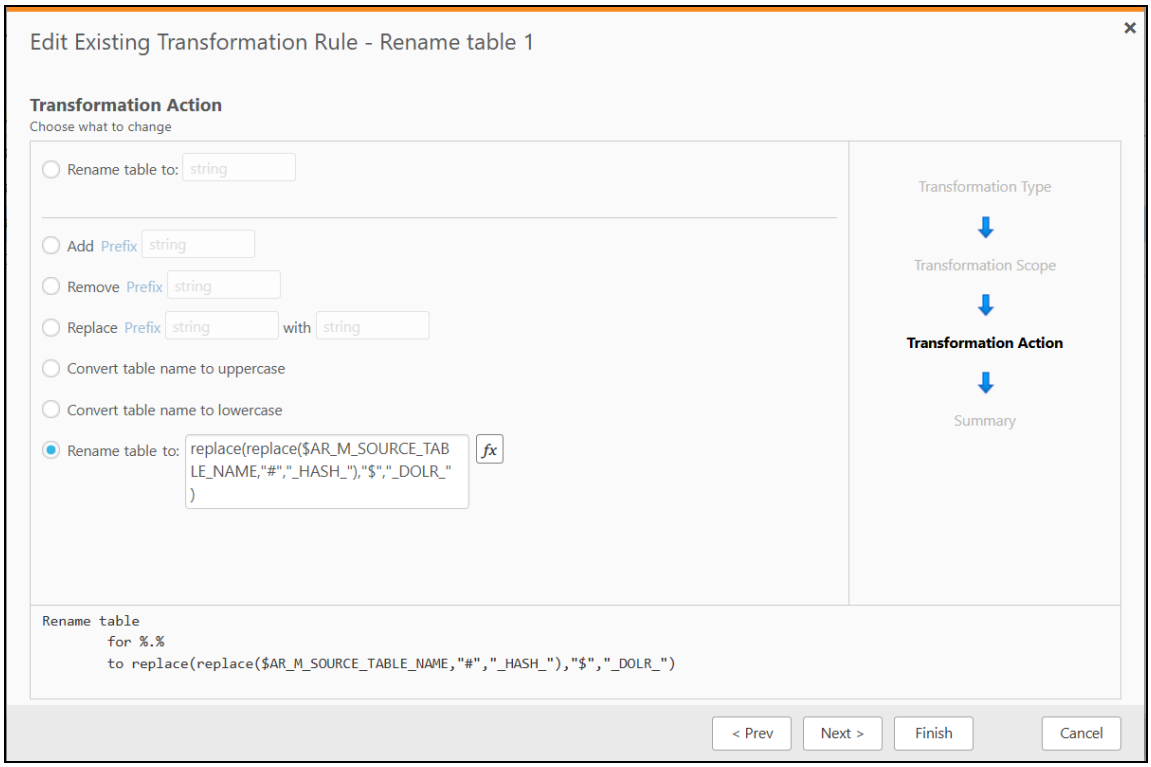
   iii. Click **Next**.

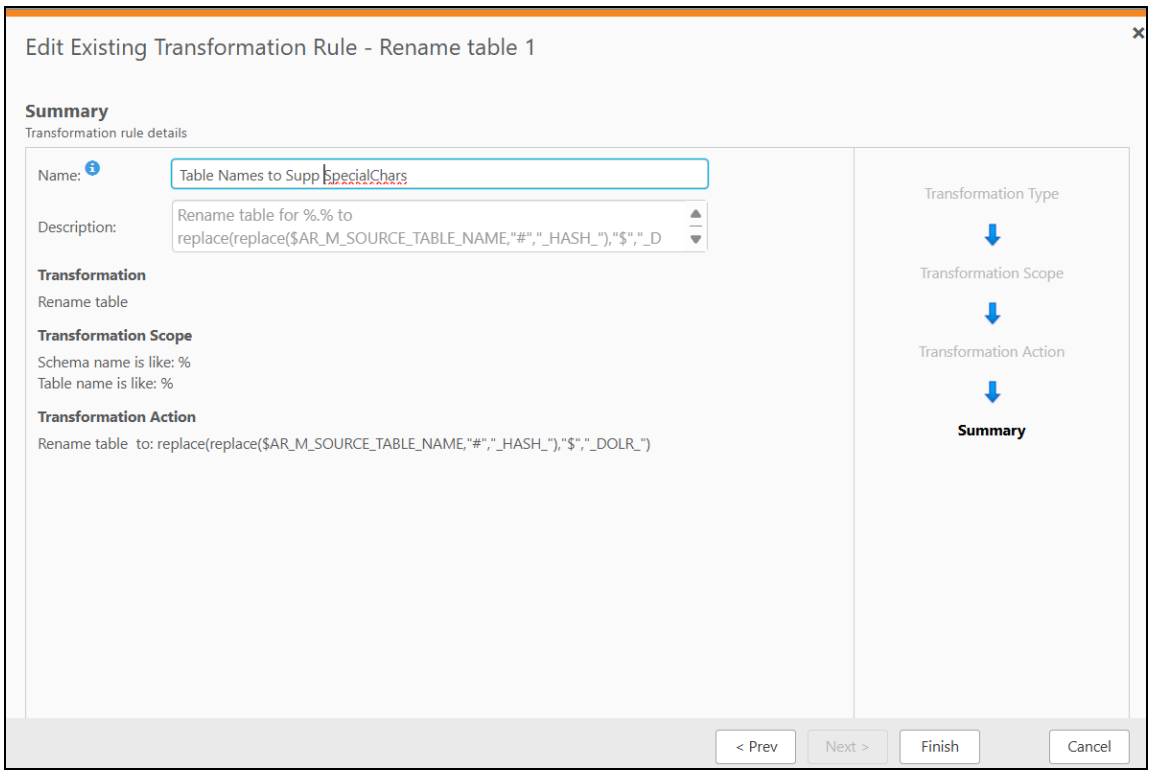


iv. In the **Rename Table/Column to** field, use a function to convert the table/column name to support special characters. You can use the following expression:

i. **Column Name :** replace(replace($AR_M_SOURCE_COLUMN_NAME,"#","HASH"),"$","DOLR")

ii. **Table Name :** replace(replace($AR_M_SOURCE_TABLE_NAME,"#","HASH"),"$","DOLR")





5. **Apply the Global Rule**.

- Ensure the global rule is enabled.
- Save the global rule settings.

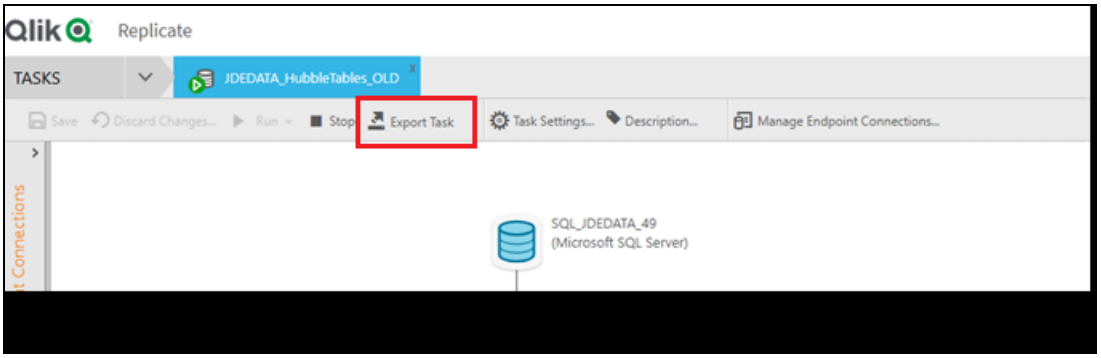
6. **Verify the Changes:** Run a task to verify that the table/column names are being retained if the table/column has special characters.

   By following these steps, you can successfully retain table/column names using global rules in the Qlik Replication tool.

Migrating Existing Qlik Tasks to Kafka

Back up your existing Qlik tasks

- If you are using the accelerator for the first time, create source and target endpoints (Kafka). Then, create a new task with all the required tables.
- If you have previously used earlier versions of the accelerator, you need to export all your existing tasks before upgrading the accelerator as part of the standard process.



Replace Existing Actian Vector Target Endpoint Connection with Kafka

After completing the installation and ensuring all components are running, create a Kafka endpoint as follows. In previous versions, different target endpoints were created based on the schemas of Actian Vector. Now, only one Kafka endpoint is needed.

- Kafka Broker List:

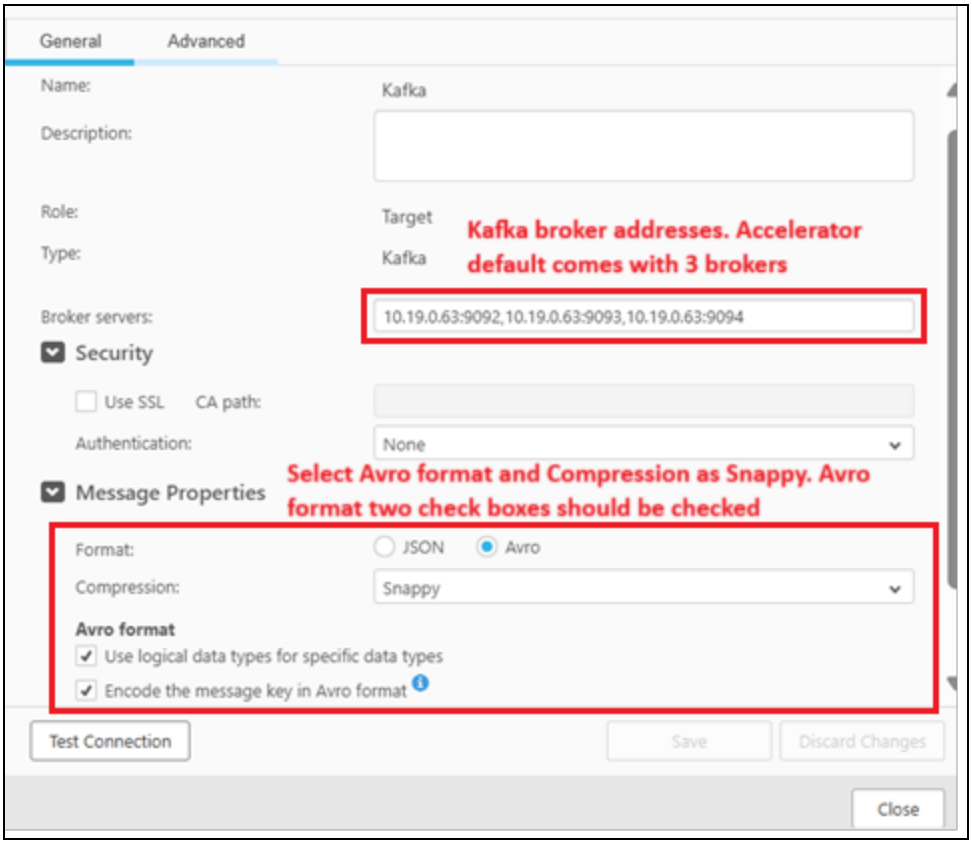
  Example: localhost:9092, localhost:9093, localhost:9094

- Topic Naming Convention:

  Example: `source.schema.table`

- Message Properties:

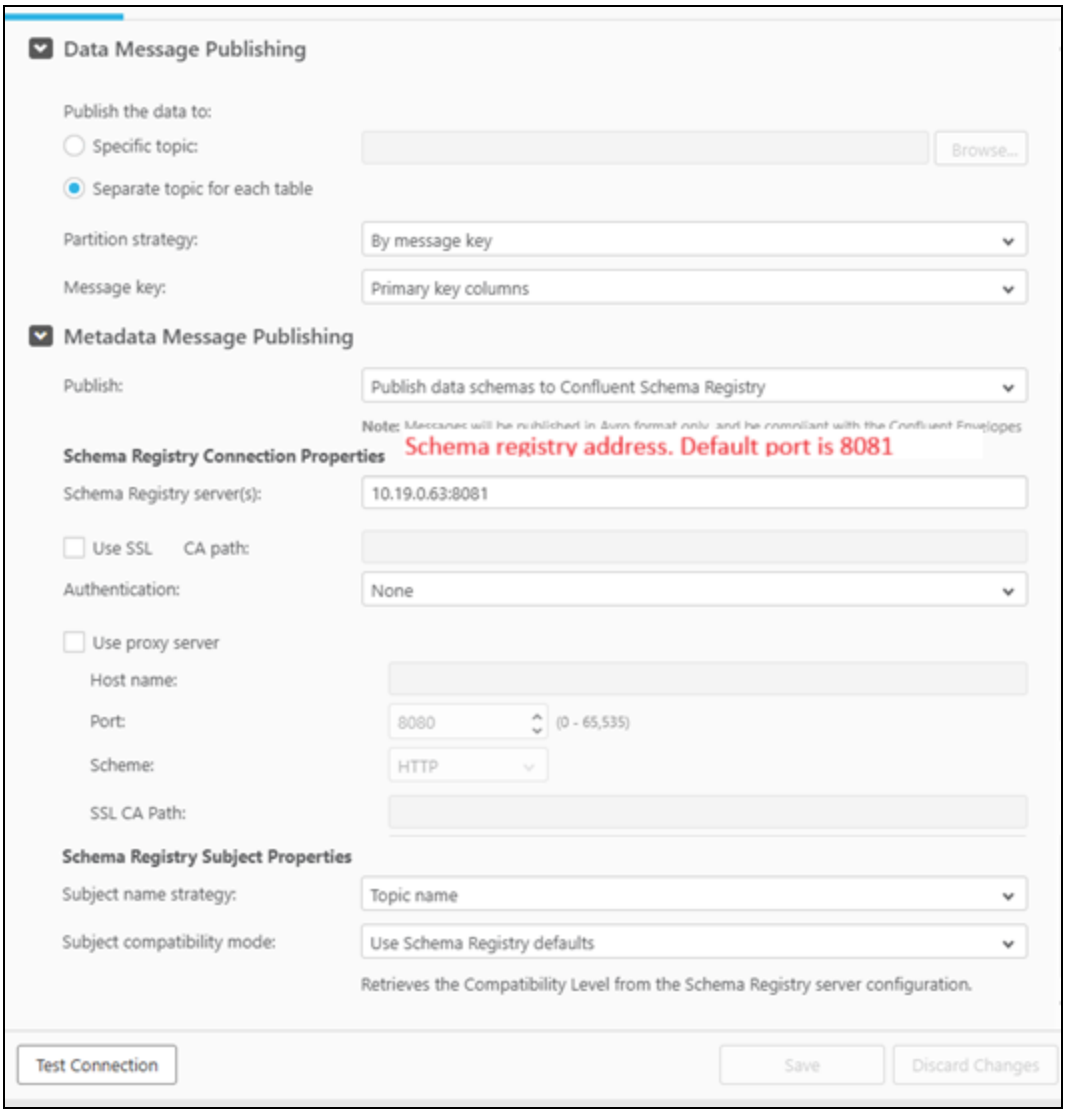
  1. Set **Avro** format.
  2. Set **Compression** to **Snappy**.
  3. Select **Use Logical data types for specific data types** in **Message Properties**.
  4. Select **Encode the message key in Avro format** in **Message Properties**.



- Data Message Publishing:

  1. Select **Separate topic for each table**.
  2. Select **By message key** as the **Partition strategy**.

3. Select **Primary key columns** as the **Message key**.

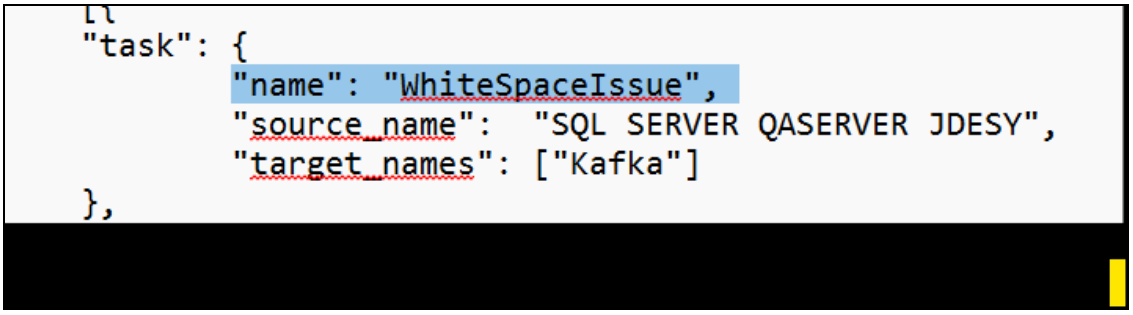


- Metadata Message Publishing:
  1. Select **Publish data schemas to Confluent Schema Registry** for **Publish label**.
  2. Input the **Schema Registry server(s)** (for example, localhost:8081).
- Schema Registry Subject Properties:
  1. Select **Topic name** for **Subject name strategy**.
  2. Select **Use Schema Registry defaults** for **Subject compatibility mode**.
- Test and Save:

1. Once all the required information is entered, click on **Test Connection**.
2. If the test connection is successful, click **Save**. Otherwise, fix the errors and save the connection.

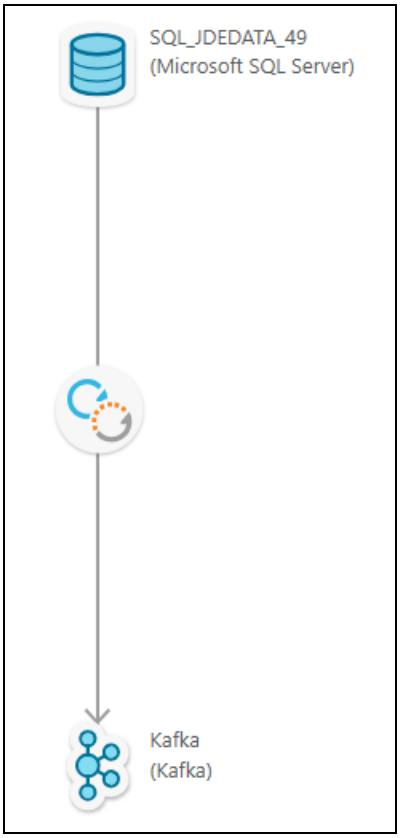
Import Existing Qlik Tasks

1. **Import Tasks:**
   - Once the Kafka endpoint is ready, import all previously exported tasks.
   - Before importing, change the task name if you want to keep both existing tasks and tasks with Kafka as the target endpoint.



2. **Change Target Endpoint to Kafka Endpoint:**

   After successfully importing all tasks, navigate to each task, remove the existing endpoint, and replace it with the Kafka endpoint. Save the task.

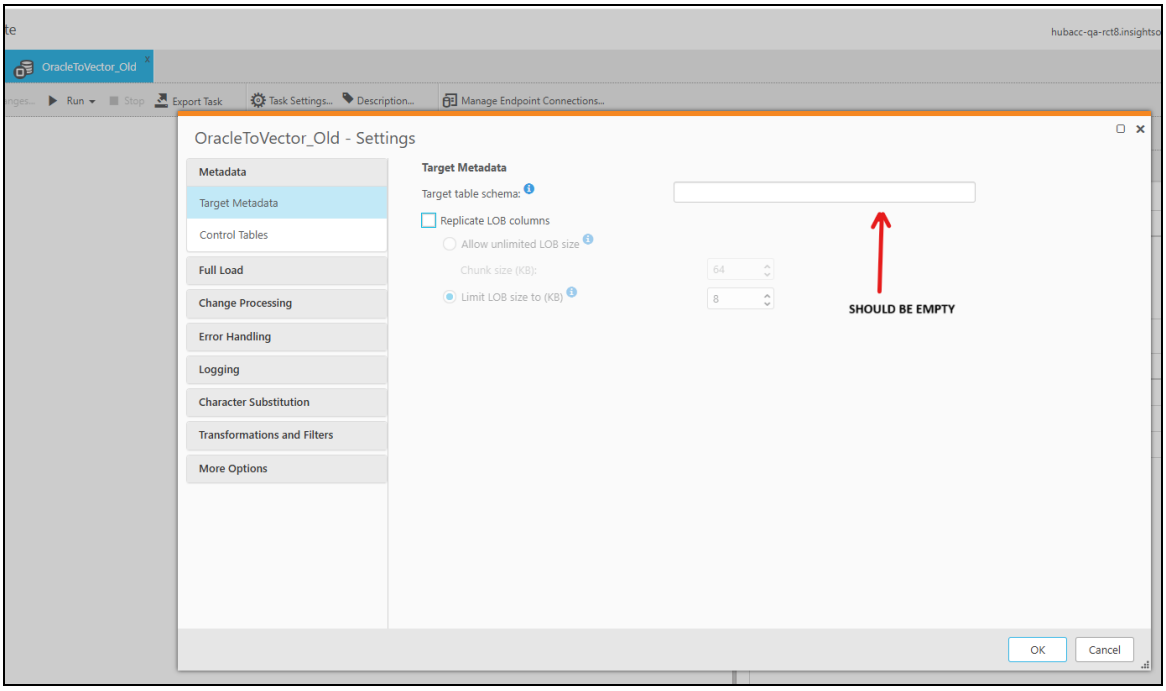


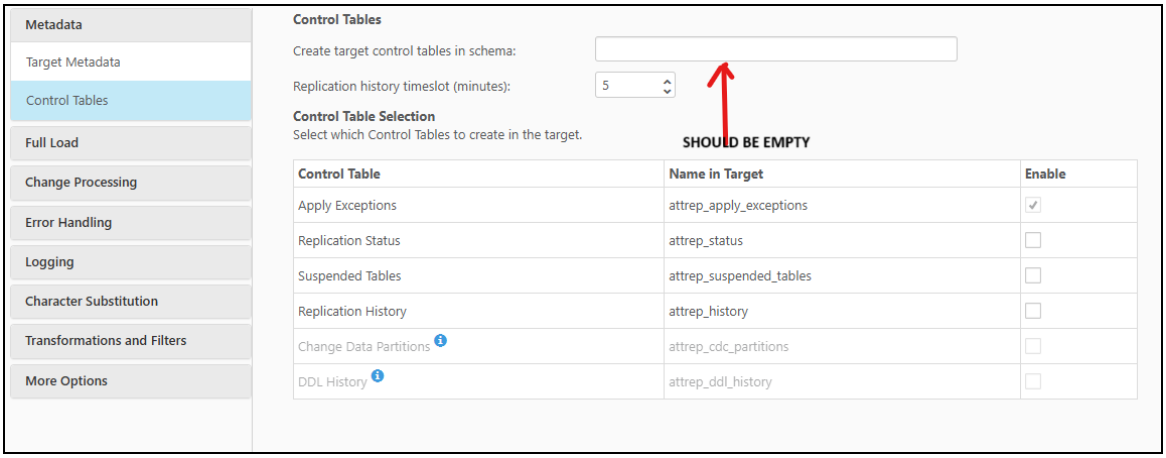
3. **Apply Global Transformations:**

   Add global transformations to include table names and column names with placeholders to support special characters via Global Rules in Qlik Replication Tool. Save the task. Refer to Support Special Characters in Table and Column Names via Global Rules in Qlik Replication.

4. **Remove Actian Vector Metadata:**

   After importing the task, open **Task Settings** and clear the target table schema details.





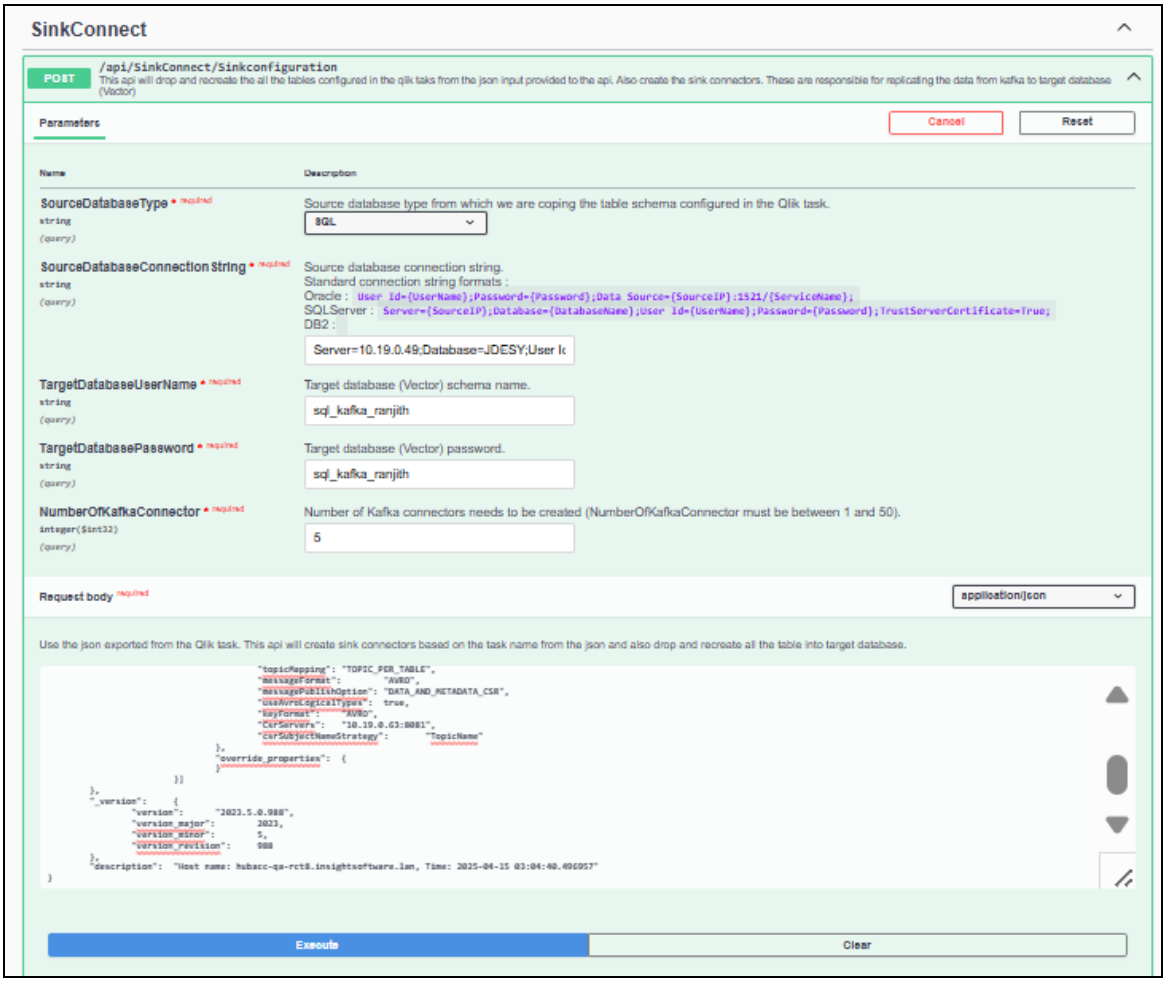
Create Sink Connectors

1. **Create Sink Connector:**

   - Use the Qlik task JSON file exported before the migration to create a sink connector, which replicates data from Kafka to the target Actian Vector database.

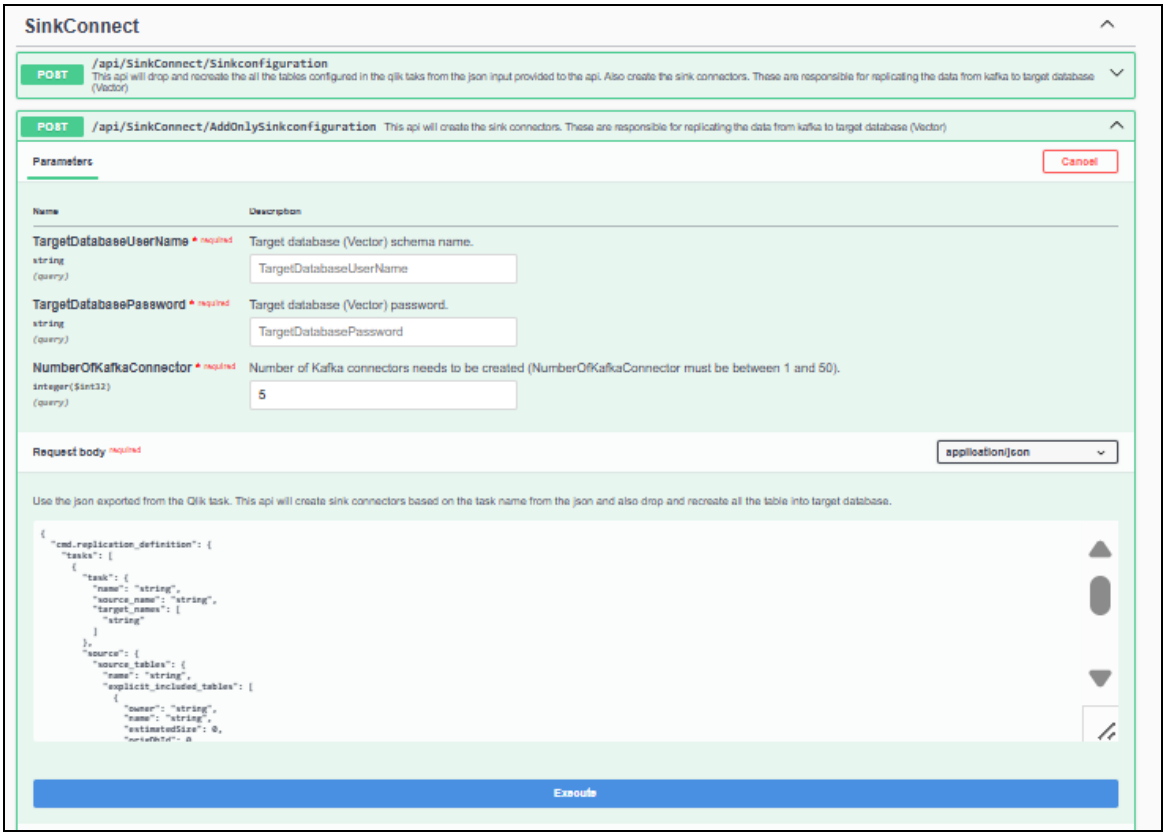
   - Access the sink connector API at http://{IP Address}/Swagger/index.html.

2. **Create Sink Connector with Table Creation:** If creating a new replication (target tables are not available), use the following endpoint to create target tables along with sink connectors:



- **SourceDatabaseType:** Select the source database type (SQL, ORACLE, DB2).
- **SourceDatabaseConnectionString:** Connection string of the source database.
- **TargetDatabaseUserName:** Target database username.
- **TargetDatabasePassword:** Target database password.
- **NumberOfKafkaConnector:** Number of Kafka connectors to be created (default 5).
- **Request Body:** Post the Qlik task JSON file to create the target tables and sink connectors.

3. **Create Sink Connector without Table Creation:** If resuming an existing process (full load already completed), use the following endpoint to create Kafka connectors without dropping and recreating tables:

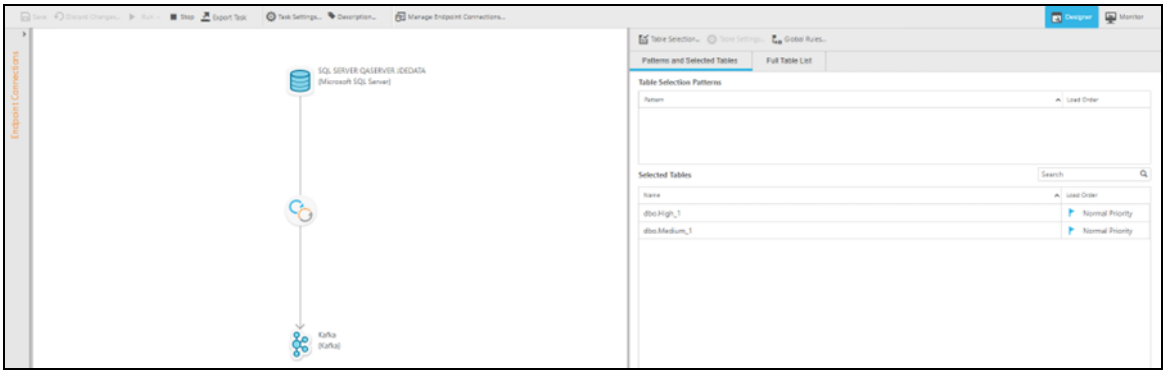


- **TargetDatabaseUserName:** Target database username.
- **TargetDatabasePassword:** Target database password.
- **NumberOfKafkaConnector:** Number of Kafka connectors to be created (default 5).
- **Request Body:** Post the Qlik task JSON file to distribute tables into different Kafka connectors for replication.

4. **Run Qlik Task:** Once the target table schema is ready, run the Qlik task to reload the target or resume processing.

Start the Qlik Task

1. Ensure target tables are created, sink connector configurations are posted, and the task is exported and pasted into the Sink API Manager.
2. Start the Qlik task to begin data replication.

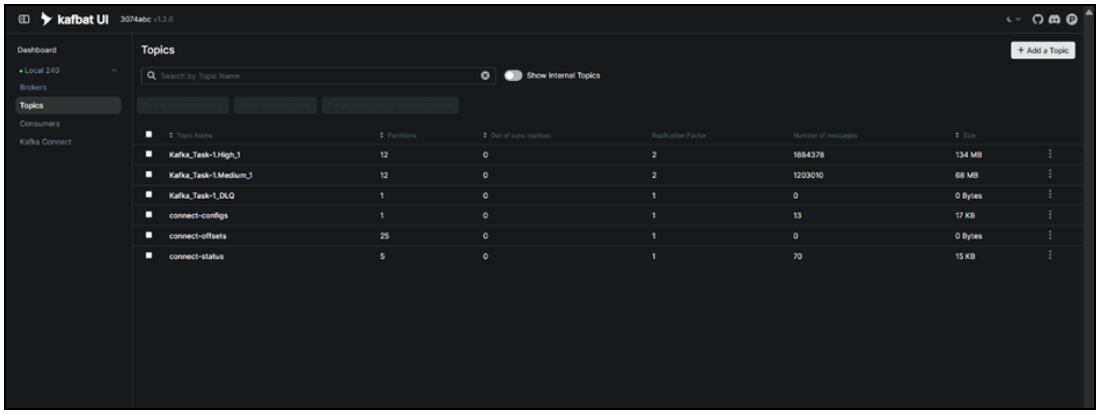


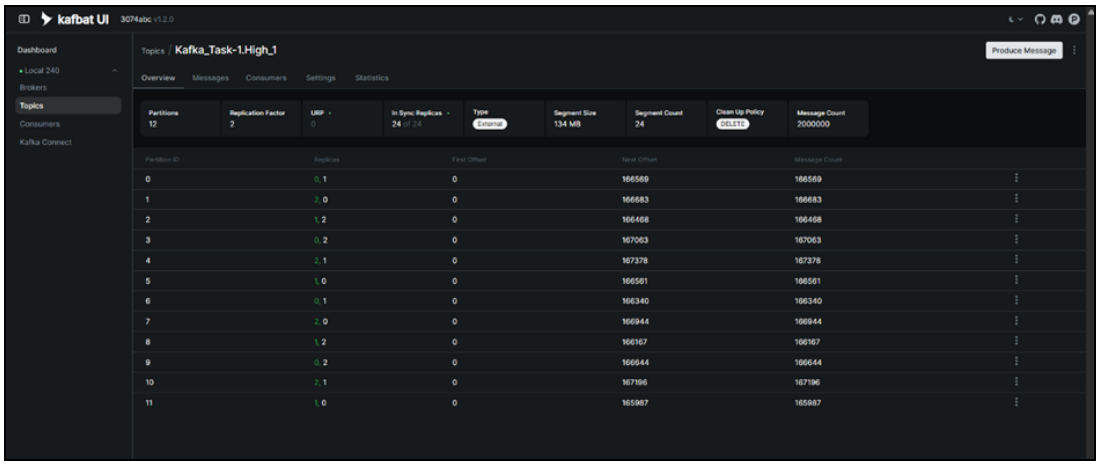
> **Note:** Exporting the task correctly is crucial for replication to work smoothly.

Monitor Replication in Kafka UI

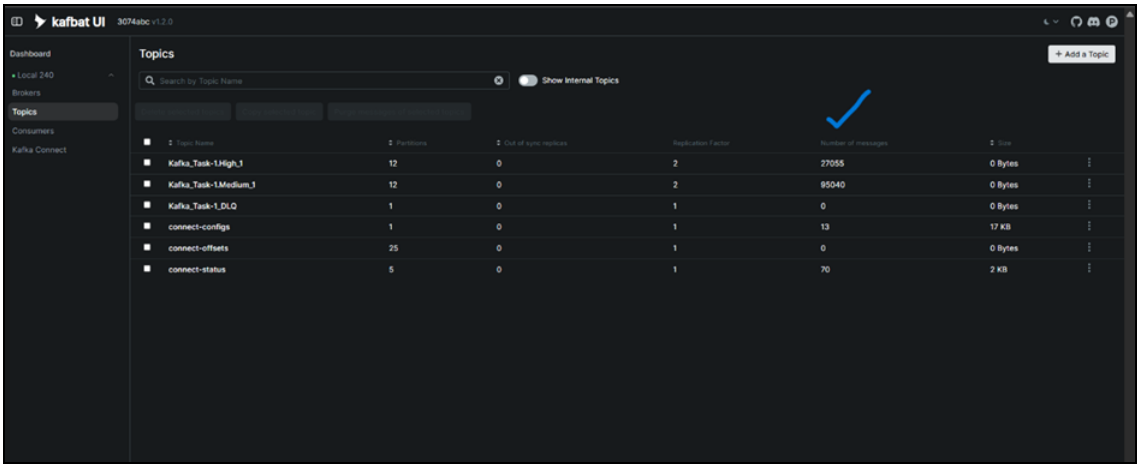
- **Monitor Progress:**

1. Use Kafka UI to monitor replication progress.
2. Check topic-wise message flow and consumer lag.
    - **Topic-wise message flow:** Each topic corresponds to a table. You will see the number of messages per topic.

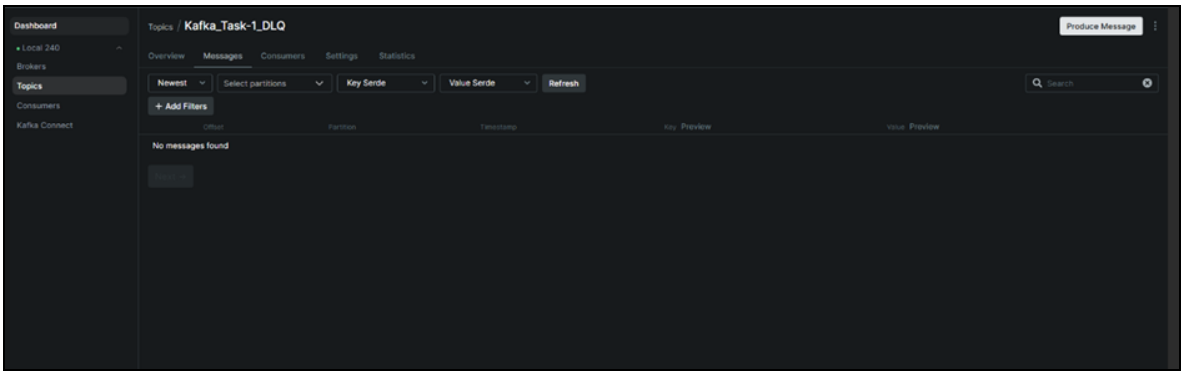




- **Consumer Lag:**
  - If consumer lag = 0, all records in that topic have been moved to the target database.
  - If lag > 0, there are still records pending processing.



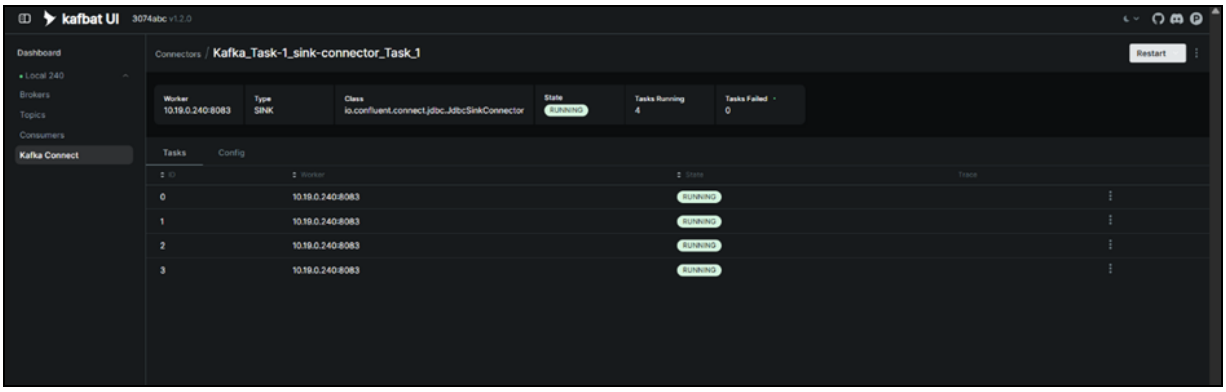
- **Failure Handling:** If replication failures occur, check DLQ topics for failed records and exceptions.



Connector Failures

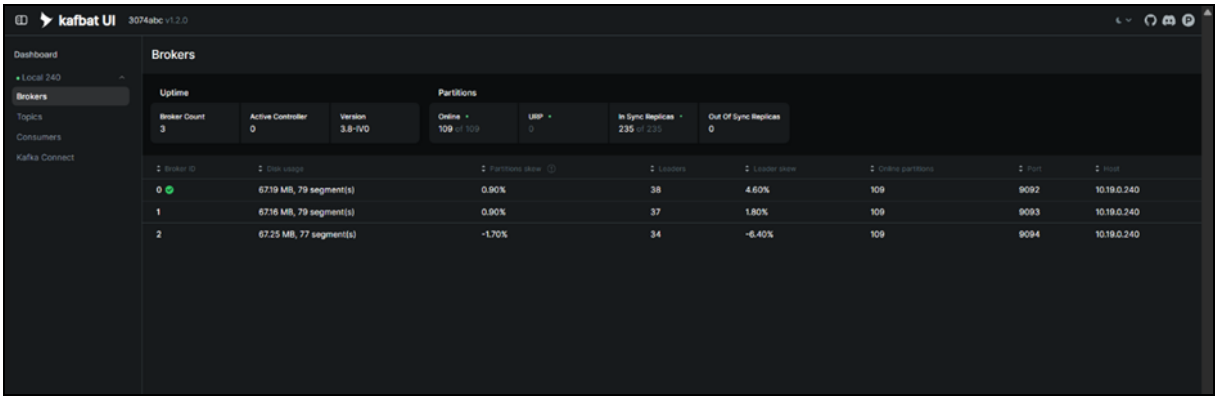
- Check the **Sink Connector status** in the **Kafka Connect UI** or via the API.
- If the connector has failed or is in a degraded state, replication to the target database will stop.
- Review the **connector logs** for:

- Schema mismatches
- Network timeouts
- Target database constraints or errors
  - Restart or redeploy the connector after resolving the issue.



Monitor Broker Health and Disk Usage

To ensure stability, monitor broker **health status** and disk space consumption.



> **Note:** Overloaded brokers or disks may lead to increased lag or replication failures.

Recommendations

Run the following source and target-specific SQL queries to monitor ongoing data replication.

Oracle:

```
SELECT SUM(row_count) AS total_rows
```

```
FROM (
```

```
SELECT COUNT(*) AS row_count FROM R12QA2_REP.AP_AE_HEADERS_ALL_1
```

```
UNION ALL
```

```
SELECT COUNT(*) FROM R12QA2_REP.SAMPLE_ALL_DATATYPES
```

```
);
```

MSSQL:

```
SELECT SUM(p.rows) AS total_rows
```

```
FROM sys.tables t
```

```
JOIN sys.partitions p ON t.object_id = p.object_id
```

```
WHERE t.name IN ('table1', 'table2') AND p.index_id IN (0,1);
```

Actian Vector:

```
select sum(num_rows)
```

```
from iitables
```

```
where table_name in () and table_owner like 'kafkalatest%'
```

Upgrade Hubble Accelerator

There are two main ways to upgrade Hubble Accelerator:

- "Upgrading/Migrating from Accelerator Versions Prior to 3.x" below
- "Upgrading from a 4.x Version" on page 80

Upgrading/Migrating from Accelerator Versions Prior to 3.x

1. On the old **Hubble Accelerator 2.x** machine:
   i. Stop the Vector Service:

   ```
   /etc/init.d/S60ingresVW stop
   ```

   ii. Stop the Replication Service:

   ```
   /etc/init.d/S65areplicate stop
   ```

   iii. Back up the current data disk (disk mounted on `/mnt/ybdata`).

   iv. Detach the data disk.

2. On the new **Hubble Accelerator 5.x** machine:
   i. Install the Hubble Accelerator 5.x (see "Install Hubble Accelerator" on page 50).

   ii. Attach the data disk from the old Hubble Accelerator to the new machine:

i. Run the following command::

```
blkid
```

You should see output similar to this. Locate the device with **LABEL="ybdata"** as it contains the old data:

```
/dev/xvda2: UUID="4a1c93d9-eb47-4f96-9f3d-920e52dc8cca"
TYPE="xfs" PARTUUID="078b129f-9e3b-4665-8871-232657ae682c"
```

```
/dev/xvdb1: LABEL="ybconfig" UUID="213187cf-5d80-4623-b4fc-53fa5153cd35" TYPE="ext2" PARTUUID="897ae9ad-83e2-4e7c-8dd8-686cd0d36837"
```

```
/dev/xvdb2: LABEL="ybdata" UUID="9a7fc249-8fd7-40ed-84a6-6789c8346f78" TYPE="ext3" PARTUUID="4c5d92db-4cf2-40f6-b0f1-c28d1f67652d"
```

```
/dev/xvda1: PARTUUID="e7184259-29aa-42a5-b24d-1143dc10e5d8"
```

ii. Stop the Vector Service:

```
systemctl stop actian-vectorVW
```

iii. Stop the Replication Service:

```
/etc/init.d/areplicate stop
```

iv. Edit `/etc/fstab`

```
vi /etc/fstab
```

Add the following lines:

```
LABEL=ybdata  /mnt/ybdata auto defaults 00
```

/mnt/ybdata/ingres/ckp	/opt/Actian/VectorVW/ingres/ckp	none	bind,_netdev	0 0
/mnt/ybdata/ingres/data	/opt/Actian/VectorVW/ingres/data	none	bind,_	0

			netdev	0
/mnt/ybdata/ingres/dmp	/opt/Actian/VectorVW/ingres/dmp	none	bind,_netdev	0 0
/mnt/ybdata/ingres/jnl	/opt/Actian/VectorVW/ingres/jnl	none	bind,_netdev	0 0
/mnt/ybdata/ingres/log	/opt/Actian/VectorVW/ingres/log	none	bind,_netdev	0 0
/mnt/ybdata/ingres/work	/opt/Actian/VectorVW/ingres/ work	none	bind,_netdev	0 0
/mnt/ybdata/areplicate/data	/opt/attunity/replicate/data	none	bind,_netdev	0 0

5. Refresh the mounts to implement the changes on `fstab`:

```
mount -a
```

6. Change user ownership for actian and attunity users on the mounted file-system folders by running:

```
chown -R actian:actian /mnt/ybdata/ingres
```

```
chown -R attunity:attunity /mnt/ybdata/areplicate/
```

7. Remove the old db configuration:

```
rm /mnt/ybdata/ingres/data/vectorwise/vectorwise.db.conf
```

8. Create a new db configuration (where 5.x.x is the Hubble Accelerator version):

```
/opt/insightsoftware/hubble-accelerator/5.x.x/vector/
configurevector.sh
```

   For example:

```
/opt/insightsoftware/hubble-accelerator/5.0.1/vector/
configurevector.sh
```

9. Reboot.

Upgrading from a 4.x Version

1. Back up your data.
2. Stop the Vector Service:

```
systemctl stop actian-vectorVW
```

3. Stop the Replication Service:

```
/etc/init.d/areplicate stop
```

4. Stop the Hubble Appliance API Service:

```
systemctl stop hubble-applianceapi
```

5. Install the new version of Hubble Accelerator (follow the installation instructions in "Install Hubble Accelerator" on page 50).

Downgrade Hubble Accelerator

If there are issues with Hubble Accelerator **version 5.0.x** and a decision is made to downgrade to version **4.3.x**, follow these steps:

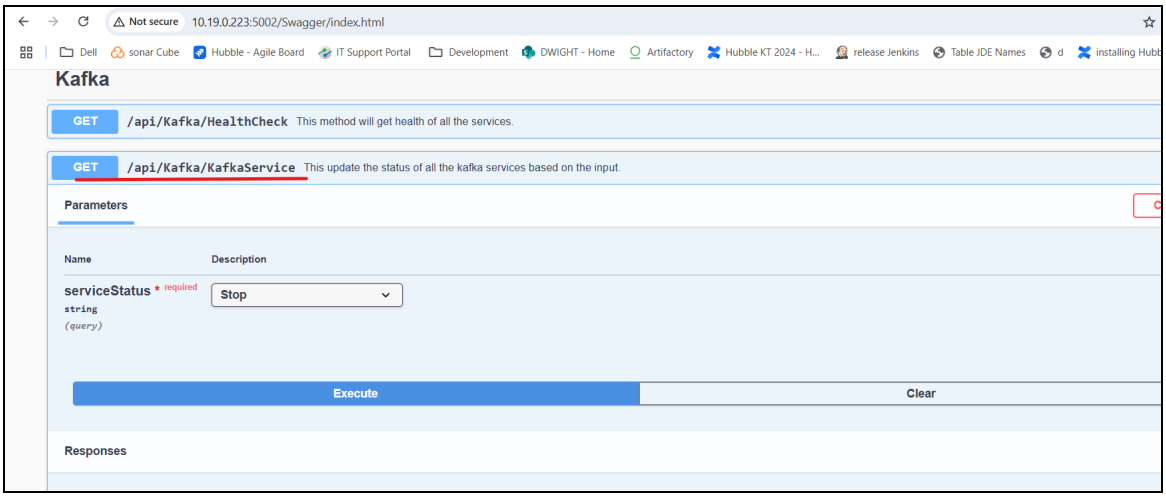
1. Perform the steps provided in the Uninstall Hubble Accelerator topic.
2. Once the uninstallation of Hubble Accelerator v5.0.x is successful, uninstall the Vector Package by running the following command:

   ```
   yum remove actian-vector.x86_64
   ```

3. Follow the installation steps for Hubble Accelerator v4.3.x available on the Hubble Product page - Hubble Accelerator User Guide for Version 4.3x .

Uninstall Hubble Accelerator

1. Navigate to Sink Connector API and run `Get - /api/Kafka/KafkaService` with **Stop** Status.



2. Stop Sink Connector Service using:

```
sudo systemctl stop hubble-sinkconnector
```

3. Stop the Replication Service:

```
/etc/init.d/areplicate stop
```

4. Stop the Vector Service:

```
systemctl stop actian-vectorVW
```

5. Stop the Hubble Appliance API Service:

```
systemctl stop hubble-applianceapi
```

6. Run the uninstall script, which will uninstall Vector, Replication and the ODBC drivers used to connect to replication sources

> **Note:** Replace 5.x.x with the actual version you wish to uninstall.

```
/opt/insightsoftware/hubble-accelerator/5.0.x/uninstall.sh
```

7. Uninstall the Hubble Accelerator RPM package.

> **Note:** Replace 5.x.x with the actual version you wish to uninstall.

```
yum remove hubble-accelerator-5.0.x-xx.exx.x86_64
```

> **Note:** Uninstalling will not affect the data for Vector and Replication, it will remain intact.

Hubble Accelerator Shutdown Steps

This section contains the instructions for safely shutting down Hubble Accelerator. The sequence should be:

1. Shut down Qlik Replication.
2. Disable new Connections to the Accelerator Database that could write transactions.
3. Propagate Accelerator In-Memory Changes and Condensing the Log.
4. Roll Transaction Log.
5. Shut down Actian Vector.

Shut Down Replication

Caution: Propagating the transaction log to disk will fail if processes are making database changes. As such the replication service must be stopped before attempting to propagate the log. Refer to Propagate the In-Memory Changes and Condense the Log.

1. Using PuTTY or your preferred terminal window, connect to the accelerator server and login as the root user.
2. Change directory to `/opt/attunity/replicate/bin`.

   ```
   # cd /opt/attunity/replicate/bin
   ```

3. Switch user to the attunity replication user.

   ```
   # su attunity
   ```

4. Get a list of all the running replication tasks.

   ```
   # ps -ef | grep repctl
   ```

```
[root@yb~]# ps -ef | grep repctl
attunity  2349     1  0 May24 ?        00:00:00 /opt/attunity/replicate/bin/repctl service start
attunity  2350  2349  0 May24 ?        00:43:51 /opt/attunity/replicate/bin/repctl service start
attunity  6660  2350  1 Nov20 ?        05:57:29 repctl reptasksrv PRODDTA 127.0.0.1:3552 2350
root     12585 12579  0 14:44 pts/0    00:00:00 grep repctl
attunity 22588  2350  0 Nov06 ?        00:18:24 repctl reptasksrv JDE812 127.0.0.1:3552 2350
attunity 23203  2350_ 0 Nov06 ?        01:44:48 repctl reptasksrv CUSTOM_TABLES 127.0.0.1:3552 2350
```

Caution: The running replication tasks are the processes that start **`repctl reptasksrv <TASK_NAME>`**. In the example above there are three running tasks:

- PRODDTA
- CUSTOM TABLES
- JDE812

5. For each task returned by the above, run the following command to get the status of the task and if it is running to stop the replication task. (You can also stop the tasks in the Qlik Console, but you should check the status as instructed below)

```
# ./repctl connect\; gettaskstatus <TASK_NAME>\; disconnect
```

```
# ./repctl connect\; stoptask <TASK_NAME>\; disconnect
```

> **Caution:** The stoptask command used above will signal the task to stop, if the task is currently showing latency, then the task will not stop immediately but will stop after processing the changes that are causing the latency. This will allow you to restart the task with the resume feature, this means you do not need to reload the data. If you run the stoptask command and manually kill the associated replication process, then you will not be able to use the resume command to restart the task, you will have to restart the task using the reload functionality in order to ensure data consistency. All the tasks must be stopped before moving on to Step 6 and 7.

6. Exit from attunity user to return to the root user:

```
# exit
```

7. Change directory to `/etc/init.d`:

```
# cd /etc/init.d
```

8. Shut down replication service. This will prevent users from connecting remotely using Qlik Console and starting new tasks.

```
# ./areplicate stop
```

9. Confirm that the services has stopped by checking that there are no `repctl` processes running:

```
# ps -ef | grep repctl
```

Disable new Connections to the Accelerator Database

> **Note:** According to the instructions for Shutdown Replication (see Hubble Accelerator Shutdown Steps), you must ensure that no transactions are added while propagating the log, as this can cause the process to fail. Besides stopping replication, you also need to prevent new SQL sessions from starting on the accelerator database during this time. Refer to the section on "Propagating Accelerator In-Memory Changes and Condensing the Log" on page 89 for more details.

1. Connect to Hubble Accelerator server.
2. Switch user to the Actian database user.

```
# su actian
```

3. Use `iinamu` to get the server number that you will need to pass to `iimonitor`.

```
# iinamu
Actian Vector NAME SERVICE MANAGEMENT UTILITY --

-- Copyright (c) 2009 Actian Corporation
 IINAMU> showINGRES    *    41271 (sole server)
 IINAMU> quit
```

4. Run `iimonitor` using the number of the "sole server" returned by the above iinamu command.

```
# iimonitor 41271
IIMONITOR>
```

5. Stop new SQL sessions from starting.

```
IIMONITOR> set server closed
User connections now disabled
IIMONITOR>
```

6. Wait a few minutes to allow user sessions to naturally terminate and then get a formatted list of all the active sessions. The following only shows the details of a single session but will give you a feel for what to expect to see.

```
IIMONITOR> show sessions formatted
Session 0000000007023C80:1399772928 (jde810a    ) cs_state: CS_
EVENT_WAIT (BIOR) cs_mask: CS_INTERRUPT_MASK,CS_NOXACT_
MASK OS_tid: 31151
```

```
DB Name: db         (Owned by: root     ) User: jde810a
(jde810a     ) Session started at 3-May-2016 12:04:56 as user
jde810a
Terminal: batch Group Id:
Role Id:
Application Code: 00000000     Current Facility: CLF (00000001)
Client user: Edward
Client host: edward0912 Client tty:
Client pid:
Client connection target: db
Client
information:     user='Edward',host='edward0912',conn='db',
server='57468',session='1c250c0'
Description:
Query:
Last Query: open ~Q cursor for SELECT CVCRCD, CVCDEC, CVDL01
FROM jde810a.F0013 for readonly
Session 000000000714BFC0:-566962432 (integration_test     ) cs_
state: CS_EVENT_WAIT (BIOR) cs_mask: CS_INTERRUPT_MASK,CS_
NOXACT_
MASK OS_tid: 4031
...
```

7. Shut down each non-terminated session using the following command.

> **Note:** The sessionID that needs to be passed is the hexadecimal number that precedes the colon. Thus for the above example, Session 0000000007023C80:1399772928, the session identifier is 0000000007023C80.

```
IIMONITOR> remove 0000000007023C80
Session 0000000007023C80 removed
```

8. Repeat steps 6 and 7 until there are no non-administrator sessions running.

9. Re enable connections to the database. This is to ensure that you can run the sql command in the next section to allow the in-memory changes to the database to be propagated and condensed.

```
IIMONITOR> set server open
User connections now allowed
```

10. Exit `iimonitor`leaving the server closed.

```
IIMONITOR> quit #
```

The `iimonitor` command is an extremely useful tool for monitoring and understanding what the accelerator database is doing. For more information on this tool refer to Appendix A of the Actian User Guide (Vector 6.3 guide).

Propagating Accelerator In-Memory Changes and Condensing the Log

This section covers instructions on how to manage in-memory changes and the transaction log for the Accelerator database. It includes steps for monitoring the size of in-memory changes, propagating these changes to the database, and condensing the log to ensure optimal performance and stability.

Introducing in-memory changes and the transaction log

To ensure high performance, the columnar database driving Accelerator performs most tasks in memory. Transactions are only written to disk when the system is idle. The Actian system tries to propagate transactions and condense the log periodically, but this can fail if the system is constantly processing transactions.

As an Accelerator administrator, you need to monitor the volume of in-memory changes to prevent them from growing too large. When stopping the accelerator, for example, for an upgrade, make sure all database transactions are written to disk. Failure to do so can cause startup delays as unpropagated in-memory changes are read into memory, or the system may fail to start if the in-memory changes exceed the allowed memory size for the Actian process.

This is a two-stage process:

1. **Propagate in-memory changes:** Write the in-memory changes to the database.

2. **Condense the log:** Remove the in-memory changes now written back to the database. For more details, refer to Appendix 1: Additional Information Relating to Log Propagation.

> **Note:** Propagating the in-memory changes and condensing the log can take some time. This will fail if in-memory changes are made via queries while this is running. As such you must ensure that the steps in the previous sections to effectively stop all queries running on the database must be carried out to ensure that this will complete successfully. This activity, given the time that it is likely to take must be planned at a time when it is least disruptive to the users of the system.

How to Monitor the size of In-Memory Changes

The command **`vwinfo -M <DATABASE>`** will return what tables are accumulating in memory change and the size of these.

Thus on a typical Accelerator the steps will be:

1. Connect to Hubble Accelerator server.
2. Switch user to the Accelerator database user.

```
# su actian
```

3. Run the above vwinfo command.

```
# vwinfo -M db
```

How to Propagate the In-Memory Changes into the Log and Condense the Log

Proceed as follows:

1. Using PuTTY or your preferred terminal window, connect to Hubble Accelerator server.
2. Switch user to the Accelerator database user.

```
# su actian
```

3. At this point you need to decide if you want to propagate specific tables where the in-memory changes are particularly large or if you want to do this for all tables in the schema. As noted above, propagating changes can take some time and as the administrator you need to know how large a downtime window your business can tolerate and plan accordingly.

   To propagate the in-memory changes for specific tables you need to run the command **`echo "call vectorwise (combine '<TABLE>')\g"`** for each table.

```
echo "call vectorwise (combine 'jde.f0911')\g" | sql db
```

   Alternatively, to propagate the in-memory changes for all tables in the database run the following command.

```
echo "call vectorwise (combine)\g" | sql db
```

4. After propagating the changes to all the desired tables in a given schema the following command should be run to condense the log.

```
echo "call vectorwise (condense_log)\g" | sql db
```

Roll Over the Database Log File

The `vectorwise.log` file, stored in **/opt/Actian/VectorVW/ingre/files**, accumulates all log messages for the accelerator database and this can grow quite large. This should be regularly rolled over to keep the size of the log manageable.

How to check the size of the vectorwise.log File

Run the following command as the Actian user:

```
# ls -lh /opt/Actian/VectorVW/ingres/files/vectorwise.log
```

How to roll the vectorwise.log File

If the file is getting large it is advisable to roll the log file. The following command should be run as the Actian user.

```
# echo "call vectorwise(vwlog_rotate)\g" | sql db
```

The expected output would be something along the lines of:

```
TERMINAL MONITOR Copyright 2015 Actian Corporation
Vector Linux Version VW 4.2.2 (a64.lnx/228) login
Fri Nov  3 03:24:47 2017
Enter \g to execute commands, "help help\g" for general help,
"help tm\g" for terminal monitor help, \q to quit

continue
* Executing . . .

continue
*
Your SQL statement(s) have been committed.

Vector Version VW 4.2.2 (a64.lnx/228) logout
Fri Nov  3 03:24:47 2017
```

Check the log size again and it should be much less in size. There should also be another file next to it named `vectorwise.log-<today's date>`. This can be safely deleted.

Shut Down Actian Vector

1. Connect to Hubble Accelerator server.
2. Switch user to the root user.

```
# su root
```

3. Run the following command.

```
# /etc/init.d/actian-vectorVW stop
```

Appendix - Additional Information relating to Log Propagation

Actian Vector is a columnar relational database that utilizes performance features in modern CPUs to perform extremely fast data analysis and reporting. Although Vector is ACID compliant, transaction secure and supports all DML operations, it is not meant to be used for an OLTP type workload and transaction processing.

Traditionally, incremental updates is a weakness of column based database systems. Vector uses a new innovative technique by keeping track of the tuple position and performing incremental updates in a new, in-memory data structure called Positional Delta Tree (PDT) for maintaining such positional updates.

Updates that reside in memory can potentially consume a significant percentage of system memory. As a result, Vector automatically propagates the changes buffered in memory to disk-resident tables (UpdatePropagation) and provides statements for manually propagating in-memory updates to disk (COMBINE).

Appendix 2: EBS 12.2 Replication Online Patching

Summary

In order to provide a better integrated solution for EBS 12.2 on a replicated environment, the current replication script required extra steps to assist automating the creation of DB objects to support online patching for EBS 12.2.

Enhancements to Replication Script

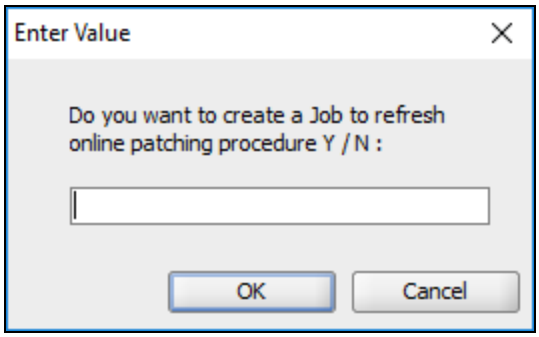
Some extra permissions have been added to the script to support the online patching process. These are as follows:

- CREATE JOB grant – this permission allows the replication schema, created through the script, to create DBMS scheduled jobs. The job that will be created through the script will have the ability to run the online patching procedure at a set time every day.
- CREATE PROCEDURE grant – this permission allows the replication schema, created through the script, to create the procedure which will create a table and populate with a list of EBS 12.2 tables that have been patched/editioned.
- CREATE TABLE grant – this permission allows the replication schema, created through the script, to create a table via the procedure in order to contain the list of EBS 12.2 tables that have been patched/editioned.
- The script will now create, only for EBS 12.2, a procedure named HUB_TABLE_SET_DEF_PK so that a DBA does not need to create it. The procedure will create a table, if one is not present, named HUB_TABLE_SET_DEF and maintain this table with the correct run set data identifier for each table that is editioned. The identifier will either be ORA$BASE, SET1 or SET2. The procedure will be created under the replication schema being created by the replication script and once created, the replication script will run the procedure. The procedure will only be created for 12.2 systems when the replication script is run on it. The DBA can also manually run the procedure once created.

- Supplemental logging for the new table HUB_TABLE_SET_DEF has been added to the script so Qlik can replicate the data with CDC. If the DBA selects auto_supplemental_log = N then the permission is specifically added but if the DBA selects auto_supplemental_log = Y then Qlik can add the permission. This is required so Qlik can add logs to the relevant tables in order to support CDC.
- The last enhancement for the replication script is the ability to add an Oracle job if the DBA requires it, in order to refresh the HUB_TABLE_SET_DEF with the latest run set data identifiers by running the HUB_TABLE_SET_DEF_PK created earlier in the script. The DBA can choose via a prompt whether to create the job or not. The name of the job is a maximum of 30 characters and job name will be made up of the prefix of HUB_TSD_ and then the replication schema name for example HUB_TSD_REP_SCHEMA_1. If the job already exists, it will be dropped and re-created. The job will be scheduled to run at midnight.
- The specific reason for creating the job is to ensure that if the DBA fails to run it as part of the Patching cutover then it will be automatically run.

Setup

After running through the script and entering the initial details like schema name and password for example, the script will run as normal granting the expected permissions for selecting data and then supplemental logging permissions, if required. Once this is complete, if running on 12.2 systems, the script will run through the online patching section for creating the procedure and job if required. The following screenshots will show the output from this process along with the job creation question and resulting output if the job is created:



Screenshot of output from a newly created replication schema, with no procedure or job present:

```
anonymous block completed
anonymous block completed
Creating HUB_TABLE_SET_DEF_PK procedure for HUB_DG_PATCH_TEST2
Procedure created
Creating Table HUB_TABLE_SET_DEF
Complete MAR-07-2018 13:49:26
Procedure executed
GRANT ALTER ON HUB_DG_PATCH_TEST2.HUB_TABLE_SET_DEF TO HUB_DG_PATCH_TEST2
Creating job
Job created
```

Screenshot of output from an existing replication schema, with procedure and job already present:

```
anonymous block completed
anonymous block completed
Creating HUB_TABLE_SET_DEF_PK procedure for HUB_DG_PATCH_TEST
Procedure created
Creating Table HUB_TABLE_SET_DEF
Table HUB_TABLE_SET_DEF already exists, existing rows will be updated.
Complete MAR-07-2018 13:46:58
Procedure executed
GRANT ALTER ON HUB_DG_PATCH_TEST.HUB_TABLE_SET_DEF TO HUB_DG_PATCH_TEST
Dropping existing job
Job dropped
Creating job
Job created
```

Screenshot of output from a newly created replication schema, with no job creation required:

```
anonymous block completed
anonymous block completed
Creating HUB_TABLE_SET_DEF_PK procedure for HUB_DG_PATCH_TEST3
Procedure created
Creating Table HUB_TABLE_SET_DEF
Complete MAR-07-2018 13:54:01
Procedure executed
GRANT ALTER ON HUB_DG_PATCH_TEST3.HUB_TABLE_SET_DEF TO HUB_DG_PATCH_TEST3
```

All of the screenshots above were used with supplemental logging set to **Y** so that Qlik can add the permissions it requires