Data Ingestion Service

Best Practices

Issue	01
Date	2025-02-12

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Huawei Cloud Computing Technologies Co., Ltd.

Address:	Huawei Cloud Data Center Jiaoxinggong Road Qianzhong Avenue Gui'an New District Gui Zhou 550029 People's Republic of China
Website:	https://www.huaweicloud.com/intl/en-us/

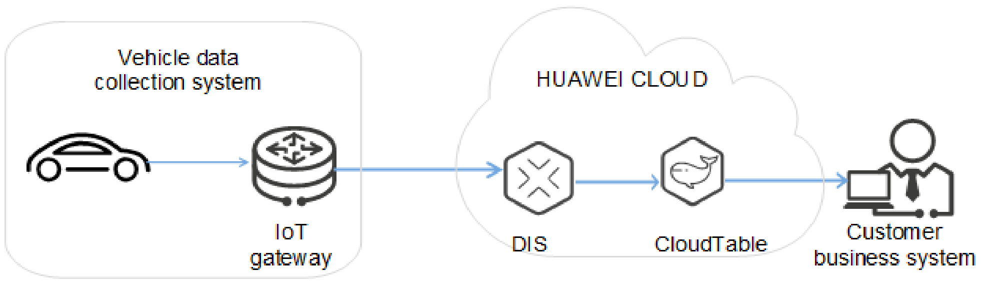
Contents

1 Using DIS to Analyze Vehicle Locations in Real Time

Scenario Overview

Data Ingestion Service (DIS) collects vehicle location data in real time and uploads the data to CloudTable Service (CloudTable). You can use CloudTable to query locations of a vehicle in a specified period.

Figure 1-1 Service process diagram



The procedure is as follows:

1. **Creating a CloudTable Cluster**
2. **Creating a Data Table on CloudTable**
3. **Creating a DIS Stream**
4. **Creating a Dump Task**
5. **Obtaining Authentication Information**
6. **Preparing a DIS Application Development Environment**
7. **Developing an Application for Sending Data to DIS**
8. **Starting the Data Upload Application**
9. **Viewing the Uploaded Data on CloudTable**
10. **Querying Locations of a Specified Vehicle on CloudTable**

Creating a CloudTable Cluster

Create a CloudTable cluster for storing the data dumped from DIS.

Creating a Data Table on CloudTable

Create a CloudTable data table after creating a CloudTable cluster.

The collected data is in JSON format. An example is as follows:

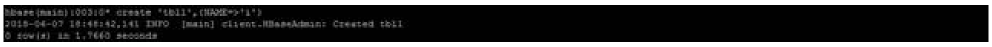
```
{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55378500","DataTime":"2017-10-23
12:19:35.000","Latitude":"34.585639","IsACCOpen":"true","Longitude":"119.193524","Velocity":0,"Direction":"
null","Carnum":"WL66666","BaiDuLatitude":"34.607069","BaiDuLongitude":"119.190093","BaiDuAddress":"N
o. 78 Tongguan North Road, Xinpu District, Lianyungang City, Jiangsu Province of
China","ReceiveTime":"2017-10-23 12:19:34.846","Altitude":"null"}
```

In this practice, you can create data tables using the HBase shell client.

Step 1 Prepare a Linux Elastic Cloud Server (ECS).

Step 2 Install and start HBase shell to access the CloudTable cluster.

Step 3 On the HBase shell client, run the **create 'tbl1',{NAME => 'i'}** command to create a data table. After the data table is successfully created, the information similar to the following is displayed:



```
hbase(main):003:0* create 'tbl1',{NAME=>'i'}
2018-06-07 18:48:42,141 INFO  [main] client.HBaseAdmin: Created tbl1
0 row(s) in 1.7660 seconds
```

----End

Creating a DIS Stream

Create a stream. For details, see **Creating a DIS Stream**.

Creating a Dump Task

Step 1 Log in to the DIS console.

Step 2 In the navigation tree, choose **Stream Management**.

Step 3 Click the stream created in **Creating a DIS Stream**. On the displayed page, click the **Dump Management** tab.

Step 4 Click **Create Dump Task**. On the **Create Dump Task** page, configure dump parameters.

NOTE

- A maximum of five dump tasks can be created for each stream.
- A dump task cannot be added to a stream whose **Source Data Type** is **FILE**.

Step 5 Click **Create Now**.

Table 1-1 Dump task parameters

Parameter	Description	Value
Dump Destination	**CloudTable**: The streaming data is stored to DIS while being imported to the HBase table and OpenTSDB of the CloudTable cluster in real time.	CloudTable

Parameter	Description	Value
Task Name	Name of the dump task. The task name must be unique in a stream. A task name is 1 to 64 characters long. Only letters, digits, hyphens (-), and underscores (_) are allowed.	-
Offset	● **Latest**: Maximum offset, indicating that the latest data will be extracted. ● **Earliest**: Minimum offset, indicating that the earliest data will be extracted.	Latest
CloudTable Cluster	Click **Select**. In the **Select CloudTable Cluster** dialog box, select a CloudTable cluster. This parameter cannot be left blank. You can only select but not enter a value in this field.	cloudtable-demo
Table Type	Two CloudTable types are available: **HBase** and **OpenTSDB**.	HBase
Table	Click **Select**. In the **Select CloudTable Table** dialog box, select a CloudTable table. You can only select but not enter a value in this field. **NOTE** This parameter is available only after you select a CloudTable cluster and create an HBase table.	tbl1
Backup	Specifies whether to back up the data that fails to be dumped to CloudTable to OBS. ● Enable: The data will be backed up to OBS. ● Disable: The data will not be backed up to OBS. Backup is disabled by default. **NOTE** If this function is disabled, data that fails the dump will be stored in DIS and be cleared when **Data Retention** is expired.	Disabled

Parameter	Description	Value
Row Key	● JSON attribute name. The value contains a maximum of 32 characters and consists only of letters, digits, underscores (_), and periods (.). The value cannot start or end with a period or contain consecutive periods and cannot be left blank. A maximum of 64 attributes can be added. ● Data type. Possible values: – Bigint – Double – Boolean – Timestamp – String – Decimal	-
Row Key Delimiter	Delimiter used to separate row keys. The value can only be a period (**.**), comma (**,**), vertical bar (**|**), semicolon (**;**), hyphen (**-**), underscore (**_**), or tilde (**~**). It can also be set to **NULL**. The maximum length is one character.	-

Parameter	Description	Value
Schema Column	• Column name. The value contains a maximum of 32 characters and consists only of letters, digits, and underscores (_). The value cannot be left blank. A maximum of 4,096 columns can be added. • Data type. Possible values: – Bigint – Double – Boolean – Timestamp – String – Decimal • JSON attribute name. The value contains a maximum of 32 characters and consists only of letters, digits, underscores (_), and periods (.). The value cannot start or end with a period or contain consecutive periods and cannot be left blank. • Column family. The value can only be selected from the drop-down list box and cannot be left blank. This parameter is available only after you select a CloudTable cluster and a CloudTable table, and set **Type** of a CloudTable table to **HBase**.	See **Table 1-2**.

Table 1-2 Schema column parameters

Column Name	Data Type	JSON Attribute Name	Column Family
DeviceID	String	DeviceID	i
Mileage	Bigint	Mileage	i
Latitude	Decimal	Latitude	i
IsACCOpen	Boolean	IsACCOpen	i

Column Name	Data Type	JSON Attribute Name	Column Family
Longitude	Decimal	Longitude	i
Velocity	Bigint	Velocity	i
Direction	String	Direction	i
BaiDuLatitude	Decimal	BaiDuLatitude	i
BaiDuLongitude	Decimal	BaiDuLongitude	i
BaiDuAdress	String	BaiDuAdress	i
ReceiveTime	Timestamp	ReceiveTime	i
Altitude	String	Altitude	i

----End

Obtaining Authentication Information

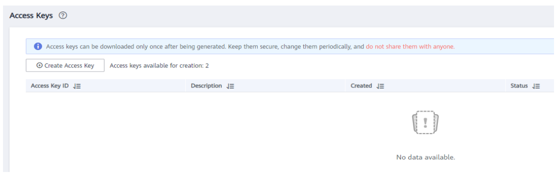
- Obtaining an Access Key ID/Secret Access Key (AK/SK)

  To obtain an access key, perform the following steps:

  a. Log in to the management console, move the cursor to the username in the upper right corner, and select **My Credentials** from the drop-down list.

  b. On the **My Credentials** page, choose **Access Keys**, and click **Create Access Key**. See **Figure 1-2**.

  Figure 1-2 Clicking Create Access Key

  Access keys can be downloaded only once after being generated. Keep them secure, change them periodically, and do not share them with anyone.

  Access keys available for creation: 2

  No data available.

  c. Click **OK** and save the access key file as prompted. The access key file will be saved to your browser's configured download location. Open the **credentials.csv** file to view **Access Key Id** and **Secret Access Key**.

  NOTE

  - Only two access keys can be added for each user.
  - To ensure access key security, the access key is automatically downloaded only when it is generated for the first time and cannot be obtained from the management console later. Keep them properly.

- Obtaining a project ID and account ID

  A project is a group of tenant resources, and an account ID corresponds to the current account. The IAM ID corresponds to the current user. You can view the project IDs, account IDs, and user IDs in different regions on the corresponding pages.

a. Register with and log in to the management console.
b. Hover the cursor on the username in the upper right corner and select **My Credentials** from the drop-down list.
c. On the **API Credentials** page, obtain the account name, account ID, IAM username, and IAM user ID, and obtain the project and its ID from the project list.

- Obtaining the endpoint

  An endpoint is the **request address** for calling an API. Endpoints vary depending on services and regions. You can obtain the endpoints of the service from **Regions and Endpoints**.

Preparing a DIS Application Development Environment

For details, see **Preparing a DIS Application Development Environment**.

Developing an Application for Sending Data to DIS

Step 1 Prepare a data sample.

Step 2 Modify the sample code.

The sample project is the **ProducerDemo.java** file in the **\dis-sdk-demo\src\main\java\com\bigdata\dis\sdk\demo** directory of the **huaweicloud-sdk-dis-java-.zip** package downloaded in the **Preparing a DIS Application Development Environment**.

Change the values of **AK**, **SK**, and **ProjectId** based on the site requirements.

```
 private static void runProduceDemo()
   {
//Create a DIS client instance.
       DIS dic = DISClientBuilder.standard()
               .withEndpoint("https://dis.cn-north-1.myhuaweicloud.com:20004")
               .withAk("${your_AK}")
               .withSk("${your_SK}")
               .withProjectId("${your_projectId}")
               .withRegion("cn-north-1")
               .build();

//Configure the stream name.
       String streamName = "dis-demo";

//Configure the data to be uploaded.
       PutRecordsRequest putRecordsRequest = new PutRecordsRequest();
       putRecordsRequest.setStreamName(streamName);

       List<PutRecordsRequestEntry> putRecordsRequestEntryList = new
ArrayList<PutRecordsRequestEntry>();
String[] messages = {     prepared data sample    };

       for (int i = 0; i < messages.length; i++)
       {
          ByteBuffer buffer = ByteBuffer.wrap(messages[i].getBytes());
           PutRecordsRequestEntry putRecordsRequestEntry = new PutRecordsRequestEntry();
           putRecordsRequestEntry.setData(buffer);

putRecordsRequestEntry.setPartitionKey(String.valueOf(ThreadLocalRandom.current().nextInt(1000000)));
           putRecordsRequestEntryList.add(putRecordsRequestEntry);
       }
       putRecordsRequest.setRecords(putRecordsRequestEntryList);
```

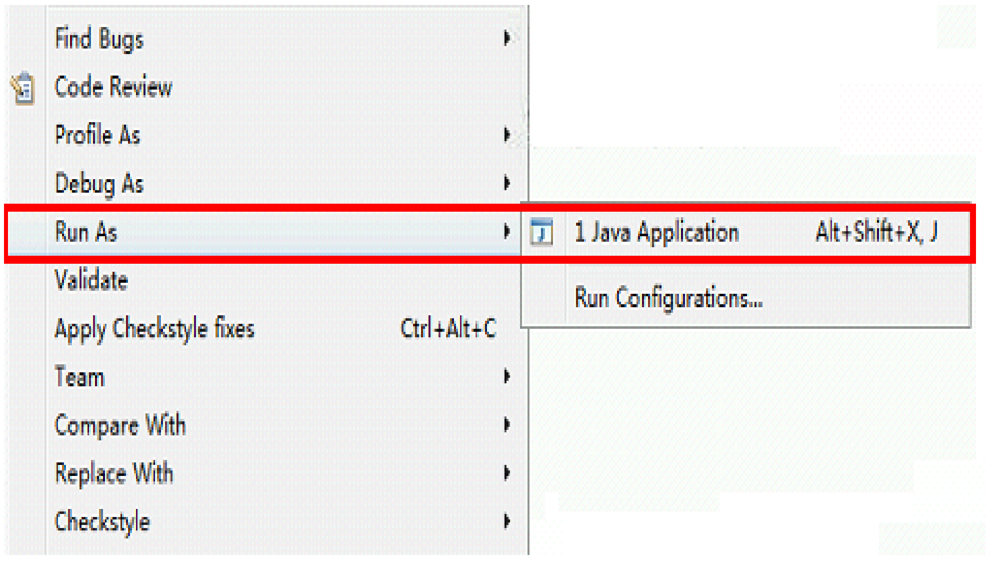
```
        log.info("========== BEGIN PUT ============");

        PutRecordsResult putRecordsResult = null;
        try
        {
            putRecordsResult = dic.putRecords(putRecordsRequest);
        }
        catch (DISClientException e)
        {
            log.error("Failed to get a normal response, please check params and retry. Error message [{}]",
                e.getMessage(),
                e);
        }
        catch (ResourceAccessException e)
        {
            log.error("Failed to access endpoint. Error message [{}]", e.getMessage(), e);
        }
        catch (Exception e)
        {
            log.error(e.getMessage(), e);
        }

        if (putRecordsResult != null)
        {
            log.info("Put {} records[{} successful / {} failed].",
                putRecordsResult.getRecords().size(),
                putRecordsResult.getRecords().size() - putRecordsResult.getFailedRecordCount().get(),
                putRecordsResult.getFailedRecordCount());

            for (int j = 0; j < putRecordsResult.getRecords().size(); j++)
            {
                PutRecordsResultEntry putRecordsRequestEntry = putRecordsResult.getRecords().get(j);
                if (putRecordsRequestEntry.getErrorCode() != null)
                {
                 //Failed to upload the data.
                    log.error("[{}] put failed, errorCode [{}], errorMessage [{}]",
                        new String(putRecordsRequestEntryList.get(j).getData().array()),
                        putRecordsRequestEntry.getErrorCode(),
                        putRecordsRequestEntry.getErrorMessage());
                }
                else
                {
                    // Data is uploaded successfully.
                    log.info("[{}] put success, partitionId [{}], partitionKey [{}], sequenceNumber [{}]",
                        new String(putRecordsRequestEntryList.get(j).getData().array()),
                        putRecordsRequestEntry.getPartitionId(),
                        putRecordsRequestEntryList.get(j).getPartitionKey(),
                        putRecordsRequestEntry.getSequenceNumber());
                }
            }
        }
        log.info("========== END PUT ============");
    }
```

----End

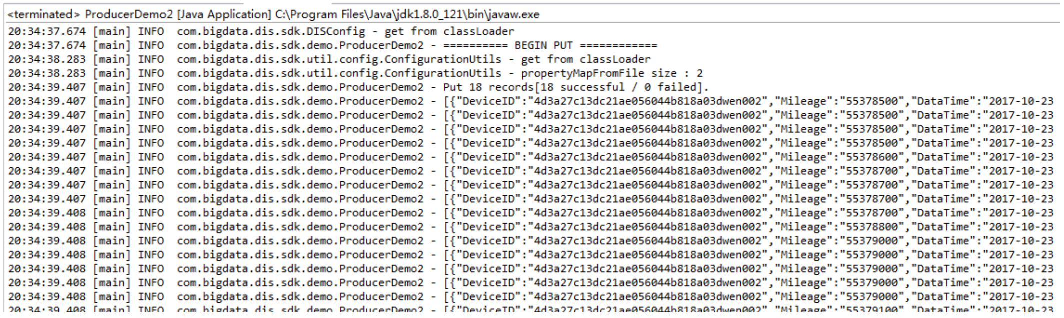
Starting the Data Upload Application

Right-click the producer application and choose **Run As** > **1 Java Application** from the shortcut menu.

Figure 1-3 Running the application



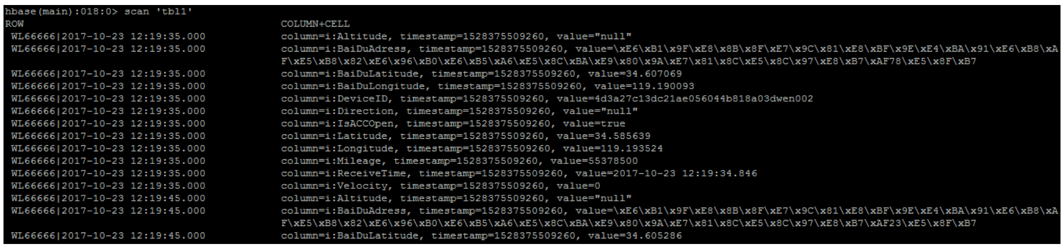
While data is being sent to DIS, the DIS console displays DIS stream information. If information similar to the following is displayed, the data has been successfully sent to DIS:



```
<terminated> ProducerDemo2 [Java Application] C:\Program Files\Java\jdk1.8.0_121\bin\javaw.exe
20:34:37.674 [main] INFO  com.bigdata.dis.sdk.DISConfig - get from classLoader
20:34:37.674 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - ========== BEGIN PUT ============
20:34:38.283 [main] INFO  com.bigdata.dis.sdk.util.config.ConfigurationUtils - get from classLoader
20:34:38.283 [main] INFO  com.bigdata.dis.sdk.util.config.ConfigurationUtils - propertyMapFromFile size : 2
20:34:39.407 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - Put 18 records[18 successful / 0 failed].
20:34:39.407 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - [{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55378500","DataTime":"2017-10-23
20:34:39.407 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - [{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55378500","DataTime":"2017-10-23
20:34:39.407 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - [{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55378500","DataTime":"2017-10-23
20:34:39.407 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - [{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55378500","DataTime":"2017-10-23
20:34:39.407 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - [{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55378600","DataTime":"2017-10-23
20:34:39.407 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - [{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55378700","DataTime":"2017-10-23
20:34:39.407 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - [{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55378700","DataTime":"2017-10-23
20:34:39.407 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - [{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55378700","DataTime":"2017-10-23
20:34:39.408 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - [{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55378700","DataTime":"2017-10-23
20:34:39.408 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - [{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55378800","DataTime":"2017-10-23
20:34:39.408 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - [{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55379000","DataTime":"2017-10-23
20:34:39.408 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - [{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55379000","DataTime":"2017-10-23
20:34:39.408 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - [{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55379000","DataTime":"2017-10-23
20:34:39.408 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - [{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55379000","DataTime":"2017-10-23
20:34:39.408 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - [{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55379000","DataTime":"2017-10-23
20:34:39.408 [main] INFO  com.bigdata.dis.sdk.demo.ProducerDemo2 - [{"DeviceID":"4d3a27c13dc21ae056044b818a03dwen002","Mileage":"55379100","DataTime":"2017-10-23
```

Viewing the Uploaded Data on CloudTable

Run the **scan 'tbl1'** command on the HBase shell client. If information similar to the following is displayed, the data has been successfully uploaded to CloudTable:



```
hbase(main):018:0> scan 'tbl1'
ROW                                        COLUMN+CELL
 WL66666|2017-10-23 12:19:35.000           column=i:Altitude, timestamp=1528375509260, value="null"
 WL66666|2017-10-23 12:19:35.000           column=i:BaiDuAdress, timestamp=1528375509260, value=\xE6\xB1\x9F\xE8\x8B\x8F\xE7\x9C\x81\xE8\xBF\x9E\xE4\xBA\x91\xE6\xB8\xA
                                           F\xE5\xB8\x82\xE6\x96\xB0\xE6\xB5\xA6\xE5\x8C\xBA\xE9\x80\x9A\xE7\x81\x8C\xE5\x8C\x97\xE8\xB7\xAF78\xE5\x8F\xB7
 WL66666|2017-10-23 12:19:35.000           column=i:BaiDuLatitude, timestamp=1528375509260, value=34.607069
 WL66666|2017-10-23 12:19:35.000           column=i:BaiDuLongitude, timestamp=1528375509260, value=119.190093
 WL66666|2017-10-23 12:19:35.000           column=i:DeviceID, timestamp=1528375509260, value=4d3a27c13dc21ae056044b818a03dwen002
 WL66666|2017-10-23 12:19:35.000           column=i:Direction, timestamp=1528375509260, value="null"
 WL66666|2017-10-23 12:19:35.000           column=i:IsACCOpen, timestamp=1528375509260, value=true
 WL66666|2017-10-23 12:19:35.000           column=i:Latitude, timestamp=1528375509260, value=34.585639
 WL66666|2017-10-23 12:19:35.000           column=i:Longitude, timestamp=1528375509260, value=119.193524
 WL66666|2017-10-23 12:19:35.000           column=i:Mileage, timestamp=1528375509260, value=55378500
 WL66666|2017-10-23 12:19:35.000           column=i:ReceiveTime, timestamp=1528375509260, value=2017-10-23 12:19:34.846
 WL66666|2017-10-23 12:19:35.000           column=i:Velocity, timestamp=1528375509260, value=0
 WL66666|2017-10-23 12:19:45.000           column=i:Altitude, timestamp=1528375509260, value="null"
 WL66666|2017-10-23 12:19:45.000           column=i:BaiDuAdress, timestamp=1528375509260, value=\xE6\xB1\x9F\xE8\x8B\x8F\xE7\x9C\x81\xE8\xBF\x9E\xE4\xBA\x91\xE6\xB8\xA
                                           F\xE5\xB8\x82\xE6\x96\xB0\xE6\xB5\xA6\xE5\x8C\xBA\xE9\x80\x9A\xE7\x81\x8C\xE5\x8C\x97\xE8\xB7\xAF23\xE5\x8F\xB7
 WL66666|2017-10-23 12:19:45.000           column=i:BaiDuLatitude, timestamp=1528375509260, value=34.605286
```

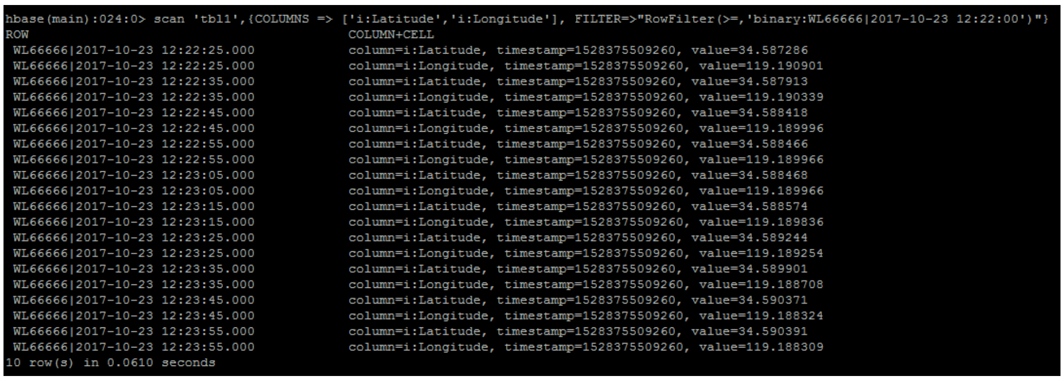
Querying Locations of a Specified Vehicle on CloudTable

The following gives an example about how to query the locations of vehicle **WL66666** after **2017-10-23 12:22:00**.

Step 1 Log in to the HBase shell client.

Step 2 Run the **scan 'tbl1',{COLUMNS => ['i:Latitude','i:Longitude'], FILTER=>"RowFilter(>=,'binary:WL66666|2017-10-23 12:22:00')"}** command to query vehicle locations.

The following query result is displayed on the HBase client:



```
hbase(main):024:0> scan 'tbl1',{COLUMNS => ['i:Latitude','i:Longitude'], FILTER=>"RowFilter(>=,'binary:WL66666|2017-10-23 12:22:00')"}
ROW                                         COLUMN+CELL
 WL66666|2017-10-23 12:22:25.000            column=i:Latitude, timestamp=1528375509260, value=34.587286
 WL66666|2017-10-23 12:22:25.000            column=i:Longitude, timestamp=1528375509260, value=119.190901
 WL66666|2017-10-23 12:22:35.000            column=i:Latitude, timestamp=1528375509260, value=34.587913
 WL66666|2017-10-23 12:22:35.000            column=i:Longitude, timestamp=1528375509260, value=119.190339
 WL66666|2017-10-23 12:22:45.000            column=i:Latitude, timestamp=1528375509260, value=34.588418
 WL66666|2017-10-23 12:22:45.000            column=i:Longitude, timestamp=1528375509260, value=119.189996
 WL66666|2017-10-23 12:22:55.000            column=i:Latitude, timestamp=1528375509260, value=34.588466
 WL66666|2017-10-23 12:22:55.000            column=i:Longitude, timestamp=1528375509260, value=119.189966
 WL66666|2017-10-23 12:23:05.000            column=i:Latitude, timestamp=1528375509260, value=34.588468
 WL66666|2017-10-23 12:23:05.000            column=i:Longitude, timestamp=1528375509260, value=119.189966
 WL66666|2017-10-23 12:23:15.000            column=i:Latitude, timestamp=1528375509260, value=34.588574
 WL66666|2017-10-23 12:23:15.000            column=i:Longitude, timestamp=1528375509260, value=119.189836
 WL66666|2017-10-23 12:23:25.000            column=i:Latitude, timestamp=1528375509260, value=34.589244
 WL66666|2017-10-23 12:23:25.000            column=i:Longitude, timestamp=1528375509260, value=119.189254
 WL66666|2017-10-23 12:23:35.000            column=i:Latitude, timestamp=1528375509260, value=34.589901
 WL66666|2017-10-23 12:23:35.000            column=i:Longitude, timestamp=1528375509260, value=119.188708
 WL66666|2017-10-23 12:23:45.000            column=i:Latitude, timestamp=1528375509260, value=34.590371
 WL66666|2017-10-23 12:23:45.000            column=i:Longitude, timestamp=1528375509260, value=119.188324
 WL66666|2017-10-23 12:23:55.000            column=i:Latitude, timestamp=1528375509260, value=34.590391
 WL66666|2017-10-23 12:23:55.000            column=i:Longitude, timestamp=1528375509260, value=119.188309
10 row(s) in 0.0610 seconds
```

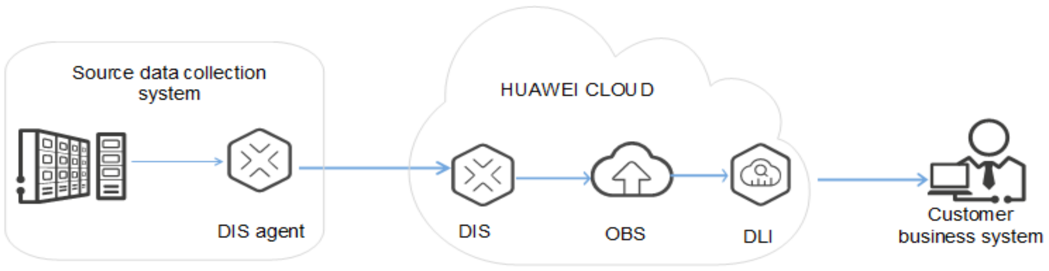
----End

2 Collecting Incremental Log Data of Driving Behavior

Scenario Overview

Data Ingestion Service (DIS) collects incremental log data of driving behavior, uploads it to Object Storage Service (OBS), and analyzes the uploaded log data through Data Lake Insight (DLI). The analysis results reflect driving behaviors and can be used by vehicle enterprises to provide value-added services such as driving habit optimization.

Figure 2-1 Service process diagram



The procedure is as follows:

1. **Creating an OBS Bucket**
2. **Creating a DIS Stream**
3. **Creating a Dump Task**
4. **Obtaining Authentication Information**
5. **Installing an Agent**
6. **Preparing a Data Sample**
7. **Configuring a DIS Agent**
8. **Starting the DIS Agent**
9. **Viewing Uploaded Files on OBS**
10. **Creating a Database**
11. **Creating an OBS Table**

12. **Querying a Data Sample**
13. **Querying Results**

Creating an OBS Bucket

Create an OBS bucket for storing the data dumped by DIS. For details, see **Creating a Bucket**.

Creating a DIS Stream

Create a stream. For details, see **Creating a DIS Stream**.

Creating a Dump Task

Step 1 Log in to the DIS console.

Step 2 In the navigation tree, choose **Stream Management**.

Step 3 Click the stream created in **Creating a DIS Stream**. On the displayed page, click the **Dump Management** tab.

Step 4 Click **Create Dump Task**. On the **Create Dump Task** page, configure dump parameters.

NOTE

- A maximum of five dump tasks can be created for each stream.
- A dump task cannot be added to a stream whose **Source Data Type** is **FILE**.

Step 5 Click **Create Now**.

Table 2-1 Dump task parameters

Parameter	Description	Value
Dump Destination	Destination to which data is dumped. **OBS**: After the streaming data is stored to DIS, it is then periodically imported to OBS. After the real-time file data is stored to DIS, it is imported to OBS immediately.	OBS
Task Name	Name of the dump task. The task name must be unique in a stream. A task name is 1 to 64 characters long. Only letters, digits, hyphens (-), and underscores (_) are allowed.	-

Parameter	Description	Value
Dump File Format	● text ● csv ● parquet ● carbon	Set this parameter as required.
Dump Bucket	Name of the OBS bucket used to store data from the DIS stream. The bucket name is created when you create a bucket on OBS.	Name of the bucket to be created when **Creating a DIS Stream**.
File Directory	Directory created in OBS to store files from the DIS stream. Different directory levels are separated by slashes. The value cannot start with a slash. This directory name is 0 to 50 characters long. This parameter is left blank by default.	-
Time Directory Format	Directory format based on the time. Data will be saved to the directory in the format of time layer under the dump file directory in the OBS bucket. For example, if the time directory is accurate to day, the data save path is "bucket name/dump file directory/year/month/day". Possible values: ● **N/A**: If this parameter is left blank, the time directory format will not be used. ● **yyyy**: year. ● **yyyy/MM**: year and month. ● **yyyy/MM/dd**: year, month, and day. ● **yyyy/MM/dd/HH**: year, month, day, and hour. ● **yyyy/MM/dd/HH/mm**: year, month, day, hour, and minute. You can only select but not enter a value in this field.	-

Parameter	Description	Value
Record Delimiter	Delimiter used to separate different dump records. Possible values: ● Comma (,) ● Semicolon (;) ● Vertical bar (|) ● Newline (\n) ● NULL You can only select but not enter a value in this field.	-
Offset	● **Latest**: Maximum offset, indicating that the latest data will be extracted. ● **Earliest**: Minimum offset, indicating that the earliest data will be extracted.	Latest
Dump Interval (s)	User-defined interval at which data is imported from the current DIS stream into OBS. If no data is pushed to the DIS stream during the current interval, no dump file package will be generated. Value range: 30s to 900s Unit: second Default value: 300	-

----End

Obtaining Authentication Information

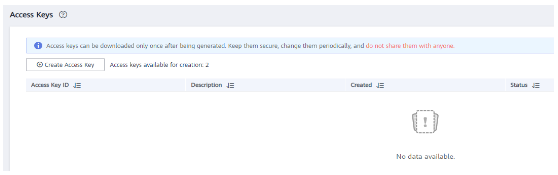
- Obtaining an Access Key ID/Secret Access Key (AK/SK)

  To obtain an access key, perform the following steps:

  a. Log in to the management console, move the cursor to the username in the upper right corner, and select **My Credentials** from the drop-down list.

  b. On the **My Credentials** page, choose **Access Keys**, and click **Create Access Key**. See **Figure 2-2**.

Figure 2-2 Clicking Create Access Key



Access Keys

Access keys can be downloaded only once after being generated. Keep them secure, change them periodically, and do not share them with anyone.

Access keys available for creation: 2

Access Key ID	Description	Created	Status

No data available.

c. Click **OK** and save the access key file as prompted. The access key file will be saved to your browser's configured download location. Open the **credentials.csv** file to view **Access Key Id** and **Secret Access Key**.

NOTE

- Only two access keys can be added for each user.
- To ensure access key security, the access key is automatically downloaded only when it is generated for the first time and cannot be obtained from the management console later. Keep them properly.

- Obtaining a project ID and account ID

  A project is a group of tenant resources, and an account ID corresponds to the current account. The IAM ID corresponds to the current user. You can view the project IDs, account IDs, and user IDs in different regions on the corresponding pages.

  a. Register with and log in to the management console.

  b. Hover the cursor on the username in the upper right corner and select **My Credentials** from the drop-down list.

  c. On the **API Credentials** page, obtain the account name, account ID, IAM username, and IAM user ID, and obtain the project and its ID from the project list.

- Obtaining the endpoint

  An endpoint is the **request address** for calling an API. Endpoints vary depending on services and regions. You can obtain the endpoints of the service from **Regions and Endpoints**.

Installing an Agent

Step 1 Obtain the Agent installation package at **https://dis-publish.obs-website.cn-north-1.myhuaweicloud.com/**.

Step 2 Decompress the **dis-agent-*X.X.X*.zip** package to the current folder.

----End

Preparing a Data Sample

Step 1 **Obtain** the sample data package.

Step 2 Decompress the package to the current folder.

----End

Configuring a DIS Agent

Step 1 Use the file manager to access the **conf** directory of the DIS agent program, for example, C:\dis-agent-*X.X.X*\conf.

Step 2 Open the **agent.yml** file using an editor and modify parameter values in the file based on the site requirements.

NOTE

- The configuration items and values must be separated by colons (:) and spaces.
- The **agent.yml** file is in Linux format. You are advised to use **Sublime Text** to edit the file.

Table 2-2 Parameters in the **agent.yml** file

Configuration Item	Mandatory	Description	Default Value
region	Yes	Region where DIS is deployed. For details about how to obtain it, see **Obtaining Authentication Information**.	-
ak	Yes	User's AK. For details about how to obtain it, see **Obtaining Authentication Information**.	-
sk	Yes	User's SK. For details about how to obtain it, see **Obtaining Authentication Information**.	-
projectId	Yes	Project ID specific to your region. For details about how to obtain it, see **Obtaining Authentication Information**.	-
endpoint	Yes	DIS gateway address. Format: https://DIS endpoint For details about how to obtain it, see **Obtaining Authentication Information**.	-

Configuration Item	Mandatory	Description	Default Value
body.serialize.type	No	Format of the DIS data packet to be uploaded. (non-raw data format) ● **json**: The DIS data packet is encapsulated in format of JSON. ● **protobuf**: The DIS data packet is encapsulated in binary format. After being encapsulated, the volume of the data packet is reduced by 1/3. This format is recommended when a massive amount of data is generated.	json
body.compress.enabled	No	Specifies whether to enable data compression.	false
body.compress.type	No	Data compression format selected when compression is enabled. Currently, the following compression formats are supported: **lz4**: a compression algorithm with a fast compression speed and high compression efficiency **zstd**: a new lossless compression algorithm with a fast compression speed and high compression ratio	lz4
PROXY_HOST	No	Proxy IP address. This parameter is mandatory when requests are sent through the proxy server.	-
PROXY_PORT	No	Proxy port.	80
PROXY_PROTOCOL	No	Proxy protocol. HTTP and HTTPS are supported.	http
PROXY_USERNAME	No	Proxy username.	-

Configuration Item	Mandatory	Description	Default Value
PROXY_PASSWORD	No	Proxy password.	-
[flows] The [flows] section presents information about the files that will be uploaded to DIS. The following upload mode is supported: **DISStream**: DIS Agent monitors text files continuously, collects incremental data in real time, parses the data by delimiter, and uploads it to DIS streams (source data type: BLOB, JSON, and CSV). **Table 2-3** describes configuration parameters. The **agent.yml** file provides example parameter settings.			

Table 2-3 DISStream configuration parameters

Configuration Item	Mandatory	Description	Default Value
DISStream	Yes	Name of the DIS stream. Parses the file content matching **filePattern** by delimiter and uploads the file to the stream.	-
filePattern	Yes	File monitoring path. Files in only one directory can be monitored. Directories cannot be monitored recursively. To monitor multiple directories, configure multiple DIS streams in **flows**. The file names can be matched by asterisk (*) • **/tmp/*.log**: Matches all files whose names end with **.log** in the **/tmp** directory. • **/tmp/access-*.log**: Matches all files whose names start with **access-** and end with **.log** in the **/tmp** directory. • In Windows, the example path is **D:\logs\*.log**.	-

Configuration Item	Mandatory	Description	Default Value
directoryRecursionEnabled	No	Specifies whether to search for a subdirectory. Possible values: ● **false**: Not to search for subdirectories recursively and match only files in the **root** directory. ● **true**: Search for all subdirectories recursively. For example, if **filePattern** is set to **/tmp/*.log**, **/tmp/one.log**, **/tmp/child/two.log**, and **/tmp/child/child/three.log** can be matched.	false
initialPosition	No	Initial position from which the file started to be monitored. Possible values: ● **END_OF_FILE**: After monitoring starts, the system does not parse the files that match **filePattern**. Instead, the newly added file or file content will be parsed by delimiter and uploaded to DIS. ● **START_OF_FILE**: All the files that match **filePattern** will be parsed by delimiter and uploaded to DIS based on the file modification time (from the earliest modified to the latest modified).	START_OF_FILE
maxBufferAgeMillis	No	The maximum time, in milliseconds, for which the agent buffers data before sending it to DIS. Unit: millisecond ● If the record queue is full with data waiting to be uploaded, data will be immediately uploaded to DIS. ● If the record queue is not full, files will be uploaded to DIS only after the specified period of time is reached.	5000
maxBufferSizeRecords	No	The maximum number of records for which the agent buffers data before sending it to DIS. If the number of records in a queue reaches the value, the data will be uploaded to DIS immediately.	500

Configuration Item	Mandatory	Description	Default Value
partitionKeyOption	No	Method for generating the partition key. Each record carries a partition key. Records with the same partition key are allocated to the same partition. Possible values: ● **RANDOM_INT**: The partition key is a random numeric string. Records with such a key are evenly distributed to each partition. ● **FILE_NAME**: The partition key is a file name string. Records with such a key is distributed to a specific partition. ● **FILE_NAME,RANDOM_INT**: The partition key is a combination of a file name string and a random numeric string, which are separated by comma (,). Records with such a key carries file names and are evenly distributed to all partitions.	RANDOM_INT
recordDelimiter	No	Delimiter used to separate records. Value range: any character that is enclosed in double quotation marks. The value cannot be empty. That is, this parameter cannot be set to "". **NOTE** If the value is a special character, use a backslash (\) to escape. For example, if the value is a quotation mark ("), set this parameter to **\"**. If the value is a backslash (\), set this parameter to \\. If the value is a control character, for example, STX, set this parameter to **\u0002**.	"\n"
isRemainRecordDelimiter	No	Specifies whether a delimiter is contained in records to be uploaded. Possible values: ● **true**: The delimiter is contained in records to be uploaded. ● **false**: The delimiter is not contained in records to be uploaded.	false

Configuration Item	Mandatory	Description	Default Value
isFileAppendable	No	Specifies whether the file contains additional content. Possible values: ● **true**: The file may contain additional content. Agent continuously monitors files. If content is added to a file, Agent parses the file by **recordDelimiter** and uploads records. In this case, ensure that the file ends with **recordDelimiter**. Otherwise, Agent considers that the content has not been added to the file and waits for **recordDelimiter** to be written. ● **false**: The file will not contain additional content. If the last row of the file does not end with **recordDelimiter**, Agent still uploads the file as the last record. After the upload is complete, Agent will delete or rename the file based on the configuration of **deletePolicy** and **fileSuffix**.	true
maxFileCheckingMillis	No	Maximum time for checking file changes. If the file size, modification time, and file ID do not change within this period of time, a complete file is generated and starts to be uploaded. Set this parameter based on the actual file change frequency to prevent an incomplete file from being uploaded. If the file is changed after being uploaded, it will be fully uploaded again. Unit: millisecond **NOTE** This parameter is available only when **isFileAppendable** is set to **false**.	5000
deletePolicy	No	Policy for deleting a file after the file content is uploaded. Possible values: ● **never**: The file will not be deleted after the file content is uploaded. ● **immediate**: The file will be deleted after the file content is uploaded. **NOTE** This parameter is available only when **isFileAppendable** is set to **false**.	never

Configuration Item	Mandatory	Description	Default Value
fileSuffix	No	Suffix of the file name that is added after the file content is uploaded. If the original file name is **x.txt** and **fileSuffix** is set to **.COMPLETED**, the name of the uploaded file is **x.txt.COMPLETED**. **NOTE** This parameter is available only when **isFileAppendable** is set to **false** and **deletePolicy** is set to **never**.	.COMPLETED
sendingThreadSize	No	Number of the threads to send data. By default, a single thread is used to send data. **NOTICE** If multiple threads are used, the following problems may occur: ● Data may not be sent in order. ● Some data is lost after the program stops abnormally and restarts.	1
fileEncoding	No	File encoding format. Possible values: **UTF8**, **GBK**, **GB2312**, and **ISO-8859-1**.	UTF8
resultLogLevel	No	Level of the calling result log generated after the DIS data sending API is called. ● **OFF**: Each API calling result is not logged. ● **INFO**: Each API calling result is logged at the INFO level. ● **WARN**: Each API calling result is logged at the WARN level. ● **ERROR**: Each API calling result is logged at the ERROR level.	INFO

Step 3 (Optional) Use the Windows notepad to modify the **agent.yml** file and select **UTF-8** when saving the file.

1. Choose **File > Save As**.
2. In the **Save As** dialog box, set **Code** to **UTF-8**.
3. Click **Save**. The confirmation dialog box is displayed.
4. Click **Yes**.

----End

Starting the DIS Agent

Step 1 Use the file manager to access the **bin** directory of the DIS agent program, for example, C:\dis-agent-*X.X.X*\bin.

Step 2 Double-click the **start-dis-agent.bat** file. If the following information is displayed in the console window, the DIS agent is successfully started:

```
[INFO ] (main) com.bigdata.dis.agent.Agent Agent: Startup completed in XXX ms.
```

After the DIS agent is started, files are uploaded immediately and logs are continuously printed. If no ERROR log is found, files are uploaded without error.

If logs are not printed within 30 seconds and the information similar to the following is displayed, the upload is completed.

```
Agent: Progress: [0 records (0 bytes) / 10 files (32573229 bytes)] parsed, and [0  records / 10 files] sent
successfully to destinations. Uptime: 30146ms
```

----End

Viewing Uploaded Files on OBS

Step 1 Log in to the OBS console.

Step 2 In the navigation tree, choose **Bucket List**.

Step 3 In the **Bucket Name** column, click the bucket name configured in **Creating a DIS Stream**.

Step 4 On the displayed page, click the **Object** tab in the navigation tree to view the uploaded files.

----End

Creating a Database

Step 1 On the homepage of the management console, choose **Service List** > **Analytics** > **Data Lake Insight**.

Step 2 Create a demo database. On the DLI console, click **Create Job** in the **SQL Job** area. The SQL job editor is displayed.

Step 3 On the left of the SQL job editor, click 🗄 and then ⊕ to create a database.

NOTE

The **default** database is a built-in database. You cannot create a database named **default**.

----End

Creating an OBS Table

Step 1 Choose the demo database and enter the following SQL statement in the editing area:

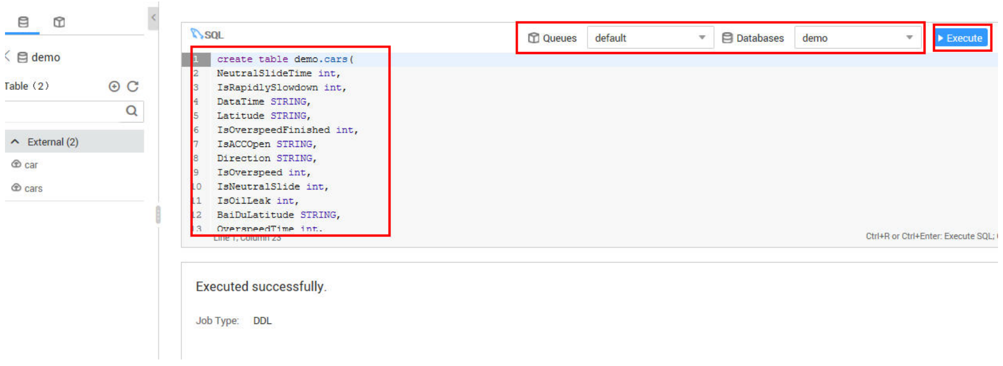
```
create table demo.cars(
  NeutralSlideTime int,
  IsRapidlySlowdown int,
  DataTime STRING,
  Latitude STRING,
  IsOverspeedFinished int,
  IsACCOpen STRING,
  Direction STRING,
  IsOverspeed int,
  IsNeutralSlide int,
  IsOilLeak int,
  BaiDuLatitude STRING,
  OverspeedTime int,
  IsRapidlySpeedup int,
  DeviceID STRING,
  Mileage STRING,
  Longitude STRING,
  Velocity double,
  IsNeutralSlideFinished int,
  IsFatgueDriving int,
  Carnum STRING,
  BaiDuLongitude STRING,
  BaiDuAdress STRING,
  IsHthrottleStop int,
  ReceiveTime STRING,
  Altitude STRING
) USING csv OPTIONS (path "obs://......")
```

NOTE

Change **csv** in the SQL statement to the format of the file to be dumped to the OBS bucket, and change the OBS path to the actual path for storing data.

Step 2 Click **Execute** to create a table.

Figure 2-3 Creating a table



```
create table demo.cars(
NeutralSlideTime int,
IsRapidlySlowdown int,
DataTime STRING,
Latitude STRING,
IsOverspeedFinished int,
IsACCOpen STRING,
Direction STRING,
IsOverspeed int,
IsNeutralSlide int,
IsOilLeak int,
BaiDuLatitude STRING,
OverspeedTime int,
```

Executed successfully.

Job Type: DDL

The fields in the table are described as follows:

Column Name	Data Type	Description
DeviceID	string	Device ID.
DataTime	string	Data time.
ReceiveTime	string	Time of receipt.
IsACCOpen	boolean	Whether to enable ACC.
Longitude	double	Longitude.
Latitude	double	Latitude.
Velocity	int	Velocity.
Direction	string	Direction.
Altitude	string	Altitude.
Mileage	long	Mileage.

Column Name	Data Type	Description
BaiDuLongitude	double	Longitude of the Baidu map.
BaiDuLatitude	double	Longitude of the Baidu map.
BaiDuAdress	string	Address of the Baidu map.
Carnum	string	License plate number.
IsRapidlySpeedup	int	Rapid acceleration.
IsRapidlySlowdown	int	Rapid deceleration.
IsNeutralSlide	int	Neutral taxiing.
IsNeutralSlideFinished	int	Neutral taxiing finished.
NeutralSlideTime	int	Time length of neutral taxiing. Unit: second.
IsOverspeed	int	Overspeed.
IsOverspeedFinished	int	Overspeed finished.
OverspeedTime	int	Overspeed time length. Unit: second.
IsFatgueDriving	int	Fatigue driving.
IsHthrottleStop	int	Stepping on the gas pedal when the vehicle is not moving.

----End

Querying a Data Sample

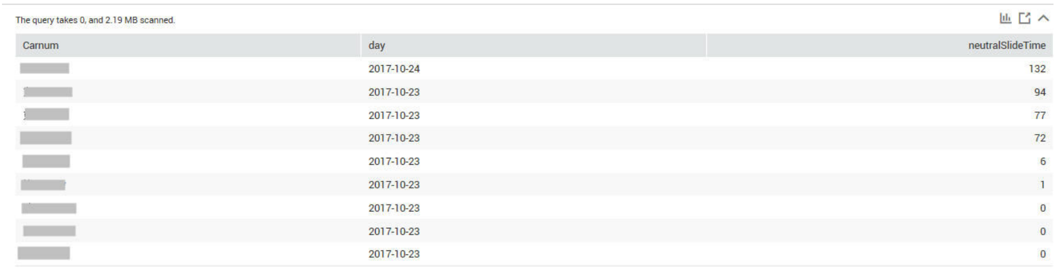
- Acceleration statistics

```
select
  Carnum,
  day,
  IFNULL(sum(isRapidlySpeedup), 0) as rapidlySpeedupTimes
from
  (
    select
      *,
      cast(DataTime as date) as day
    from
      demo.cars
  ) t1
group by
  Carnum,
  day
```

Querying Results

Figure 2-4 shows the execution results of the query statements.

Figure 2-4 Speeding statistics



The query takes 0, and 2.19 MB scanned.

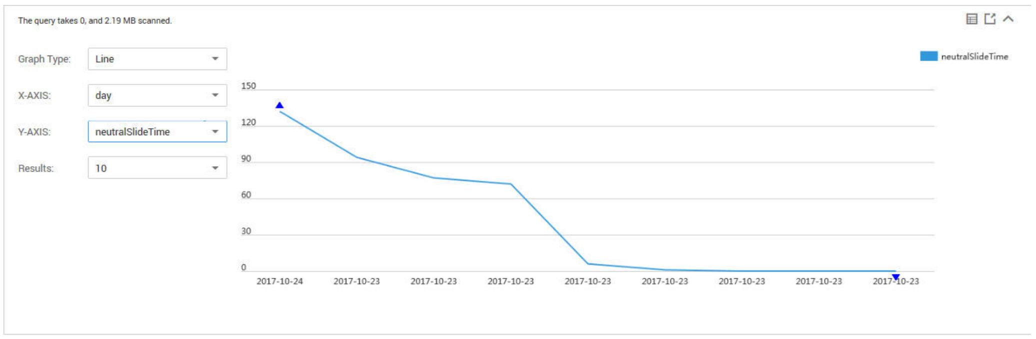
Carnum	day	neutralSlideTime
[redacted]	2017-10-24	132
[redacted]	2017-10-23	94
[redacted]	2017-10-23	77
[redacted]	2017-10-23	72
[redacted]	2017-10-23	6
[redacted]	2017-10-23	1
[redacted]	2017-10-23	0
[redacted]	2017-10-23	0
[redacted]	2017-10-23	0

Click **Graphical Result** to display the query result in a graph. Set **Chart Type**, **X Axis**, and **Y Axis**. Then the speeding statistics diagram is displayed, as shown in Figure 2-5.

NOTE

- If no column of the numeric type is displayed in the execution result, the result cannot be represented in charts.
- The chart types include the bar chart, line chart, and fan chart.
- In the bar chart and line chart, the X axis can be any column, while the Y axis supports only columns of the numeric type. The fan chart displays the corresponding legends and indicators.

Figure 2-5 Speeding statistics



The query takes 0, and 2.19 MB scanned.

Graph Type: Line
X-AXIS: day
Y-AXIS: neutralSlideTime
Results: 10