HCLSoftware

Unica Link V12.1.8 JDBC Connector User Guide

Contents

Chapter 1. Overview..... 1
Chapter 2. Prerequisites..... 2
Chapter 3. Configuration.....3
Chapter 4. Testing the Connection.....4
Chapter 5. Process Box or Touchpoint Configuration.....5
Chapter 6. Event Tracking.....6

Chapter 1. Overview

You can use the JDBC connector to perform database operations like `insert`, `delete`, `update` and `upsert`. You can also execute SQL queries.

Run the JDBC connector from Unica Journey, which maintains a list of uploaded data. A response is sent back to the Kafka location containing fields like `status`, `timestamp`, `errormessage`, and `IDENITITY`, which comes with the input.

Chapter 2. Prerequisites

To use the JDBC connector, your system must meet the following prerequisites:

- Database version and drivers that supports JDBC connector.
- Database drivers that comply with JDBC 4.2 or later specifications.
- Based on the installation type, place the drivers in the appropriate location. Depending on your database, place the appropriate JAR files within the `com.hcl.hip.adapters.m4jdbc` folder in the `DTXHOME/jars` location.
    - **Native installation**

      `DTXHOME` is `<Link_Installation_Folder>/tomcat-context/install`
    - **Docker installation**

      `DTXHOME` for Server: `/opt/hcl/hip`

      `DTXHOME` for Rest/Runtime: `/opt/runtime`
    - **Windows Installation**

      `DTXHOME` is `<Link_Installation_Folder>`

Note: Unica Link installation does not package drivers with the installer.

Chapter 3. Configuration

To use JDBC connector, contact your Unica administrator. The administrator will configure a connection for you.

To configure a connection, the Unica administrator must configure a few connection properties. The connection properties are listed in the following table:

Table 1. Connection properties to configure a JDBC connector

Property	Description
`URL`	The URL used to connect to the database. it should contain the connection string. For example: **MYSQL**: `jdbc:mysql://Server Name/Server IP>:<Port>/<DB name>` **Oracle**: `jdbc:oracle:thin:@<Server Name/ Server IP>:<PORT>:<Oracle service name>` where *thin* is the driver, *<Server Name/ Server IP>* on which oracle is running, *<PORT>* is the port number and *<Oracle service name>* is the Oracle service name.
`User`	Username of the database on which you want to perform the operation.
`Password`	Password of the database on which you want to perform the operation.
`Driver`	Specify the driver class name. **Driver name for MySq**: `com.mysql.cj.jdbc.Driver` **Driver name for Oracle**: `oracle.jdbc.driver.OracleDriver`

Note:

- To perform update operation on a record in the database table, you need a primary key (the primary key is a column, or set of columns, whose values uniquely identify each row in the table) in the table, which helps in updating the existing record in the table.
- JDBC connector has the batch size of 10 with each batch containing `10,000` records.
- JDBC connector supports Unica Journey and not Unica Campaign.
- Logs are generated and stored in the assigned location.

Chapter 4. Testing the Connection

To test the connection, complete the following steps:

1. Click the **Test** button.

   The system invokes a query operation on JDBC connector to confirm if the following conditions are met:

   - The server is reachable.
   - The connection parameters are valid.
2. To create a connection, select JDBC connection for the connection type.
3. Select the details and provide the credentials to test the connection.

Chapter 5. Process Box or Touchpoint Configuration

Property	Description
`Operation`	To select the operation type table or custom SQL query (Table / Custom SQL).
`Write Mode (Table)`	What kind of mode we want to perform over database. For example, `insert`, `update`, `delete`, or `upsert`. Similar query is performed over the database.
`Catalog (Table)`	Helps in selecting the catalog present in the database. Depends on the database as to how the catalog is created and used (depending on the JDBC driver, the Schema, the Catalog, or both Schema and Catalog are required).
`Schema (Table)`	Depends on the database and the selected schema present in the database (depending on the JDBC driver, the Schema, the Catalog, or both Schema and Catalog are required).
`Table (Table)`	Select the table on which operation need to performed.
`SQL Query (Custom SQL)`	Helps in writing complex queries. You can pass the variables enclosed in “`{}`” which will be visible on the mapping screen.

In case of Table operation, the field mapping screen is generated based on the selected table type, and in case of Custom SQL type operation, the field mapping screen is generated based on the variable passed inside the {} in Custom SQL textbox.

Field name	Description
`Table` - dynamic field	This depends on the table selected in the action screen and accordingly fields will visible as per the columns existing in the database table. There are other details visible with column names like description, data type, and length of the field in the database.
`Custom SQL` - dynamic field	Fields which are passed as variable enclosed in "{}" are visible in the mapping screen and input will passed accordingly. Similar description will be provided and values are parsed from the braces will be visible on the mapping screen.

Chapter 6. Event Tracking

The JDBC connector, as an input, provides fields that exist in the database and performs operations over the database using the input file.

Responses will be generated for each record, with their description, based on the final response.

The response contains the following details:

- `Status`
- `TimeStamp`
- `ErrorMessage`
- `Identity field`