Kafka Connect DBMaker Guide

Version: 5.4

Document No: 54/DBM54-T02032023-01-DBKA

Author: DBMaster Support Team, SYSCOM Computer Engineering CO.

Print Date: February 03, 2023

凌羣電腦
THE SYSCOM GROUP

Table of Content

1. Introductions

1.1 Purpose

In this Kafka Connect DBMaker tutorial, we'll cover reading from DBMaker to Kafka and reading from Kafka to DBMaker. The focus will be keeping it simple and get it working.

Let's run this on your environment.

2. Prerequisites

In this Kafka Connect with DBMaker tutorial, you'll need

- DBMaker 5.4.x
- DBMaker JDBC driver
- DBMaker dialect
- Kafka 3.3.1
- Confluentinc-kafka-connect-jdbc
- Java 8+

3. Installing DBMaker

3.1 Step 1: Get DBMaker

Windows platform

```
ftp://dev.dbmaker.com.tw/pub/DBMaker/5.4.5/testing/dbmaker-5.4.5-win64.exe
```

Linux platform

```
ftp://dev.dbmaker.com.tw/pub/DBMaker/5.4.5/testing/dbmaker-5.4.5-Linux2.x86_64.tar.gz
```

3.2 Step 2: Installing DBMaker

Windows platform

```
Click Install dbmaker-5.4.5-win64.exe and follow the setup instructions on the screen
```

Linux platform

```
# Create a dbmaker account and log in to this account

$ tar -xvzf dbmaker-5.4.5-Linux2.x86_64.tar.gz
$ cd 5.4
```

3.3 Step 3: startup database and create tables

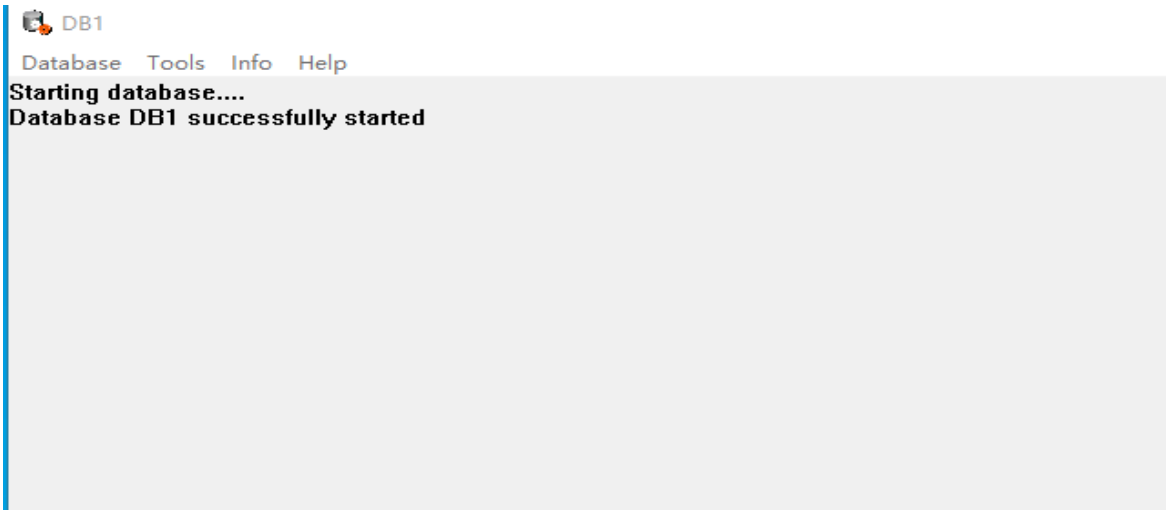
Create the two databases: The source DB1 and the target DB2.

The file C:\DBMaker\5.4\dmconfig.ini as follows:

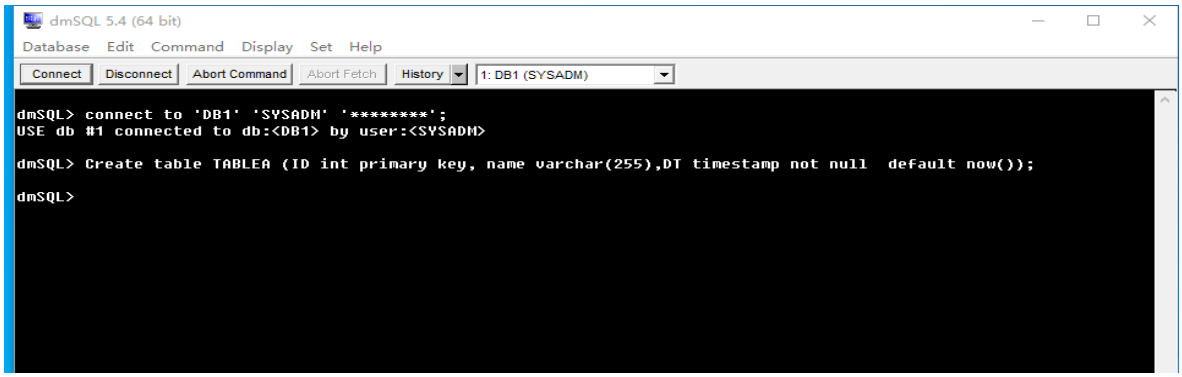
```
[DB1]
DB_DBDIR = C:\DBMaker\5.4\bin
DB_FODIR = C:\DBMaker\5.4\bin\fo
DB_PTNUM = 3333
DB_SvAdr=127.0.0.1
DB_UsrID=SYSADM
DB_TMOFM = hh:mm:ss.fff

[DB2]
DB_DBDIR = C:\DBMaker\5.4\bin
DB_FODIR = C:\DBMaker\5.4\bin\fo
DB_PTNUM = 4444
DB_SvAdr=127.0.0.1
DB_UsrID=SYSADM
DB_TMOFM = hh:mm:ss.fff
```

- Open DBMaker Server Application, Startup the source database DB1

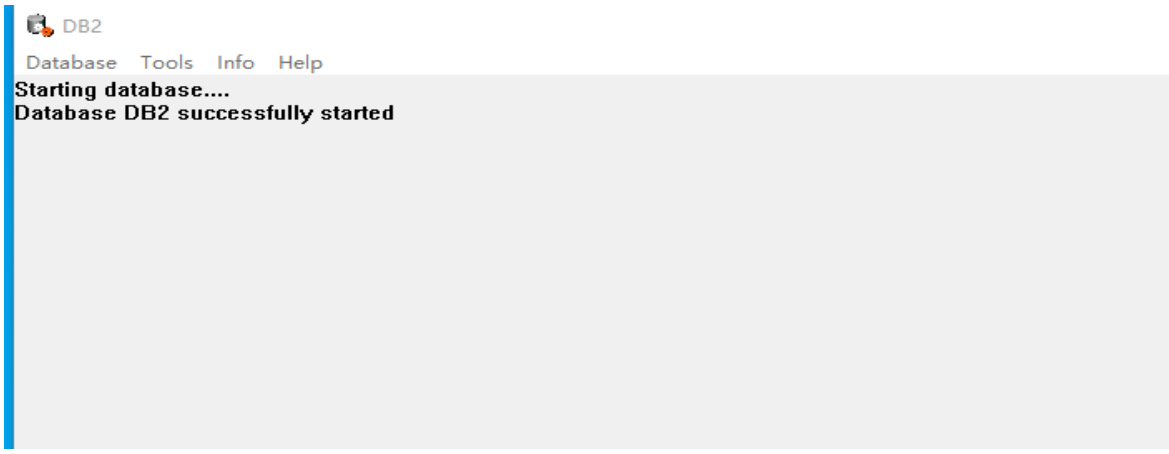


- Open dmSQL Tool Application, create the source table TABLEA



dmSQL> Create table TABLEA (ID int primary key, name varchar(255),DT timestamp not null default now());

- Open DBMaker Server Application, Startup the target database DB2



Note: We don't need to create the target table, Kafka creates it automatically.

4. Installing Kafka

4.1 Step 1: Get Kafka

Download the latest Kafka release and extract it:

https://www.apache.org/dyn/closer.cgi?path=/kafka/3.3.1/kafka_2.13-3.3.1.tgz

Windows platform

```
Extract kafka_2.13-3.3.1.tgz to C:\kafka
```

Linux platform

```
$ tar -xzf kafka_2.13-3.3.1.tgz
$ cd kafka_2.13-3.3.1
```

4.2 Step 2: Installing Kafka connect plugins

The JDBC source connector and sink connector allow you to import/export data from any relational database with a JDBC driver into Kafka topics.

- Download installation

https://docs.confluent.io/kafka-connectors/jdbc/current/index.html

Download File Name: confluentinc-kafka-connect-jdbc-10.6.0.zip

Extract it into one of the directories that is listed on the Connect worker's plugin.path configuration properties.

- Defining plugins

Edit the config/connect-standalone.properties file (C:\kafka), add or change the plugin.path configuration property match the following, and save the file:

```
# Examples:

# plugin.path= /usr/local/share/java,/usr/local/share/kafka/plugins,/opt/connectors,



plugin.path=C:/kafka/plugins
```

Create a new folder C:\kafka\plugins\kafka-connect-jdbc path on the C:\kafka and copy confluentinc-kafka-connect-jdbc-10.6.0* (extracted in step 2) to the folder.

```
C:\kafka\plugins\kafka-connect-jdbc
 ├──assets
 ├──doc
 │    └──licenses
 ├──etc
 ├──lib
 └──manifest.json
```

4.3 Step 3: Copy DBMaker Connector Jar files

It contains two files:

- dmjdbc30.jar (DBMaker jdbc driver, C:\DBMaker\5.4\bin)
- dbmaker-kafka-10.2.0-2.0.jar (DBMaker dialect file)

Copy the dmjdbc30.jar and dbmaker-kafka-10.2.0-2.0.jar to kafka connect plugin lib/.

```
C:\kafka\plugins\kafka-connect-jdbc\lib\dmjdbc30.jar

C:\kafka\plugins\kafka-connect-jdbc\lib\dbmaker-kafka-10.2.0-2.0.jar
```

Then you need to configure an instance of your connector.

4.4 Step 4: Configure kafka connectors instance

You can create a connector configuration file with the connector's settings, and deploy that to a Connect worker. Simply, copy C:\kafka\plugins\kafka-connect-jdbc\etc\ sample files to get the new properties files to config\.

Properties file: config\source-dbmaker.properties

```
# Configuration specific to the JDBC source connector.
name=test-connector-dbmaker
connector.class=io.confluent.connect.jdbc.JdbcSourceConnector
tasks.max=1

connection.url=jdbc:dbmaker:db1
connection.user=SYSADM
connection.password=
dialect.name=DBMakerDatabaseDialect
table.whitelist=TABLEA

mode=timestamp
timestamp.column.name=DT
quote.sql.identifiers=always
```

Properties file: config\sink-dbmaker.properties

```
# Configuration specific to the JDBC sink connector.
name=test-connector-dbmaker
connector.class=io.confluent.connect.jdbc.JdbcSinkConnector
tasks.max=1

# The topics to consume from - required for sink connectors like this one
```

```
topics=TABLEA


connection.url=jdbc:dbmaker:db2
connection.user=SYSADM
connection.password=
dialect.name=DBMakerDatabaseDialect


auto.create=true
quote.sql.identifiers=always
```

5. Starting kafka

5.1 Step 1: Start Zookeeper and Kafka services

Run the following commands in order to start all services in the correct order:

Open a terminal session and start zookeeper service:

Windows platform

```
# Start the ZooKeeper service
bin\windows\zookeeper-server-start.bat config\zookeeper.properties
```

Linux platform

```
# Start the ZooKeeper service
$ bin/zookeeper-server-start.sh config/zookeeper.properties
```

Open another terminal session and start kafka:

Windows platform

```
bin\windows\kafka-server-start.bat config\server.properties
```

Linux platform

```
# Start the Kafka broker service
$ bin/kafka-server-start.sh config/server.properties
```

Once all services have successfully launched, you will have a basic Kafka environment running and ready to use.

5.2 Step 2: Configure Kafka Connect in standalone mode

We'll see how to run Kafka Connect with jdbc connectors that import data from DBMaker to a Kafka topic and export data from a Kafka topic to DBMaker.

Next, we'll start source connector running in *standalone* mode, which means they run in a single, local, dedicated process.

Windows platform

```
bin\windows\connect-standalone.bat config\connect-standalone.properties config\source-dbmaker.properties
```

Linux platform

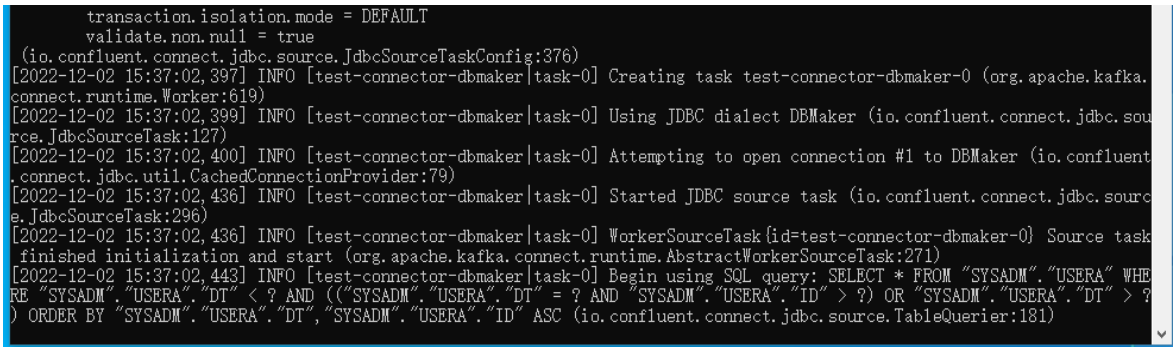
```
$bin/connect-standalone.sh config/connect-standalone.properties config/source-dbmaker.properties
```

It provides two configuration files as parameters:

- connect-standalone.properties (Kafka Connect process)
- source-dbmaker.properties (connector configuration file)

Once the Kafka Connect process has started, the source connector should start reading data from DBMaker and producing them to the topic TABLEA.

connect-standalone.bat screen as follows:



```
        transaction.isolation.mode = DEFAULT
        validate.non.null = true
 (io.confluent.connect.jdbc.source.JdbcSourceTaskConfig:376)
[2022-12-02 15:37:02,397] INFO [test-connector-dbmaker|task-0] Creating task test-connector-dbmaker-0 (org.apache.kafka.
connect.runtime.Worker:619)
[2022-12-02 15:37:02,399] INFO [test-connector-dbmaker|task-0] Using JDBC dialect DBMaker (io.confluent.connect.jdbc.sou
rce.JdbcSourceTask:127)
[2022-12-02 15:37:02,400] INFO [test-connector-dbmaker|task-0] Attempting to open connection #1 to DBMaker (io.confluent
.connect.jdbc.util.CachedConnectionProvider:79)
[2022-12-02 15:37:02,436] INFO [test-connector-dbmaker|task-0] Started JDBC source task (io.confluent.connect.jdbc.sourc
e.JdbcSourceTask:296)
[2022-12-02 15:37:02,436] INFO [test-connector-dbmaker|task-0] WorkerSourceTask{id=test-connector-dbmaker-0} Source task
 finished initialization and start (org.apache.kafka.connect.runtime.AbstractWorkerSourceTask:271)
[2022-12-02 15:37:02,443] INFO [test-connector-dbmaker|task-0] Begin using SQL query: SELECT * FROM "SYSADM"."USERA" WHE
RE "SYSADM"."USERA"."DT" < ? AND (("SYSADM"."USERA"."DT" = ? AND "SYSADM"."USERA"."ID" > ?) OR "SYSADM"."USERA"."DT" > ?
) ORDER BY "SYSADM"."USERA"."DT","SYSADM"."USERA"."ID" ASC (io.confluent.connect.jdbc.source.TableQuerier:181)
```

5.3 Step 3: Processing source data

Add two rows to DBMaker source Table and check if the Console Consumer receives this message.

```
dmSQL> connect to 'DB1' 'SYSADM' '********';
USE db #1 connected to db:<DB1> by user:<SYSADM>


dmSQL> Insert into TABLEA values(1,'kafka-1');
dmSQL> Insert into TABLEA values(2,'kafka-2');
```

Running a console consumer to see the data in the topic.

Windows platform

```
bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic TABLEA
--isolation-level "read_committed" --from-beginning
```

Linux platform

```
$bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic TABLEA --
isolation-level "read_committed" --from-beginning
```

Important: the topic name is the DBMaker table name <TABLEA>.

If the consumer receives the message. You will get the following output:


```
{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"ID"},{"type":"string","opti
onal":true,"field":"NAME"},{"type":"int64","optional":false,"name":"org.apache.kafka.connect.data.
Timestamp","version":1,"field":"DT"}],"optional":false,"name":"TABLEA"},"payload":{"ID":1,"NAME"
:"kafka-1","DT":1670250079929}}

{"schema":{"type":"struct","fields":[{"type":"int32","optional":false,"field":"ID"},{"type":"string","opti
onal":true,"field":"NAME"},{"type":"int64","optional":false,"name":"org.apache.kafka.connect.data.
Timestamp","version":1,"field":"DT"}],"optional":false,"name":"TABLEA"},"payload":{"ID":2,"NAME"
:"kafka-2","DT":1670250452896}}
```

5.4 Step 4: Processing sink data

Next, close the front connect-standalone.bat session and start the sink connector with configuration file <config\sink-dbmaker.properties>.

Windows platform

```
bin\windows\connect-standalone.bat config\connect-standalone.properties config\sink-dbmaker.properties
```

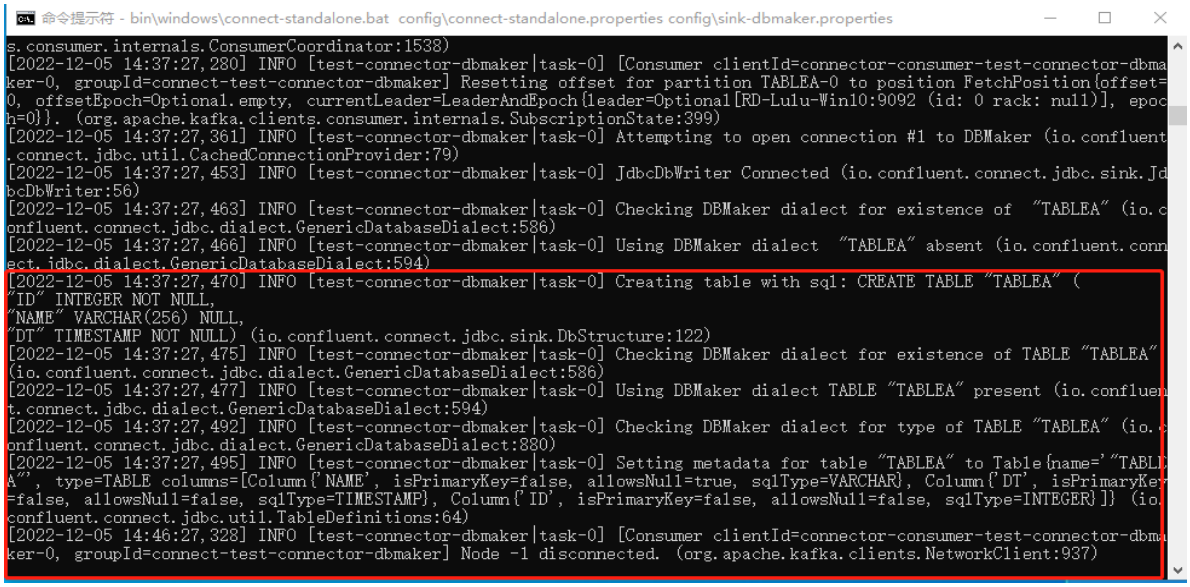
Linux platform

```
$bin/connect-standalone.sh config/connect-standalone.properties config/sink-dbmaker.properties
```

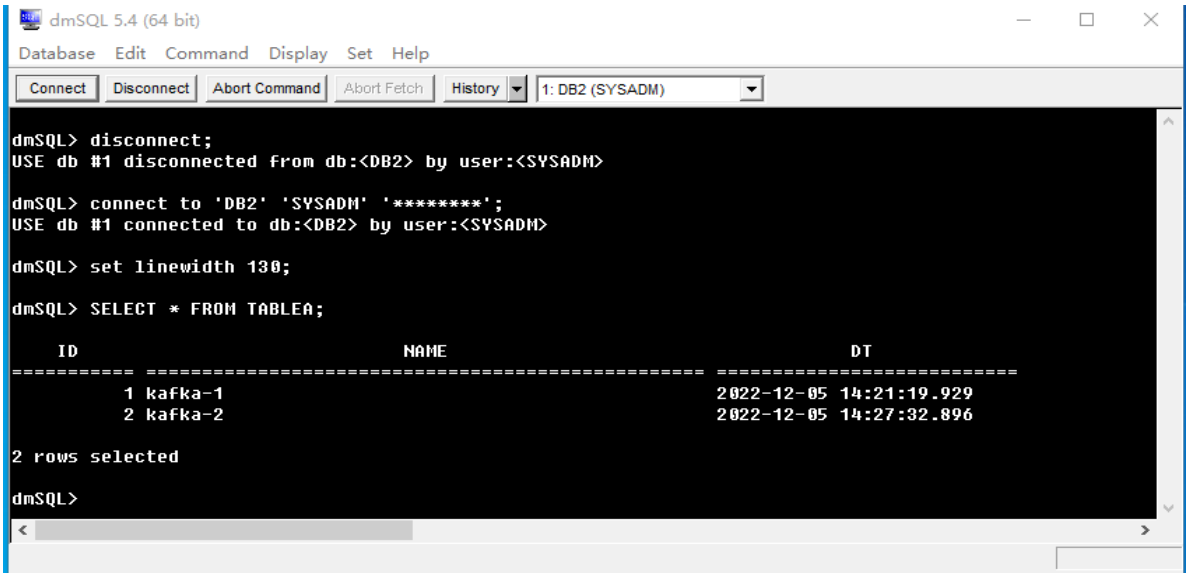
It provides two configuration files as parameters:

- connect-standalone.properties (Kafka Connect process)
- sink-dbmaker.properties (connector configuration file)

Once the Kafka Connect process has started, and the sink connector should start reading messages from the topic TABLEA and write them to the DBMaker DB2 database.



- Open dmSQL Tool Application, connect to the target database and check TABLEA data.



5.5 Step 5: Checking the Kafka status

You can now check and verify the connectors using the following line of code:

List the Kafka topics:

Windows platform

```
bin\windows\kafka-topics.bat --bootstrap-server localhost:9092 -list
```

Linux platform

```
$ bin/kafka-topics.sh --bootstrap-server localhost:9092 --list
```

List active connectors on a worker:

```
Curl localhost:8083/connectors
```

Checking the connector Status:

```
curl -s -XGET "http://localhost:8083/connectors/test-connector-dbmaker/status"
```

Checking the connector config:

```
curl -s -XGET "http://localhost:8083/connectors/test-connector-dbmaker/config"
```

Checking the kafka logs:

```
C:\kafka\logs
```

6. Terminate Kafka

Tear down the Kafka environment.

- Stop the producer and consumer clients with **Ctrl-C**, if you haven't done so yet.
- Stop the Kafka broker with **Ctrl-C**.
- Lastly, if the Kafka with ZooKeeper section was followed, stop the ZooKeeper server with **Ctrl-C**.

7. CONCLUSION

In this article, you have learned how to perform DBMaker to Kafka ETL using the Kafka connector.

You can still test a **distributed mode** locally by setting a different rest port.

let me know if you have any questions or suggestions for improvement. Feedback always welcomed.

8. Additional Documents

FileName: source-dbmaker.properties

```
# Copyright 2018 Confluent Inc.
name=test-connector-dbmaker
connector.class=io.confluent.connect.jdbc.JdbcSourceConnector
tasks.max=1

connection.url=jdbc:dbmaker:db1
connection.user=SYSADM
connection.password=
dialect.name=DBMakerDatabaseDialect
table.whitelist=TABLEA

mode=timestamp
timestamp.column.name=DT
quote.sql.identifiers=always
```

FileName: sink-dbmaker.properties

```
# Copyright 2018 Confluent Inc.
name=test-connector-dbmaker
connector.class=io.confluent.connect.jdbc.JdbcSinkConnector
tasks.max=1

# The topics to consume from - required for sink connectors like this one
topics=TABLEA

connection.url=jdbc:dbmaker:db2
connection.user=SYSADM
connection.password=
dialect.name=DBMakerDatabaseDialect
auto.create=true
quote.sql.identifiers=always
```