

# **Data Transfer Service**

## **Data Subscription (OldDTS)**

### **Product Documentation**



## Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

## Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

## Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

# Contents

## Data Subscription (OldDTS)

Database/Table Data Subscription to Local File System

Database/Table Data Subscription to Redis

Database/Table Data Subscription to Kafka

# Data Subscription (OldDTS) Database/Table Data Subscription to Local File System

Last updated : 2021-01-06 10:04:08

This document provides a simple example that walks you through how to pull a table from data subscription to a local file system as well as a simple [Local Demo](#). The following operations are performed on CentOS.

## Configuring Environment

- Java environment configuration

```
yum install java-1.8.0-openjdk-devel
```

- [Download the data subscription SDK](#)

## Getting Key

Log in to the [CAM console](#) to get a key.

## Selecting Data Subscription

1. Log in to the [DTS console](#), and select **Data Subscription** on the left to go to the data subscription page.
2. Select the name of the TencentDB instance to be synced, click **Start**, return to the data subscription page, and click the data subscription you created. For detailed directions, please see [How to Get a Data Subscription](#).
3. Check the corresponding DTS channel, IP, and port and enter them together with the obtained key into the corresponding `LocalDemo.java` file.

```
// Enter the key obtained from TencentCloud API here  
context.setSecretId("AKIDfdfsdfsdt1331431sdfs"); Enter the `secretID` obtained  
d from TencentCloud API  
context.setSecretKey("test111usdfsdfsddsfrkeT"); Enter  
the `secretKey` obtained from TencentCloud API
```

```
// Enter the IP and port obtained through data subscription in the DTS service here
context.setServiceIp("10.66.112.181"); Enter the IP obtained from the data subscription configuration
context.setServicePort(7507);
Enter the port obtained from the data subscription configuration
final DefaultSubscribeClient client = new DefaultSubscribeClient(context);
// Enter the names of both the database and table to sync and modify the name of the file where they will be stored
final String targetDatabase = "test"; Enter the name of the database to subscribe to
final String targetTable = "alantest"; Enter the name of the table to subscribe to

final String fileName = "test.alan.txt"; Enter the name of the local file for s torage

client.addClusterListener(listener);
// Enter the `dts-channel` configuration information obtained from the data subscription configuration here
client.askForGUID("dts-channel-e4FQxtYV3It4test"); Enter the DTS channel name o btained from the data subscription
client.start();
```

## Compiling and Testing

1. `javac -classpath binlogsdk-2.6.0-release.jar -encoding UTF-8 LocalDemo.java`

2. Launch the program. If no errors are reported, the program works properly. Check the previously configured local file.

```
java -XX:-UseGCOverheadLimit -Xms2g -Xmx2g -classpath .:binlogsdk-2.6.0-release.jar LocalDemo
```

3. Check the previously configured `test.alan.txt` file and you can see that data has been pulled into it.

```
[root@VM_71_10_centos ~]# cat test.alan.txt
checkpoint:144251@3@357589@317713
record_id:000001000000000004D9110000000000000001
record_encoding:utf8
fields_enc:latin1,utf8
gtid:4f21864b-3bed-11e8-a44c-5cb901896188:1562
source_category:full_recorded
source_type:mysql
```

```
table_name:alantest
record_type:INSERT
db:test
timestamp:1523356039
primary:
Field name: id
Field type: 3
Field length: 2
Field value: 26
Field name: name
Field type: 253
Field length: 4
Field value: alan
```

# Database/Table Data Subscription to Redis

Last updated : 2020-04-27 17:04:09

This document provides a simple example that walks you through how to pull a table from data subscription to Redis as well as a simple [Redis Demo](#). The following operations are performed on CentOS.

## Configuring Environment

- Java environment configuration

```
yum install java-1.8.0-openjdk-devel
```

- [Download the data subscription SDK](#)
- [Download jedis-2.9.0.jar](#)

## Getting Key

Log in to the [CAM Console](#) to get a key.

## Selecting Data Subscription

1. Log in to the [DTS Console](#) and select **Data Subscription** on the left sidebar to enter the data subscription page.
2. Select the name of the TencentDB instance to be synced, click "Start", return to the data subscription page, and click the data subscription you created. For detailed directions, please see [How to Get a Data Subscription](#).
3. Check the corresponding DTS channel, IP, and port and enter them together with the obtained key into the corresponding `RedisDemo.java`.

```
context.setSecretId("AKIDfsdfsdfsdt1331431sdfs"); Enter the `secretID` obtained
from TencentCloud API.
context.setSecretKey("test111usdfsdfsddsfrkeT"); Enter the `secretKey` obtained f
rom TencentCloud API.
// Enter the IP and port obtained through data subscription in the DTS service he
re
context.setServiceIp("10.66.112.181"); Enter the IP obtained from the data subscr
iption configuration
context.setServicePort(7507); Enter the port obtained from the data subscription
configuration
```

```
// Create a consumer
//SubscribeClient client=new DefaultSubscribeClient(context,true);
final DefaultSubscribeClient client = new DefaultSubscribeClient(context);

final Jedis jedis = new Jedis("127.0.0.1", 6379); Enter your corresponding Redis
host and port

final String targetDatabase = "test"; Enter the name of the database to subscribe
to
final String targetTable = "alantest"; Enter the name of the table to subscribe t
o. The table has two fields, namely, `id` and `name`, of which `id` is the primar
y key

// Create a subscription listener
ClusterListener listener = new ClusterListener() {
@Override
public void notify(List<ClusterMessage> messages) throws Exception {
// System.out.println("-----:" + messages.size());
for(ClusterMessage m:messages){
DataMessage.Record record = m.getRecord();
// Filter out uninteresting subscription messages
if(!record.getDbName().equalsIgnoreCase(targetDatabase) || !record.getTablename
().equalsIgnoreCase(targetTable)){
// Note: Ack must be performed for uninteresting messages too
m.ackAsConsumed();
continue;
}

if(record.getOpt() != DataMessage.Record.Type.BEGIN && record.getOpt() != DataMes
sage.Record.Type.COMMIT){
List<DataMessage.Record.Field> fields = record.getFieldList();

//INSERT RECORD
//String pk = record.getPrimaryKeys();

if(record.getOpt() == DataMessage.Record.Type.INSERT){

String keyid="";
String value="";
for (DataMessage.Record.Field field : fields) {

// Get the `id` value as the primary key first, find the `name` column, and inser
t the corresponding values of `key` and `name` in Redis
if(field.getFieldname().equalsIgnoreCase("id")){
keyid=field.getValue();
continue;
```



```
}  
if(field.getFieldname().equalsIgnoreCase("name")){  
value=field.getValue();  
  
}  
jedis.set(keyid, value);  
}  
  
}
```

## Compiling and Testing

1. 

```
[root@VM_71_10_centos ~]# javac -classpath binlogsdk-2.6.0-release.jar:jedis-2.9.0.jar -encoding UTF-8 RedisDemo.java
```

2. Launch the program. If no errors are reported, the program works properly. Check the previously configured local file.

```
java -XX:-UseGCOverheadLimit -Xms2g -Xmx2g -classpath .:binlogsdk-2.6.0-release.jar:jedis-2.9.0.jar RedisDemo
```

3. Perform `INSERT` and `UPDATE` operations. If you find that the data has been indeed inserted and modified successfully in Redis, you can perform `DELETE` operations to delete the corresponding data from Redis.

```
MySQL [test]> insert into alantest values(1001,'alan1');  
Query OK, 1 row affected (0.00 sec)
```

```
MySQL [test]> update alantest set name='alan2' where id=1001;  
Query OK, 1 row affected (0.00 sec)  
Rows matched: 1 Changed: 1 Warnings: 0
```

```
-----  
127.0.0.1:6379> get 1001  
"alan2"
```

```
MySQL [test]> update alantest set name='alan3' where id=1001;  
Query OK, 1 row affected (0.00 sec)  
Rows matched: 1 Changed: 1 Warnings: 0
```

```
-----  
127.0.0.1:6379> get 1001
```

```
"alan3"
```

```
MySQL [test]> delete from alantest where id=1001;  
Query OK, 1 row affected (0.00 sec)
```

```
-----
```

```
127.0.0.1:6379> get 1001  
(nil)
```

# Database/Table Data Subscription to Kafka

Last updated : 2021-01-06 10:04:08

This document provides a simple example that walks you through how to pull a table from data subscription to Kafka as well as a simple [Kafka Demo](#).

## Configuring Environment

- OS: CentOS
- Download relevant resources
  - [Data subscription SDK](#)
  - [SLF4J components](#)
  - [Kafka-clients](#)
- Java environment configuration

```
yum install java-1.8.0-openjdk-devel
```

## Installing Kafka

1. Please install Kafka as instructed in [Kafka Quick Start](#).
2. After Kafka is launched, create a "testtop" topic.

```
bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1  
--partitions 1 --topic testtop  
Created topic "testtop".
```

## Getting Key

Log in to the [CAM console](#) to get a key.

## Selecting Data Subscription

1. Log in to the [DTS console](#), and select **Data Subscription** on the left to go to the data subscription page.

- In the subscription list, click a subscription name to enter the subscription details page and view the corresponding channel ID, service IP, and service port.
- Enter them together with the obtained key into the corresponding `KafkaDemo.java`.

```
import com.qcloud.dts.context.SubscribeContext;
import com.qcloud.dts.message.ClusterMessage;
import com.qcloud.dts.message.DataMessage;
import com.qcloud.dts.subscribe.ClusterListener;
import com.qcloud.dts.subscribe.DefaultSubscribeClient;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.clients.producer.ProducerRecord;
import org.apache.kafka.common.serialization.StringSerializer;
import org.apache.log4j.Logger;

import java.util.List;
import java.util.Properties;

public class KafkaDemo {
    public static void main(String[] args) throws Exception {
        //Initialize a kafka producer
        final String TOPIC = "testtop";
        Properties props = new Properties();
        props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, "10.168.1.6:9092");
        props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
        props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);

        final Producer<String, String> producer = new KafkaProducer<String, String>(prop
s);

        //Create a context
        SubscribeContext context = new SubscribeContext();
        context.setSecretId("AKIDPko5fVtvTDE0WffffkCwd4NzKcdePt79uauy");
        context.setSecretKey("ECtY8F5e2QqtdXAe18yX0EBqK");
        // Subscription channel region
        context.setRegion("ap-beijing");
        final DefaultSubscribeClient client = new DefaultSubscribeClient(context);

        // Create a subscription listener
        ClusterListener listener = new ClusterListener() {
            @Override
            public void notify(List<ClusterMessage> messages) throws Exception {
                System.out.println("-----:" + messages.size());
                for (ClusterMessage m:messages){
```

```
DataMessage.Record record = m.getRecord();

if(record.getOpt() != DataMessage.Record.Type.BEGIN && record.getOpt() != DataMessage.Record.Type.COMMIT) {
List<DataMessage.Record.Field> fields = record.getFieldList();

//Print the information of each column
for (int i = 0; i < fields.size(); i++) {
DataMessage.Record.Field field = fields.get(i);
System.out.println("Database Name:" + record.getDbName());
System.out.println("Table Name:" + record.getTablename());
System.out.println("Field Value:" + field.getValue());
System.out.println("Field Value:" + field.getValue().length());
System.out.println("Field Encoding:" + field.getFieldEnc());
}

//Send the entire record to the specified Kafka topic
System.out.println("Record+++++++");
producer.send(new ProducerRecord<String, String>(TOPIC, record.toString()));
}

m.ackAsConsumed();
}
}
@Override
public void onException(Exception e){
System.out.println("listen exception" + e);
}
};
// Add a listener
client.addClusterListener(listener);
client.askForGUID("dts-channel-p15e9eW9rn8hA68K");
client.start();
}
}
```

## Compiling and Testing

1. Compile the client program `KafkaDemo.java` .

```
javac -classpath binlogsdk-2.9.1-jar-with-dependencies.jar:slf4j-api-1.7.25.jar:slf4j-log4j12-1.7.2.jar:kafka-clients-1.1.0.jar -encoding UTF-8 KafkaDemo.java
```

2. Launch the program. If no errors are reported, the program works properly.

```
java -XX:-UseGCOverheadLimit -Xms2g -Xmx2g -classpath .:binlogsdk-2.9.1-jar-with-dependencies.jar:kafka-clients-1.1.0.jar:slf4j-api-1.7.25.jar:slf4j-log4j12-1.7.2.jar KafkaDemo
```

3. Insert a data entry into the `alantest` table, and you will find that the data has been stored in the `testtop` subscribed to by Kafka.

```
MySQL [test]> insert into alantest values(123456, 'alan');
Query OK, 1 row affected (0.02 sec)
[root@VM_71_10_centos kafka_2.11-1.1.0]# bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic testtop --from-beginning
checkpoint:144251@3@1275254@1153089
record_id:0000010000000000011984100000000000000001
record_encoding:utf8
fields_enc:latin1,utf8
gtid:4f21864b-3bed-11e8-a44c-5cb901896188:5552
source_category:full_recorded
source_type:mysql
table_name:alantest
record_type:INSERT
db:test
timestamp:1524649133
primary:id
Field name: id
Field type: 3
Field length: 6
Field value: 123456
Field name: name
Field type: 253
Field length: 4
Field value: alan
```