Requirements for Kafka

Contents

Kafka

Capture	Hub	Integrate
✗	✗	✓

This section describes the requirements, access privileges, and other features of HVR when using Kafka for replication. For information about compatibility and supported versions of Kafka with HVR platforms, Platform Compatibility Matrix.

For the Capabilities supported by HVR on Kafka, see Capabilities for Kafka.

For instructions to quickly setup replication into Kafka, see Quick Start for HVR - Kafka.

Installation Dependencies

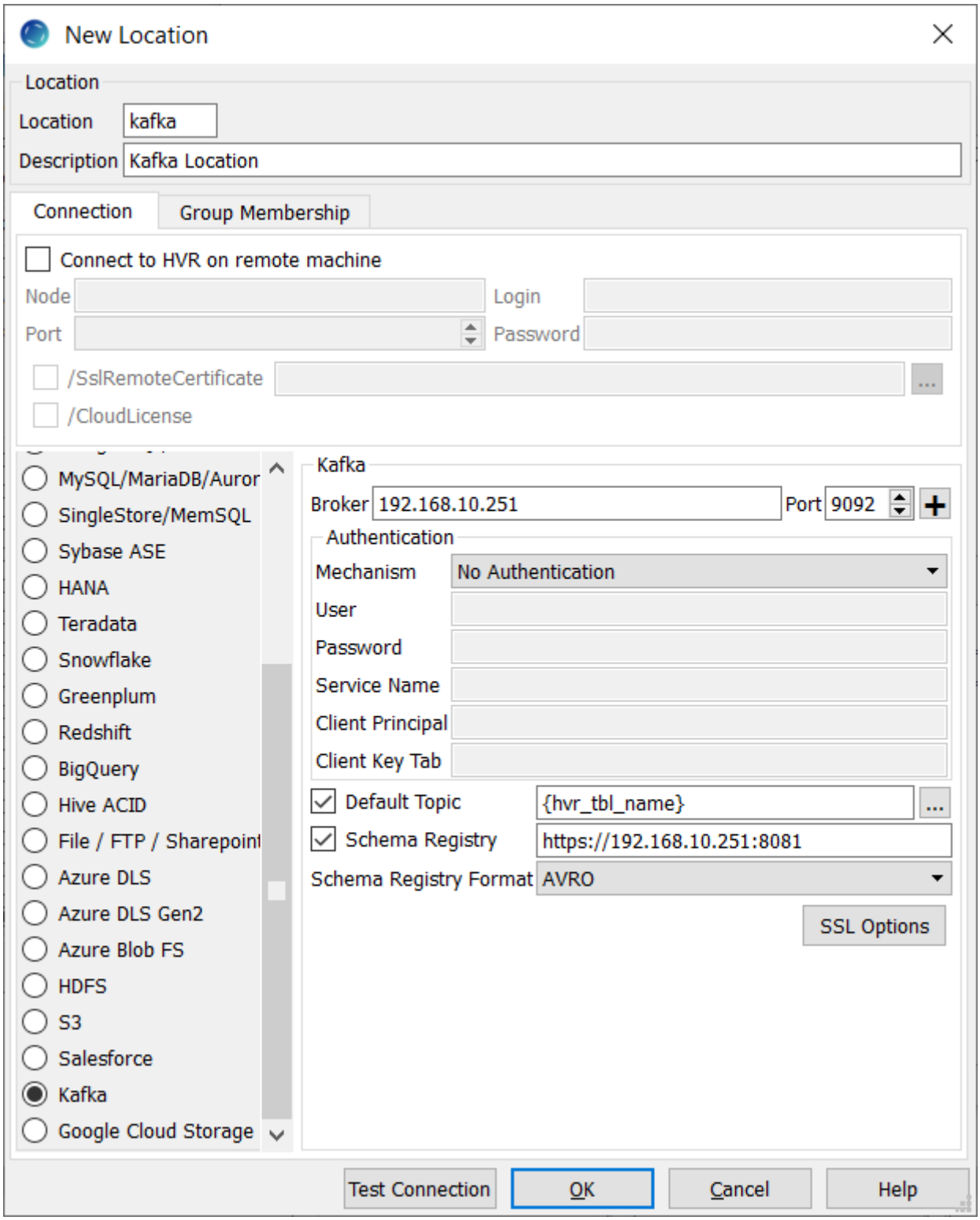
On Linux, to use either of the Kafka authentication **Mechanism** - **User Name and Password** or **Kerberos** (see Location Connection below), HVR requires the library **libsasl2.so.2** to be installed. This library is part of Cyrus SASL and can be installed as follows:

```
$ yum install cyrus-sasl          # On Red Hat Linux, CentOS
$ zypper install cyrus-sasl          # On SUSE Linux
```

There are no special requirements for installing Kafka on Windows.

Location Connection

This section lists and describes the connection details required for creating Kafka location in HVR. HVR uses **librdkafka** (C library which talks Kafka's protocol) to connect to Kafka.

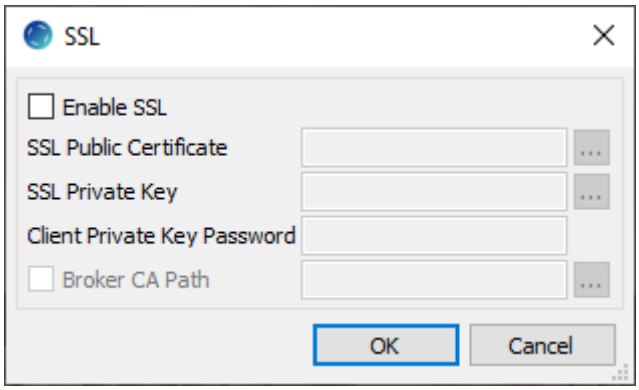


Field	Description
Kafka	
Broker	The hostname or IP address of the Kafka broker server. **Example:** 192.168.10.251 When connecting to Kafka clusters in Confluent Cloud, use the **Bootstrap server** from the **Cluster settings** section of the Confluent Cloud web interface. **Example:** pkc_lgwgm.eastus2.azure.confluent.cloud

Port	The TCP port that the Kafka server uses to listen for client connections. The default port is 9092. **Example:** 9092 HVR supports connecting to more than one Kafka broker servers. Click to add more Kafka brokers.
Authentication	
Mechanism	The authentication mode for connecting HVR to Kafka server (**Broker**). Available options: • **No Authentication** (default) • **User Name and Password** • **Kerberos** On Linux, to use **User Name and Password** or **Kerberos**, HVR requires the **libsasl2.so.2** library to be installed. For more information, see Installation Dependencies. When connecting to Kafka clusters in Confluent Cloud, set the authentication mode to **User Name and Password**.
User	The username to connect HVR to Kafka server. This field is enabled only if **Mechanism** is **User Name and Password**. When connecting to Kafka clusters in Confluent Cloud, use the **API access key** obtained from the Confluent Cloud web interface.
Password	The password of the **User** to connect HVR to Kafka server. This field is enabled only if **Mechanism** is **User Name and Password**. When connecting to Kafka clusters in Confluent Cloud, use the **API access secret** obtained from the Confluent Cloud web interface.
Service Name	The Kerberos Service Principal Name (SPN) of the Kafka server. This field is enabled only if **Mechanism** is **Kerberos**.
Client Principal	The full Kerberos principal of the client connecting to the Kafka server. This is required only on Linux. This field is enabled only if **Mechanism** is **Kerberos**.
Client Key Tab	The directory path where the Kerberos keytab file containing key for **Client Principal** is located. This field is enabled only if **Mechanism** is **Kerberos**.
Default Topic	The Kafka topic to which the messages are written. **Example:** {hvr_tbl_name}_avro
Schema Registry	The URL (**http://** or **https://**) of the schema registry to use Confluent compatible messages. **Example:** http://192.168.10.251:8081 If the basic authentication is configured for the schema registry, then the login credentials (username and password) must be specified in the URL. The format is **http**[**s**]**://***user*:*password*@*schemaregistry_url*:*port* **Example:** https://myuser:mypassword@abcd-efgh.eu-central-1.aws.confluent.cloud

Schema Registry Format	The Kafka message format. Available options: • **Avro** • **JSON** For more information, see Kafka Message Format.
SSL Options	Show **SSL Options**.

SSL Options



Field	Description
Enable SSL	Enable/disable (one way) SSL. If enabled, HVR authenticates the Kafka server by validating the SSL certificate shared by the Kafka server.
SSL Public Certificate	The directory path where the **.pem** file containing the client's SSL public certificate is located.
SSL Private Key	The directory path where the **.pem** file containing the client's SSL private key is located.
Client Private Key Password	The password of the private key file that is specified in **SSL Private Key**.
Broker CA Path	The directory path where the file containing the Kafka broker's self-signed CA certificate is located. When connecting to Kafka clusters in Confluent Cloud, a Broker CA must be specified here. HVR supplies the CA in **$HVR_HOME/lib/cert/ca-bundle.crt**

Integrate and Refresh Target

HVR allows you to **Integrate** or **Refresh** changes into Kafka as a target location. HVR uses **librdkafka** to send data packages into Kafka message bus during **Integrate** (continuous) and **Refresh** (bulk).

This section describes the configuration requirements for integrating changes (using **Integrate** and **HVR Refresh**) into Kafka location. For the list of supported Kafka versions, into which HVR can integrate changes, see Integrate changes into location in Capabilities.

Kafka Message Format

HVR's Kafka location sends messages in the JSON format by default using mode **SCHEMA_PAYLOAD** (see **FileFormat /JsonMode**), unless the location connection option **Schema Registry Format** is used, in which case each message uses the compact AVRO-based format (**Schema Registry Format** - **Avro**) or the JSON file format using mode **ROW_FRAGMETS** (**Schema Registry Format** - **JSON**).

- **Schema Registry Format - Avro**: This is not a true AVRO because each message would not be a valid AVRO file (e.g. no file header). Rather, each message is a 'micro AVRO' containing fragments of data encoded using AVRO data type serialization format.
- **Schema Registry Format - JSON**: Schema is encoded according to the JSON schema specification.
- **All Formats**: the JSON format (using mode **SCHEMA_PAYLOAD**, see **FileFormat /JsonMode**), the 'micro AVRO' format, and the JSON schema encoding (starting from Confluent Platform 5.5.0) conform to the Confluent's 'Kafka Connect' message format standard and can be used with any implementation of Kafka sink connectors. When Kafka location is configured with option **Schema Registry Format** (see section Location Connection above), action **FileFormat** with parameters **/Xml**, **/Csv**, **/AvroCompression** or **/Parquet** cannot be used.

Backward compatibility: The Avro format (default for the **Schema Registry Format** location connection option) is compatible with any Confluent platform below 5.5.0.

If you want to use the Cloudera **Schema Registry Format**, you must use it in the Confluent compatible mode. This can be achieved by indicating the URL in the following format: **http://***FQDN***:***PORT***/api/v1/confluent**, where *FQDN:PORT* is the address of the Cloudera Schema Registry specified in the **Schema Registry Format** field when configuring the location (see section Location Connection above).

Action **FileFormat** with parameters **/Xml**, **/Csv**, **/Avro** or **/Parquet** can be used to send messages in other formats. If parameter **/Avro** is chosen without enabling location connection option **Schema Registry Format**, then each message would be a valid AVRO file (including a header with the schema and column information), rather than the Kafka Connect's compact AVRO-based format.

The Kafka messages should also contain special columns containing the operation type (delete, insert and update) and the sequence number. For achieving this, define action **ColumnProperties** for the Kafka location as mentioned below:

Group	Table	Action
KAFKA	*	**ColumnProperties /Name=op_val /Extra /Datatype=integer /IntegrateExpression="{hvr_op}"**
KAFKA	*	**ColumnProperties /Name=integ_seq /Extra /Datatype=varchar /Length=36 /IntegrateExpression="{hvr_integ_seq}" /TimeKey**

Metadata for Messages

To process HVR's messages, a Kafka consumer will often need metadata (table and column names, data types etc) about the replicated table. If location connection option **Schema Registry Format** is set, then it can read this from that registry. For JSON format with the default mode (**SCHEMA_PAYLOAD**), each message contains this information. Another way to include metadata to each message is to add actions **ColumnProperties /Extra /IntegrateExpression** to add values like **{hvr_tbl_name}** and **{hvr_op}**.

Kafka Message Bundling and Size

By default, each Kafka message contains just one row, regardless of the format chosen. Multiple rows can be bundled into a single message using **Integrate /MessageBundling** with either of the following bundling modes:

- **CHANGE:** update message contains both 'before' and 'after' rows, inserts and deletes just contain one row
- **TRANSACTION:** message contains all rows associated with a captured transaction
- **THRESHOLD:** message is filled with rows until it reaches limit. Bundled messages simply consist of the contents of several single-row messages concatenated together.

For more information on bundling modes, see parameter **/MessageBundling** of action **Integrate**.

Although bundling of multiple rows can be combined with the Kafka Connect compatible formats (JSON with default mode **SCHEMA_PAYLOAD**), the resulting (longer) messages no longer conform to Confluent's 'Kafka Connect' standard.

For bundling modes **TRANSACTION** and **THRESHOLD**, the number of rows in each message is affected by action **Integrate** **/MessageBundlingThreshold** (default is **800,000**). For those bundling modes, rows continue to be bundled into the same message until after this threshold is exceeded. After that happens, the message is sent and new rows are bundled into the next message.

Parameter **/MessageBundlingThreshold** has no effect on the bundling modes **ROW** or **CHANGE**.

By default, the minimum size of a Kafka message sent by HVR is 4096 bytes; the maximum size of a Kafka message is 1,000,000 bytes; HVR will not send a message exceeding this size and will instead give a fatal error; if **Integrate** **/MessageCompress** parameter is used, this error will be raised by a Kafka broker. You can change the maximum Kafka message size that HVR will send by defining **$HVR_KAFKA_MSG_MAX_BYTES**, but ensure not to exceed the maximum message size configured in Kafka broker (settings **message.max.bytes**). If the message size exceeds this limit then the message will be lost.

HVR_KAFKA_MSG_MAX_BYTES works in two ways:

- checks the size of a particular message and raises an HVR error if the size is exceeded even before transmitting it to a Kafka broker.
- checks the maximum size of compressed messages inside the Kafka transport protocol.

If the message is too big to be sent because it contains multiple rows, then less bundling (e.g. **/MessageBundling=ROW**) or using a lower **MessageBundlingThreshold** can help reducing the number of rows in each message. Otherwise, the number of bytes used for each row must be lowered; either with a more compact message format or even by actually truncating a column value (by adding action **ColumnProperties** **/TrimDatatype** to the capture location).

HVR does not recommend to use parameters **/Compress** or **/AvroCompression** of action **FileFormat** with Kafka location (even if the location connection option **Schema Registry Format** is not set). Instead, use parameter **/MessageCompress** of action **Integrate**.

Syncing Kafka, Interruption of Message Sending, and Consuming Messages with Idempotence

An HVR integrate job performs a sync of messages sent into Kafka at the end of each integrate cycle, instead of after each individual message. This means if the job is interrupted while it is sending messages, and when it is restarted, the sending of multiple rows from the interrupted cycle may be repeated. Programs consuming Kafka messages must be able to cope with this repetition; this is called being 'idempotent'. One technique to be idempotent is to track an increasing sequence in each message and use detect which messages have already been processed. A column with such an increasing sequence can be defined using action **ColumnProperties** **/Name=integ_key /Extra /Datatype=varchar /Length=32 /IntegrateExpression="{hvr_integ_seq}"**. If HVR resends a message, its contents will be identical each time, including this sequence number.

Kafka Message Keys and Partitioning

Kafka messages can contain a 'key' which Kafka uses to put messages into partitions, so consuming can be parallelized. HVR typically puts a key into each message which contains a hash computed from values in the 'distribution key' column of each row. This key is present only if the messages are in JSON or AVRO format. It is not present when the message bundling (**/MessageBundling**) mode is **TRANSACTION** or **THRESHOLD**.

Known Issue

When using Kafka broker version 0.9.0.0 or 0.9.0.1, an existing bug (KAFKA-3547) in Kafka causes timeout error in HVR.

The workaround to resolve this issue is to define action **Environment** for the Kafka location as mentioned below:

Group	Table	Action
KAFKA	*	**Environment /Name=HVR_KAFKA_BROKER_VERSION /Value=0.9.0.1**

If the Kafka broker version used is 0.9.0.0 then **/Value=0.9.0.0**