Data Ingestion Service

User Guide

Issue	01
Date	2023-06-20

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions

HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 IAM Permissions Management

1.1 Creating a User and Granting Permissions

This chapter describes how to use **IAM** to implement fine-grained permissions control for your DIS resources. With IAM, you can:

- Create IAM users for employees using the HUAWEI CLOUD account based on the organizational structure of your enterprise. Each IAM user has their own security credentials, providing access to DIS resources.
- Grant only the permissions required for users to perform a task.
- Entrust a HUAWEI CLOUD account or cloud service to perform professional and efficient O&M on your DIS resources.

If your HUAWEI CLOUD account does not need individual IAM users, then you may skip over this chapter.

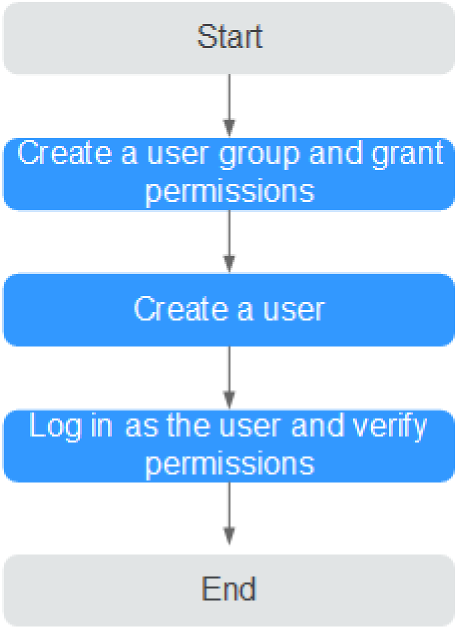
This section describes the procedure for granting permissions. **Figure 1-1** shows the procedure.

Prerequisites

Learn about the permissions (see **Permissions Management**) supported by DIS and choose policies or roles according to your requirements.

Process Flow

Figure 1-1 Process for granting DIS permissions


Start
Create a user group and grant permissions
Create a user
Log in as the user and verify permissions
End

1. **Create a user group and assign permissions** to it.

   Create a user group on the IAM console and assign the DIS Operator policy to the group.

2. **Create an IAM user**.

   Create a user on the IAM console and add the user to the group created in **1**.

3. **Log in** and verify permissions.

   Log in to the console by using the user created, and verify that the user has the granted permissions.

   – Choose **Service List** > **Data Ingestion Service**. On the DIS console, create a stream. If no message appears indicating insufficient permissions to perform the operation, the **DIS Operator** policy has already taken effect.

   – Choose any other service in **Service List**. If a message appears indicating that you have insufficient permissions to access the service, the **DIS Operator** policy has already taken effect.

2 Getting Started

2.1 General Procedure

The following is the general procedure for using DIS:

Step 1: Creating a DIS Stream

You need to create a stream before using DIS.

Step 2: Preparing a DIS Application Development Environment

Before developing a DIS application, install an application development tool, and import your SDK package and sample project into the development environment.

Step 3: Sending Data to DIS

Write a producer application and run it to send data to the cloud. The DIS stream information can be viewed on the DIS console.

Step 4: Obtaining Data from DIS

Write a consumer application and run it to retrieve data from the cloud.

2.2 Step 1: Creating a DIS Stream

You can create a DIS stream on the DIS management console.

Procedure

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click 📍 in the upper left corner of the page and select a region and project.

Step 3 Click **Buy Stream** and set related parameters.

Table 2-1 Stream parameters

Parameter	Description	Example
Billing Mode	Pay-per-use	Pay-per-use
Region	Physical location of the cloud service. You can select a different region from the drop-down list.	-
Basic Information		
Stream Name	Name of the DIS stream to be created. A stream name is 1 to 64 characters long. Only letters, digits, hyphens (-), and underscores (_) are allowed.	dis-Tido
Stream Type	● **Common**: Each partition supports a maximum read speed of 2 MB/s and a maximum write speed of 1 MB/s. ● **Advanced**: Each partition supports a maximum read speed of 10 MB/s and a maximum write speed of 5 MB/s.	-
Partitions	Partitions are the base throughput unit of a DIS stream.	5

Parameter	Description	Example
Partition Calculator	Calculator used to calculate the estimated number of partitions based on the information you entered. 1. Click **Partition Calculator**. 2. In the **Partition Calculator** dialog box, configure the **Average Record Size (KB)**, **Max. Records Written**, and **Consumer Applications** parameters. The **Estimated Partitions** field then displays the recommended number of partitions. The value of this field cannot be modified. **NOTE** Partition calculation formulas: – Based on the traffic (the final value must be rounded up): Common stream: Average record size x (1 + 20%) x Maximum records written/ (1 x 1024 KB) (20% is the reserved partition percentage.) Advanced stream: Average record size x (1 + 20%) x Maximum records written/ (5 x 1024 KB) (20% is the reserved partition percentage.) – Based on the consumer program quantity (the final value must be rounded up): (Number of consumer programs/2) x Number of partitions calculated based on the traffic (The result of the number of consumer programs/2 must reserve two decimals.) The largest value among the values calculated based on the previous three formulas is considered as the estimated partition value. 3. Click **Use Estimated Value**. The estimated value is automatically used as the value of **Partitions**.	-
Data Retention (hours)	The maximum number of hours for which data can be preserved in DIS. Data will be deleted when the retention period expires. Value range: an integer ranging from 24 to 72.	24

Parameter	Description	Example
Source Data Type	● **BLOB**: a collection of binary data stored as a single entity in a database management system. If **Source Data Type** is set to **BLOB**, the supported **Dump Destination** can be **OBS** or **MRS**. ● **JSON**: an open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types. If **Source Data Type** is set to **JSON**, the supported **Dump Destination** can be **OBS**, **MRS**, **DLI**, or **DWS**. ● **CSV**: a simple text format for storing tabular data in a plain text file. If **Source Data Type** is set to **CSV**, the supported **Dump Destination** can be **OBS**, **MRS**, **DLI**, or **DWS**.	JSON
Auto-Scaling	You can choose to enable or disable auto-scaling when creating a stream. You can click ● or ○ to disable or enable auto-scaling.	**NOTE** You can choose whether to enable auto-scaling when creating a stream. You can also modify the auto-scaling attributes for a created stream.
Auto-Scale Down To	Lower limit for automatic scale-down. The number of target partitions for automatic scale-down must be greater than or equal to the lower limit.	-
Auto-Scale Up To	Upper limit for automatic scale-up. The number of target partitions for automatic scale-up must be smaller than the lower limit.	-
Data Delimiter	Data delimiter when **Source Data Type** is **CSV**.	-

Parameter	Description	Example
Schema	Whether to create a schema when creating a stream. This parameter is available when **Source Data Type** is **JSON** or **CSV**. You can click ● or ○ to disable or enable the schema configuration. **NOTE** If no data schema is created when a stream is created, you can also create it later after the stream is created. Create a schema on the **Stream Management** page. For details, see **Managing a Source Data Schema**.	You can create a schema only when the source data type is set to **JSON** or **CSV**.
Source Data Schema	You can enter or import source data samples in JSON or CSV format. For details, see **Managing a Source Data Schema**. 1. In the left text box, enter a JSON or CSV source data sample or click Import Source Data Sample to import a source data sample. 2. In the left text box, click 🗑 to delete your entered or imported source data sample. 3. In the left text box, click ▶ to generate an Avro schema in the right text box according to the source data sample. 4. In the right text box, click 🗑 to delete the generated Avro schema. 5. In the right text box, click ✎ to modify the generated Avro schema.	This parameter is mandatory only when **Schema** is set to **Enable**.
Enterprise Project	Configure the enterprise project to which streams belong. You can configure this parameter only when the Enterprise Management service is enabled. The default value is **default**. An enterprise project facilitates project-level management and grouping of cloud resources and users. You can select the default enterprise project (**default**) or other existing enterprise projects. To create an enterprise project, log in to the Enterprise Management console. For details, see the *Enterprise Management User Guide*.	-

Parameter	Description	Example
Configure	Click **Configure now**. The **Tag** parameter is displayed. For details about how to add tags, see **Managing Stream Tags**.	-
Skip	No advanced settings need to be configured.	-
Tag	Identifier of the stream. Adding tags to streams can help you identify and manage your stream resources.	-

Step 4 Click **Next**. The **Details** page is displayed.

Step 5 Click **Submit**.

----End

2.3 Step 2: Preparing a DIS Application Development Environment

Before developing DIS applications, prepare an application development environment, and then obtain a software development kit (SDK) and sample project and import them to the development environment.

Prerequisites

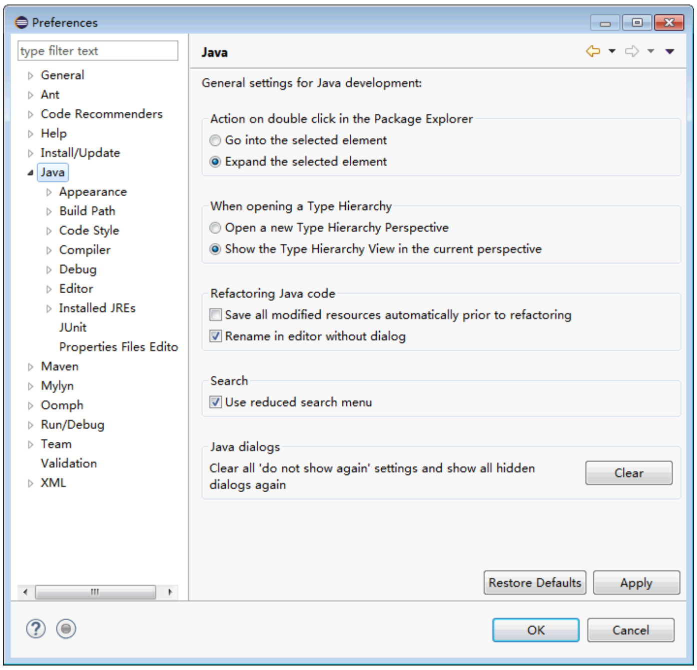
- JDK 1.8 or later has been installed.
- Eclipse has been installed.

Procedure

Step 1 Configure a JDK using Eclipse.

1. Start Eclipse and choose **Window** > **Preferences**. The **Preferences** dialog box is displayed.
2. In the navigation tree, choose **Java**. On the **Java** page, configure general settings for Java development and then click **OK**.

Figure 2-1 Preferences



Java

General settings for Java development:

Action on double click in the Package Explorer

○ Go into the selected element

● Expand the selected element

When opening a Type Hierarchy

○ Open a new Type Hierarchy Perspective

● Show the Type Hierarchy View in the current perspective

Refactoring Java code

□ Save all modified resources automatically prior to refactoring

☑ Rename in editor without dialog

Search

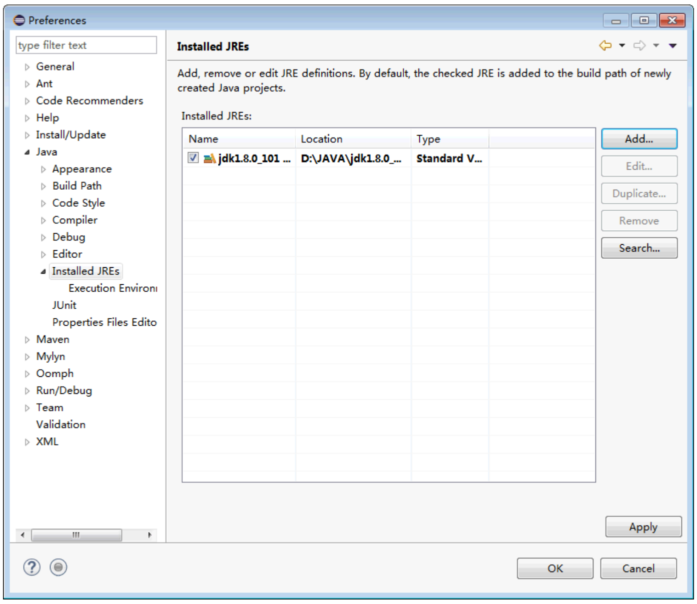
☑ Use reduced search menu

Java dialogs

Clear all 'do not show again' settings and show all hidden dialogs again

3. In the navigation tree, choose **Java** > **Installed JREs**.
   - Ensure that configured JDK environmental variables are displayed on the **Installed JREs** page. Then go to **Step 1.3.a**.
   - To configure different variables for different versions of JDK, perform **Step 1.3.b** to **Step 1.3.d**.

Figure 2-2 Installed JREs



Installed JREs

Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.

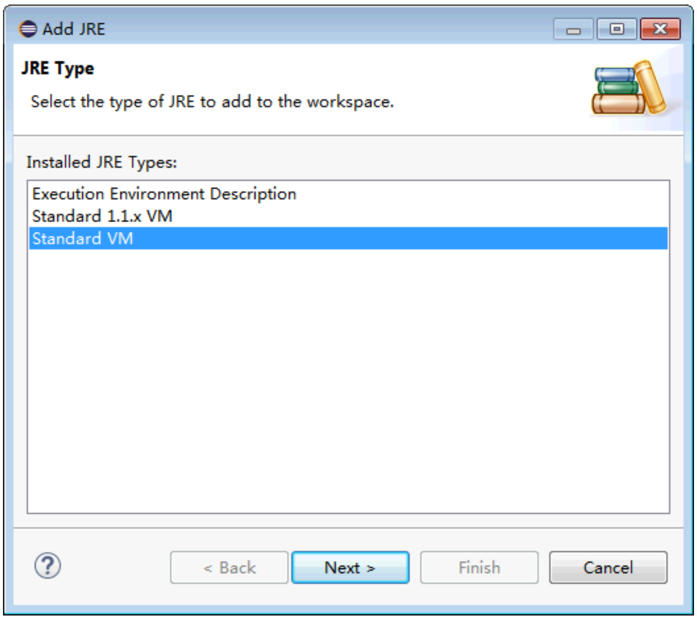
Installed JREs:

Name	Location	Type
☑ jdk1.8.0_101 ...	D:\JAVA\jdk1.8.0_...	Standard V...

a. Select the installed JDK and click **OK**.

b. Click **Add**. The **Add JRE** dialog box is displayed.

Figure 2-3 JRE Type



Add JRE

JRE Type

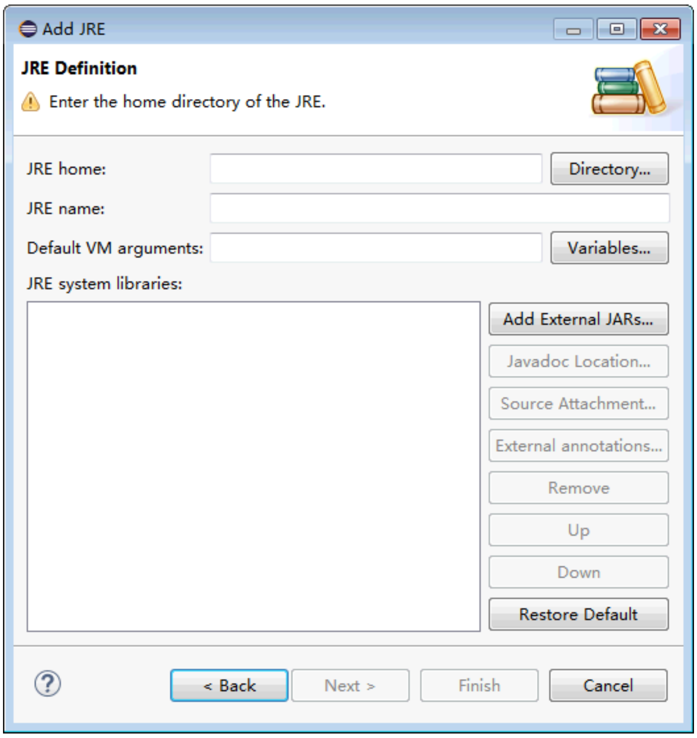
Select the type of JRE to add to the workspace.

Installed JRE Types:

Execution Environment Description
Standard 1.1.x VM
Standard VM

c. Select a JRE type and click **Next**.

Figure 2-4 JRE Definition



Add JRE

JRE Definition

⚠ Enter the home directory of the JRE.

JRE home:

JRE name:

Default VM arguments:

JRE system libraries:

d. Configure the basic information about JDK and click **Finish**.

- JRE home: JDK installation path.
- Default VM arguments: JDK running parameters.

Step 2 Download resource packages.

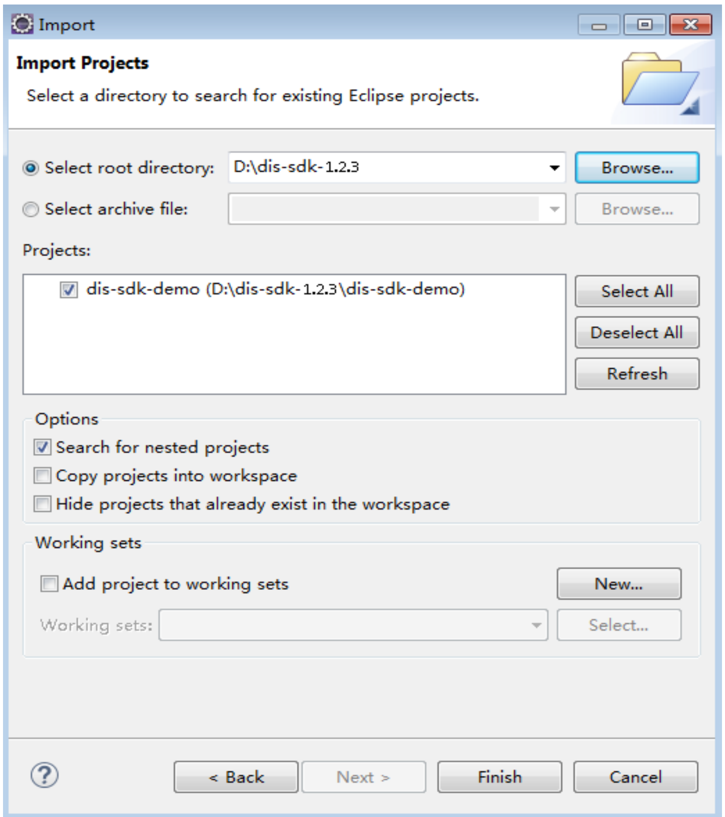
Download the DIS Java SDK from **https://github.com/huaweicloud/huaweicloud-sdk-java-dis**.

Obtain **huaweicloud-sdk-dis-java-*X.X.X*.zip** from the **DIS SDK**. The package contains the demo package of the sample project.

Step 3 Import the Eclipse project.

1. Start Eclipse. Choose **File** > **Import**. The **Import** dialog box is displayed.
2. Choose **Maven** > **Existing Maven Projects**, and click **Next**. The **Import** dialog box is displayed.
3. Click **Browse** and select a save location for the **dis-sdk-demo** sample project. In the **Projects** area, select a sample project.

Figure 2-5 Importing a project



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: D:\dis-sdk-1.2.3

○ Select archive file:

Projects:

☑ dis-sdk-demo (D:\dis-sdk-1.2.3\dis-sdk-demo)

Options

☑ Search for nested projects

□ Copy projects into workspace

□ Hide projects that already exist in the workspace

Working sets

□ Add project to working sets

Working sets:

4. Click **Finish** to import the project.

Step 4 Configure the demo project.

1. Set the project code to **UTF-8**.
   a. In the navigation tree, right-click the required project under **Project Explorer** and choose **Properties** from the shortcut menu. The **Properties for dis-sdk-demo** dialog box is displayed.
   b. In the navigation tree, choose **Resource**. The **Resource** page is displayed in the right pane.
   c. In the **Other** drop-down list, select **UTF-8**.
   d. Click **Apply and Close**.

2. Add the JDK.

   a. In the navigation pane, choose **Project Explorer**. Right-click the chosen project and choose **Properties** from the shortcut menu.

   b. In the navigation tree, choose **Java Build Path**. The **Java Build Path** page is displayed in the right pane.

   c. Click the **Libraries** tab, and then click **Add Library**. The **Add Library** dialog box is displayed.

   d. Select **JRE System Library** and click **Next**. Verify that the version of **Workspace default JRE** is **jdk1.8** or later.

   e. Click **Finish** to exit the **Add Library** dialog box.

   f. Click **Apply and Close**.

Step 5 Initialize a DIS client sample. For details about **endpoint**, **ak**, **sk**, **region**, and **projectId**, see **Obtaining Authentication Information**.

----End

2.4 Step 3: Sending Data to DIS

Function

Local data is continuously uploaded to DIS.

NOTE

Data can be stored in MRS, DIS, OBS, and DLI. For how to configure a storage location, see **Creating a Dump Task**.

The maximum number of days for DIS to preserve data cannot exceed **Data Retention (days)**.

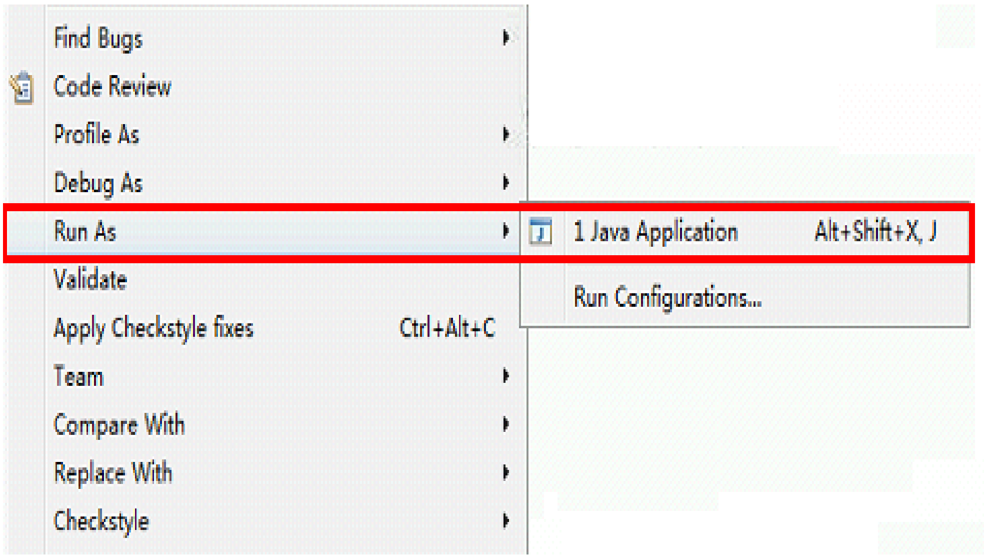
Sample Code

The example code file is the **ProducerDemo.java** file in the **\dis-sdk-demo\src\main\java\com\bigdata\dis\sdk\demo** directory decompressed from the **huaweicloud-sdk-dis-java-*X.X.X*.zip** package. The compression package is downloaded from the **DIS SDK**.

Running the Producer Program

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

Figure 2-6 Running a producer 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:

```
14:40:20.090 [main] INFOcom.bigdata.dis.sdk.DISConfig - get from classLoader
14:40:20.093 [main] INFODEMOT - ========== BEGIN PUT ============
14:40:21.186 [main] INFOcom.bigdata.dis.sdk.util.config.ConfigurationUtils - get from classLoader
14:40:21.187 [main] INFOcom.bigdata.dis.sdk.util.config.ConfigurationUtils - propertyMapFromFile size : 2
14:40:22.092 [main] INFOcom.bigdata.dis.sdk.demo.ProducerDemo - Put 3 records[3 successful / 0 failed].
14:40:22.092 [main] INFOcom.bigdata.dis.sdk.demo.ProducerDemo - [hello world.] put success, partitionId
[shardId-0000000000], partitionKey [964885], sequenceNumber [0]
14:40:22.092 [main] INFOcom.bigdata.dis.sdk.demo.ProducerDemo - [hello world.] put success, partitionId
[shardId-0000000000], partitionKey [910960], sequenceNumber [1]
14:40:22.092 [main] INFOcom.bigdata.dis.sdk.demo.ProducerDemo - [hello world.] put success, partitionId
[shardId-0000000000], partitionKey [528377], sequenceNumber [2]
14:40:22.092 [main] INFOcom.bigdata.dis.sdk.demo.ProducerDemo - ========== PUT OVER ============
```

2.5 Step 4: Obtaining Data from DIS

Function

You can retrieve data from DIS when needed.

Sample Code

The example code file is the **ConsumerDemo.java** file in the **\dis-sdk-demo\src\main\java\com\bigdata\dis\sdk\demo** directory decompressed from the **huaweicloud-sdk-dis-java-***X.X.X***.zip** package. The compression package is downloaded from the **DIS SDK**.

Running the Consumer Application

If information similar to the following appears, data has been successfully retrieved from DIS:

```
14:55:42.954 [main] INFOcom.bigdata.dis.sdk.DISConfig - get from classLoader
14:55:44.103 [main] INFOcom.bigdata.dis.sdk.util.config.ConfigurationUtils - get from classLoader
```

```
14:55:44.105 [main] INFOcom.bigdata.dis.sdk.util.config.ConfigurationUtils - propertyMapFromFile size : 2
14:55:45.235 [main] INFOcom.bigdata.dis.sdk.demo.ConsumerDemo - Get stream
streamName[partitionId=0] cursor success :
eyJnZXRJdGVyYXRvclBhcmFtIjp7InN0cmVhbS1uYW1lIjoiZGlzLTEzbW9uZXkiLCJwYXJ0aXRpb24taWQiOiIwIiwiY
3Vyc29yLXR5cGUiOiJBVF9TRVFVRU5DRV9OVU1CRVIiLCJzdGFydGluZy1zZXF1ZW5jZS1udW1iZXIiOiIxMDY4O
TcyIn0sImdlbmVyYXRlVGltZXN0YW1wIjoxNTEzNjY2NjMxMTYxfQ
14:55:45.305 [main] INFOcom.bigdata.dis.sdk.demo.ConsumerDemo - Get Record [hello world.],
partitionKey [964885], sequenceNumber [0].
14:55:45.305 [main] INFOcom.bigdata.dis.sdk.demo.ConsumerDemo - Get Record [hello world.],
partitionKey [910960], sequenceNumber [1].
14:55:46.359 [main] INFOcom.bigdata.dis.sdk.demo.ConsumerDemo - Get Record [hello world.],
partitionKey [528377], sequenceNumber [2].
```

2.6 Obtaining Authentication Information

Obtaining AK/SK

Access Key ID/Secret Access Key (AK/SK) is created on Identity and Access Management (IAM) to authenticate calls to application programming interfaces (APIs) on the public cloud. To obtain an AK/SK pair, choose **My Credentials** > **Access Keys**.

Obtaining Project ID

A project is a group of tenant resources. To view the project IDs of different regions, choose **My Credentials** > **API Credentials**.

Obtaining Region Information and Endpoint Information

For details about regions and endpoints, see **Regions and Endpoints**.

2.7 Interconnecting with OBS

Introduction

DIS can upload data to Object Storage Service (OBS).

Prerequisites

An IAM agency has been created by following the procedure in **Creating an IAM Agency**. This IAM agency entrusts DIS to access your OBS resources.

Data Dumping

You can set **Dump Bucket** when **creating a dump task**. If Dump Destination is set to **OBS**, DIS periodically imports data from DIS streams to OBS.

2.8 Creating an IAM Agency

Introduction

If you choose to dump data from DIS to OBS, MRS, or DLI, create an IAM agency that grants DIS permissions to access OBS, MRS, or DLI.

Creating an IAM Agency

Step 1 Log in to the management console.

Step 2 Click **Service List**. Under **Management & Deployment**, select **Identify and Access Management**.

Step 3 Select **Agencies** in the navigation tree pane, and click **Create Agency**.

Step 4 Configure agency parameters and click **OK**.

Table 2-2 Agency parameters

Parameter	Description
Agency Name	Name of the agency to be created. The value of this parameter is 1 to 64 characters long and cannot be left unspecified.
Agency Type	Type of the agency to be created. This parameter must be set to **Cloud service**.
Cloud Service	Click **Select** next to **Cloud Service**. In the **Select Cloud Service** dialog box, select **DIS** and click **OK**.
Validity Period	Select **Unlimited**. **NOTE** Currently, this parameter must be set to **Unlimited**. Using another value may result in authorization failures.
Description	Agency description. The entered description cannot exceed 255 characters.
Permissions	To modify agency policies, click **Modify** in the **Operation** column. In the **Available Policies** area, select your required policy and click **OK**. **NOTE** After an agency is created, its policies cannot be modified.

----End

3 Managing DIS Streams

3.1 Listing DIS Streams

The **Stream Management** page displays all DIS streams created. After clicking a stream, you can view the following information about this stream:

- **Name/ID**: Unique name of the DIS stream to be created. A stream name is 1 to 64 characters long. Only letters, digits, hyphens (-), and underscores (_) are allowed.
- **Status**: Stream status.
- **Stream Type**: **Common** and **Advanced**.
  - **Common**: Each partition supports a maximum read speed of 2 MB/s and a maximum write speed of 1 MB/s or 1,000 records/s.
  - **Advanced**: Each partition supports a maximum read speed of 10 MB/s and a maximum write speed of 5 MB/s or 2,000 records/s.
- **Partitions**: The number of partitions into which data records in the newly created DIS stream will be distributed. Multiple partitions of a stream can concurrently transmit data to improve efficiency.
- **Source Data Type**: BLOB, JSON, and CSV.
- **Data Retention (hours)**: The maximum number of hours for DIS to preserve data. Data will be deleted when the retention period expires. Value range: an integer ranging from 24 to 72. Unit: hour
- **Created**: Time at which the DIS stream is created. The creation time is in the *yyyy/MM/dd HH:mm:ss* **GMT** format. For example, 2017/05/09 08:00:00 GMT +08:00.
- **Billing Mode**: Only pay-per-use is supported.
- **Operation**: Supported operations include **Change Source Data Type**, **Delete**, and **View Dump Task**.

3.2 Viewing Stream Monitoring Metrics

You can view stream monitoring information on the console and monitor the data consumed by applications in the stream.

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click 📍 in the upper left corner and select a region and a project.

Step 3 In the navigation tree on the left, choose **Stream Management**.

Step 4 In the stream list, click the name of the DIS stream whose monitoring metrics you want to view.

Step 5 On the **Monitoring** page, click the **Streams** or **Partitions** tab to view stream or partition monitoring metrics. **Table 3-1** describes the monitoring parameters. For details about basic stream information, see **Table 2-1**.

Table 3-1 DIS monitoring information

Parameter	Description
Time Range	● Monitoring time range. Values: – 1h – 3h – 12h ● You can customize the time range for viewing monitoring information. – Click 📅 next to **Custom**, and then set the start time and end time. – The end time cannot be later than the current system time. – The difference between the start time and end time cannot exceed 72 hours.
Partitions	
Partition ID	ID of the partition. It starts from 0 by default. Select any of the following values from the **Partition ID** drop-down.
Data Rate (KB/s)	Rates at which data is sent to and retrieved from the chosen partition within the specified time range. Unit: KB/s
Records Per Second	The number of records sent to and retrieved from the chosen partition within the specified time range.
Streams	
Data Rate (KB/s)	Rates at which data is sent to and retrieved from the chosen DIS stream within the specified time range. Unit: KB/s
Records Per Second	The number of records sent to and retrieved from the chosen DIS stream within the specified time range.

Parameter	Description
Successful Requests Per Second	The number of PutRecords and GetRecords requests successfully fulfilled within the specified time range.
Throttled Requests Per Second	The number of PutRecords and GetRecords requests rejected within the specified time range due to flow control.
Average Request Processing Time (ms)	The average amount of time spent in processing a PutRecords or GetRecords request.

Step 6 In the upper right corner of the graph, click to enlarge the graph for viewing details.

----End

3.3 Changing a Source Data Type

The source data schema is required for data conversion of a specific dump task in a stream. If it is incorrectly configured, data conversion fails and the dump task is abnormal. You can configure the source data schema when creating a stream or when creating a dump task. You can modify the configured source data schema on the stream details page.

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click in the upper left corner of the page and select a region and project.

Step 3 In the navigation tree on the left, choose **Stream Management**.

1. Click the name of a stream to access its details page.
2. Click next to **Source Data Type** and select a desired source data type from the drop-down list. Alternatively, in the **Operation** column of the stream for which you want to change its source data type, choose **More** > **Change Source Data Type**. In the displayed dialog box, change the source data type.

NOTE

- You can change the source data type for a stream whose **Source Data Type** is **BLOB**, **JSON**, or **CSV** only when the stream has no dump tasks.
- After you change the source data type of a stream for which you have configured source data schema, the existing source data schema will become invalid and cannot be recovered. You need to reconfigure the source data schema for the stream.

----End

3.4 Managing a Source Data Schema

A source data schema is a user's JSON or CSV data sample used to describe the JSON or CSV data format. For example, DIS can generate an Avro schema based on the JSON or CSV data sample and convert the JSON or CSV data uploaded to a stream to the Parquet or CarbonData format.

Three entrances are available for creating a source data schema:

- Enable **Schema** when creating a stream. For details, see **Figure 3-1**.
- Keep **Schema** disabled when creating a stream. After the stream is created, choose **Stream Management** in the navigation tree and click the created stream. Click **Create Source Data Schema** next to **Source Data Type**. For details, see **Figure 3-2**.
- Keep **Schema** disabled when creating a stream. After the stream is created, choose **Stream Management** in the navigation tree and click the created stream. On the **Dump Tasks** tab page, click **Create Dump Task**. On the displayed page, create a source data schema. For details, see **Figure 3-3**.

Figure 3-1 Entrance 1 for creating a schema

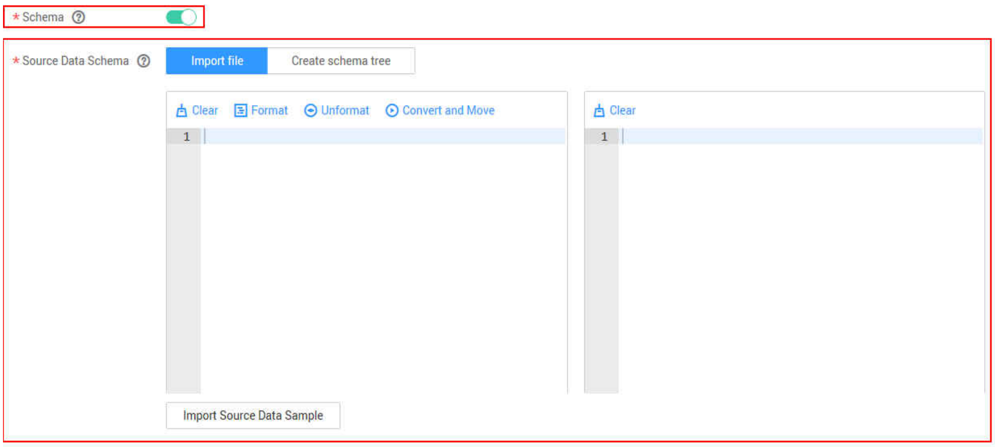
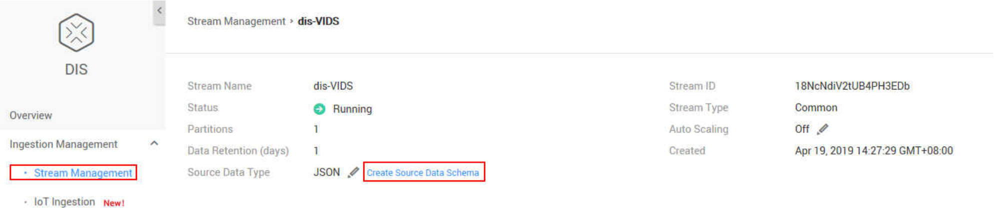


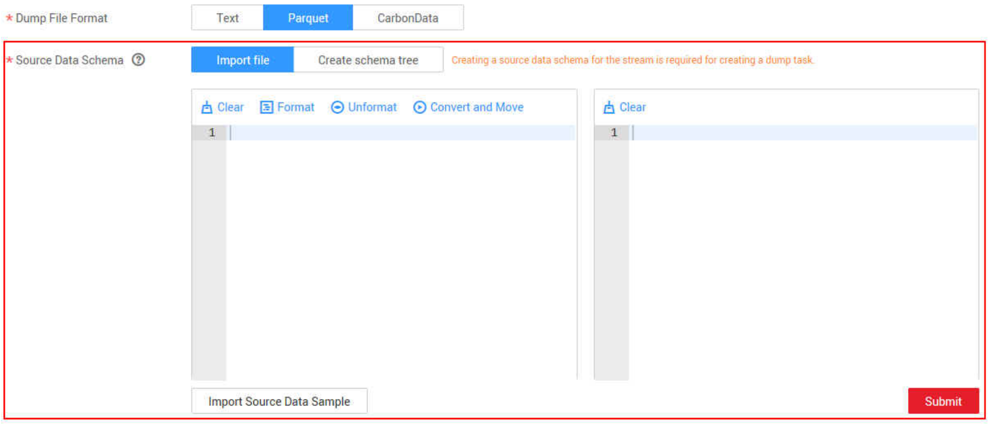
Figure 3-2 Entrance 2 for creating a schema



Stream Management › **dis-VIDS**

Stream Name	dis-VIDS	Stream ID	18NcNdiV2tUB4PH3EDb
Status	Running	Stream Type	Common
Partitions	1	Auto Scaling	Off
Data Retention (days)	1	Created	Apr 19, 2019 14:27:29 GMT+08:00
Source Data Type	JSON Create Source Data Schema		

Figure 3-3 Entrance 3 for creating a schema



* Dump File Format: Parquet

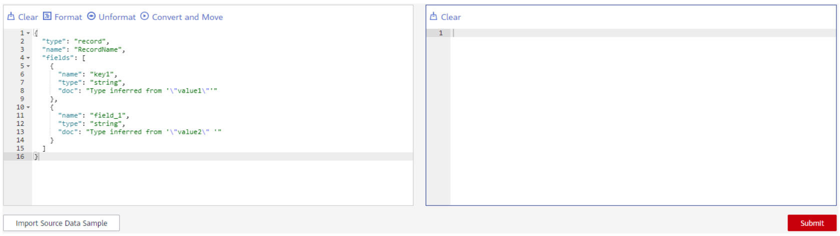
* Source Data Schema: Creating a source data schema for the stream is required for creating a dump task.

Creating a Schema for Source Data by Importing Files

Use the following method to create a source data schema:

Step 1 When configuring **Source Data Schema**, click **Import File**.

Step 2 In the left text box, enter a JSON or CSV source data sample or click Import Source Data Sample to import a source data sample. Example:

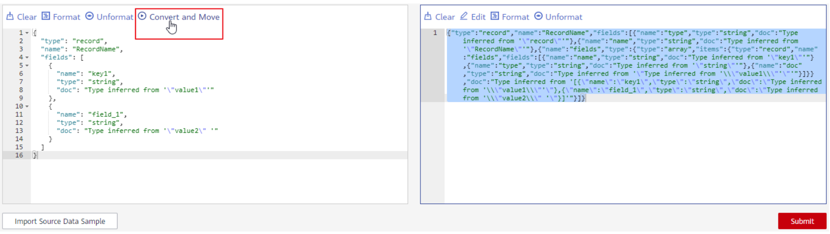


```
{
  "type": "record",
  "name": "RecordName",
  "fields": [
    {
      "name": "key1",
      "type": "string",
      "doc": "Type inferred from '\"value1\"'"
    },
    {
      "name": "field_1",
      "type": "string",
      "doc": "Type inferred from '\"value2\" '"
    }
  ]
}
```

NOTE

When importing source data samples, you can import only .txt, .json, .csv, and .java files.

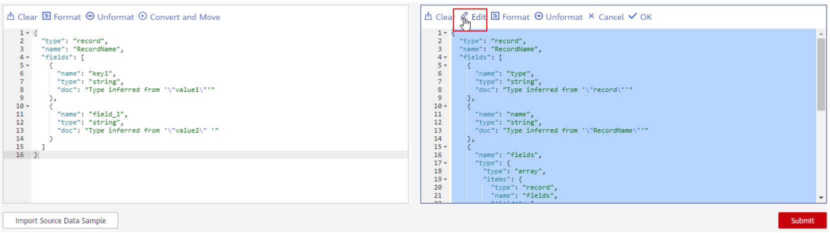
Step 3 In the left text box, click ⊙ to generate an Avro schema in the right text box according to the source data sample. Example:



```
{
  "type": "record",
  "name": "RecordName",
  "fields": [
    {
      "name": "key1",
      "type": "string",
      "doc": "Type inferred from '\"value1\"'"
    },
    {
      "name": "field_1",
      "type": "string",
      "doc": "Type inferred from '\"value2\" '"
    }
  ]
}
```

```
{"type":"record","name":"RecordName","fields":[{"name":"type","type":"string","doc":"Type
    inferred from '\"record\"'"},{"name":"name","type":"string","doc":"Type inferred from
    '\"RecordName\"'"},{"name":"fields","type":{"type":"array","items":{"type":"record","name"
    :"fields","fields":[{"name":"name","type":"string","doc":"Type inferred from '\"key1\"'"}
    ,{"name":"type","type":"string","doc":"Type inferred from '\"string\"'"},{"name":"doc"
    ,"type":"string","doc":"Type inferred from '\"Type inferred from '\\\"value1\\\"'\"'"}]}}
    ,"doc":"Type inferred from '[{\"name\":\"key1\",\"type\":\"string\",\"doc\":\"Type inferred
    from '\\\"value1\\\"'\"},{\"name\":\"field_1\",\"type\":\"string\",\"doc\":\"Type inferred
    from '\\\"value2\\\" '\"}]'"}]}
```

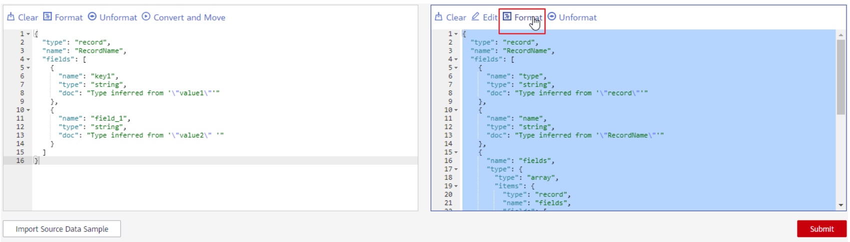
Step 4 In the right text box, click ✎ to modify the Avro schema. Example:



```
{
  "type": "record",
  "name": "RecordName",
  "fields": [
    {
      "name": "key1",
      "type": "string",
      "doc": "Type inferred from '\"value1\"'"
    },
    {
      "name": "field_1",
      "type": "string",
      "doc": "Type inferred from '\"value2\" '"
    }
  ]
}
```

```
{
  "type": "record",
  "name": "RecordName",
  "fields": [
    {
      "name": "type",
      "type": "string",
      "doc": "Type inferred from '\"record\"'"
    },
    {
      "name": "name",
      "type": "string",
      "doc": "Type inferred from '\"RecordName\"'"
    },
    {
      "name": "fields",
      "type": {
        "type": "array",
        "items": {
          "type": "record",
          "name": "fields",
```

Step 5 Click **Format** to format the parsed data. Example:



```
{
  "type": "record",
  "name": "RecordName",
  "fields": [
    {
      "name": "key1",
      "type": "string",
      "doc": "Type inferred from '\"value1\"'"
    },
    {
      "name": "field_1",
      "type": "string",
      "doc": "Type inferred from '\"value2\" '"
    }
  ]
}
```

```
{
  "type": "record",
  "name": "RecordName",
  "fields": [
    {
      "name": "type",
      "type": "string",
      "doc": "Type inferred from '\"record\"'"
    },
    {
      "name": "name",
      "type": "string",
      "doc": "Type inferred from '\"RecordName\"'"
    },
    {
      "name": "fields",
      "type": {
        "type": "array",
        "items": {
          "type": "record",
          "name": "fields",
```

Step 6 To delete the source data sample, click 🧹.

----End

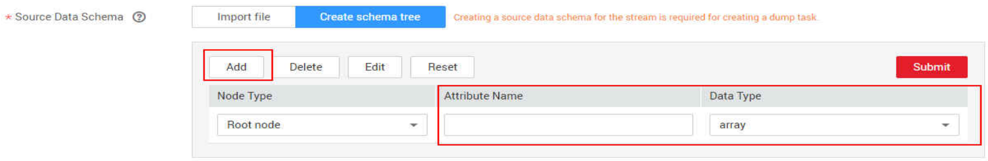
Creating a Schema for Source Data by Creating a Schema Tree

Use the following method to create a source data schema:

Step 1 When configuring **Source Data Schema**, click **Create Schema Tree**.

Step 2 After configuring an attribute name and data type, click **Add** to add a root node, as shown in **Figure 3-4**.

Figure 3-4 Adding the root node

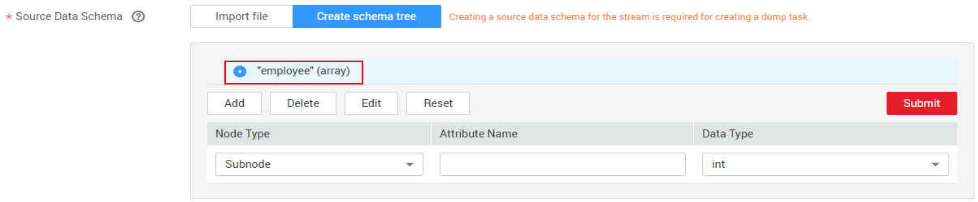


Creating a source data schema for the stream is required for creating a dump task.

Node Type	Attribute Name	Data Type
Root node		array

Step 3 Select the created root node and configure an attribute name and data type in the same way to add subnodes.

Figure 3-5 Creating a subnode



Creating a source data schema for the stream is required for creating a dump task.

"employee" (array)

Node Type	Attribute Name	Data Type
Subnode		int

NOTE

- To delete a node, select the check box of the node and click **Delete**.
- To edit the attributes of a node, select the check box of the node and click **Edit**.
- To delete all nodes, click **Reset**.

Step 4 Click **Submit**.

----End

Modifying a Source Data Schema

NOTE

Do not modify the source data schema of a stream if the stream has dump tasks.

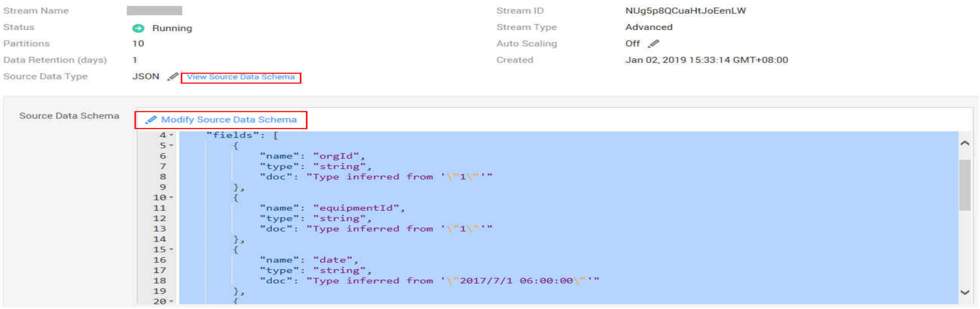
Step 1 Use the account to log in to the **DIS console**.

Step 2 Click in the upper left corner of the page and select a region.

Step 3 In the navigation tree on the left, choose **Stream Management**.

1. Click a stream name to access its details page.
2. Click **View Existing Source Data Schema** next to **Source Data Type**.
3. In the **Source Data Schema** text box, click to modify the source data schema.

Figure 3-6 Modifying the source data schema



Stream Name	[redacted]	Stream ID	NUg5p8QCuaHtJoEenLW
Status	Running	Stream Type	Advanced
Partitions	10	Auto Scaling	Off
Data Retention (days)	1	Created	Jan 02, 2019 15:33:14 GMT+08:00
Source Data Type	JSON View Source Data Schema		

Source Data Schema

```
    "fields": [
        {
            "name": "orgId",
            "type": "string",
            "doc": "Type inferred from '\"1\"'"
        },
        {
            "name": "equipmentId",
            "type": "string",
            "doc": "Type inferred from '\"1\"'"
        },
        {
            "name": "date",
            "type": "string",
            "doc": "Type inferred from '\"2017/7/1 06:00:00\"'"
        },
        {
```

NOTE

If a stream has dump tasks, modifying the source data schema of the stream will cause some data unable to be successfully dumped.

4. After the modification is complete, click **Submit**. Click **Cancel** to give up modifying the source data schema.

----End

3.5 Managing Stream Tags

A tag is an identifier of a stream. Adding tags to streams can help you identify and manage your stream resources.

You can add a maximum of 10 tags to a stream when creating the stream or add them on the details page of the created stream.

A tag consists of a tag key and a tag value. **Table 3-2** describes the rules for naming the tag key and value.

Table 3-2 Naming rules for a tag key and value

Parameter	Rule	Example
Key	A tag key cannot be left blank. A stream can have only one tag key. You can enter a maximum of 36 characters for **Key**. The value of **Key** cannot start or end with a space and cannot contain any of the following characters: is equal to (=), asterisk (*), is less than (<), is greater than (>), backslash (\), comma (,), delimiter (|), and slash (/).	Organization
Tag value	The value can be null. You can enter a maximum of 43 characters for **Key**. The value of **Key** cannot start or end with a space and cannot contain any of the following characters: is equal to (=), asterisk (*), is less than (<), is greater than (>), backslash (\), comma (,), delimiter (|), and slash (/).	Apache

Adding a Tag to a Stream

You can add a tag to a stream on the **Buy Stream** page.

1. Log in to the management console.
2. On the management console, choose **Service List** > **Analytics** > **Data Ingestion Service**.
3. On the DIS management console, click **Buy Stream**.
4. On the **Advanced Settings** tab page, select **Configure**.

   Enter the key and value of a tag to be added.

   You can add a maximum of 10 tags to the stream and use intersections of tags to search for the target stream.

NOTE

You can also add tags to existing streams. For details, see **Managing Tags**.

Searching for a Target Stream

You can search for a target stream by tag on the **Stream Management** page.

1. Log in to the management console.
2. Choose **EI Enterprise Intelligent** > **Data Ingestion Service**.
3. In the navigation tree, choose **Ingestion Management** > **Stream Management**. In the upper right corner of the page, click **Search by Tag**.
4. Enter the tag key and value of the stream you are searching for.

   You can select a tag key or tag value from its drop-down list. When the tag key or tag value is exactly matched, the system can automatically locate the target stream. If you enter multiple tags, their intersections are used to search for the stream.
5. Click **Search**.

   The system searches for the target stream by tag key or value.

Managing Tags

You can add, delete, modify, and view tags on the **Tags** tab page of a stream.

1. Log in to the management console.
2. Choose **EI Enterprise Intelligent** > **Data Ingestion Service**.
3. In the navigation tree, choose **Ingestion Management** > **Stream Management**. Click a stream to which the tags to be managed belong to.

   The stream details page is displayed.
4. Click the **Tags** tab and add, deleted, modify, and view tags.
   - View

     On the **Tags** tab page, you can view details about tags of the stream, including the number of tags and the key and value of each tag.
   - Add

     Click **Add Tag** in the upper left corner. In the displayed **Add Tag** dialog box, enter the key and value of the tag to be added, and click **OK**.
   - Modify

     In the **Operation** column of the tag, click **Edit**. In the displayed **Edit Tag** page, enter new tag key and value and click **OK**.
   - Delete

     In the **Operation** column of the tag, click **Delete**. After confirmation, click **OK** in the displayed page for deleting a tag.

3.6 Managing Applications

An application is a program identifier. Multiple applications can access data in the same stream. Checkpoints generated for each application are used to record the consumed data in the stream by each application.

You can create applications and view details about existing application on the DIS console.

Creating an Application

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click 📍 in the upper left corner of the page and select a region and project.

Step 3 In the navigation tree, select **App Management**.

On the **Applications** page, click **Create App** and enter an application name.

----End

Viewing an Application

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click 📍 in the upper left corner of the page and select a region and project.

Step 3 In the navigation tree on the left, choose **Stream Management**.

Step 4 Click the name of a stream that you want to view.

Step 5 Click **Applications** to view all applications that access the stream.

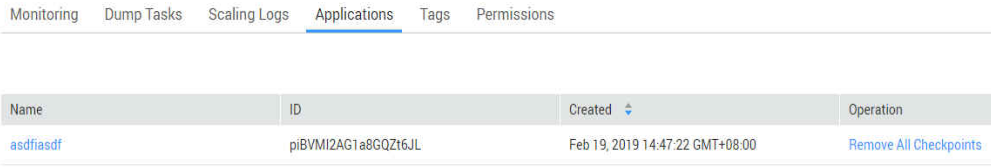
You can view the names, IDs, and creation time of the applications that access the stream.

After you click **Remove All Checkpoints**, all checkpoints of the application will be deleted.

NOTE

When an application consumes data, the latest SN of the consumed data is recorded as a checkpoint. When the data is reconsumed, the consumption can be continued based on this checkpoint.

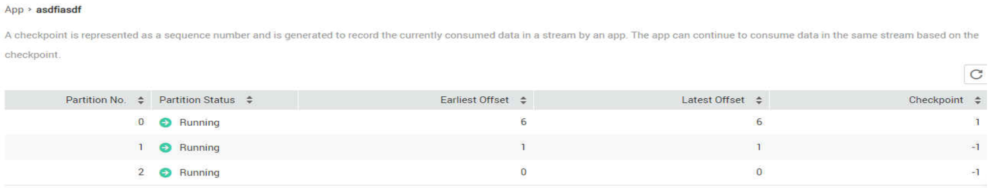
Figure 3-7 Viewing applications



Name	ID	Created	Operation
asdfiasdf	piBVMI2AG1a8GQZt6JL	Feb 19, 2019 14:47:22 GMT+08:00	Remove All Checkpoints

Step 6 Click a specific application to view its consumption details about the stream.

Figure 3-8 Viewing stream consumption details



App › **asdfiasdf**

A checkpoint is represented as a sequence number and is generated to record the currently consumed data in a stream by an app. The app can continue to consume data in the same stream based on the checkpoint.

Partition No.	Partition Status	Earliest Offset	Latest Offset	Checkpoint
0	Running	6	6	1
1	Running	1	1	-1
2	Running	0	0	-1

----End

3.7 Managing Authorization

Authorized users can upload data to or download data from DIS after an upload or download permission is granted.

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click 📍 in the upper left corner of the page and select a region and project.

Step 3 In the navigation tree on the left, choose **Stream Management**.

Step 4 Click a stream that you want to view.

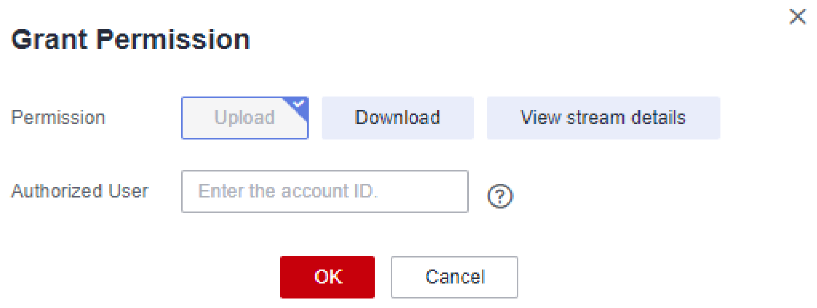
Step 5 On the **Permissions** tab page, click **Grant Permission**.

Select a permission, and then set the user information in the **Authorized User** text box.

NOTE

- To authorize all accounts, enter a wildcard (*).
- To authorize multiple accounts, enter these account names. These account names must be separated by commas (,).
- To authorize a subaccount of an account, enter the account name and then click **Query Subaccount**.

Figure 3-9 Granting permissions



Grant Permission

Permission: Upload ☑ | Download | View stream details

Authorized User: Enter the account ID. ?

----End

3.8 Debugging Streams

After creating a DIS stream, you can perform upload and download operations on the DIS console to verify the stream availability.

When the stream partition status is ACTIVE, both upload and download are supported.

If the partition status is DELETED, only download is supported.

Step 1 Use the account to log in to the **DIS console**.

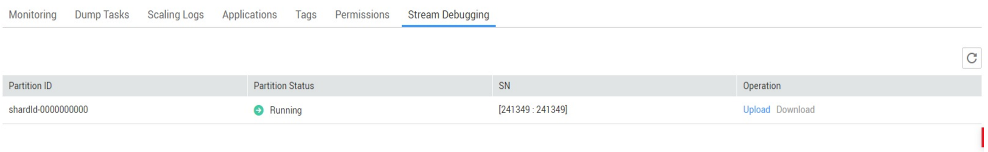
Step 2 Click in the upper left corner of the page and select a region and project.

Step 3 In the navigation tree on the left, choose **Stream Management**.

Step 4 Click a stream that you want to view.

Step 5 Click **Stream Debugging**, and select **Upload** or **Download** in the **Operation** column of the target partition to upload or download data.

Figure 3-10 Stream debugging page

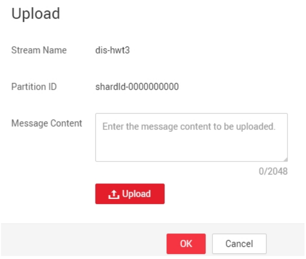


Partition ID	Partition Status	SN	Operation
shardId-0000000000	Running	[241349 : 241349]	Upload Download

Step 6 Click **Upload**, enter the content to be uploaded in the **Upload** text box, and click **OK**.

The system displays a message indicating that the data is uploaded successfully and displays the sequence number of the data upload success record.

Figure 3-11 Uploading data


Upload

Stream Name dis-hwt3

Partition ID shardId-0000000000

Message Content Enter the message content to be uploaded.

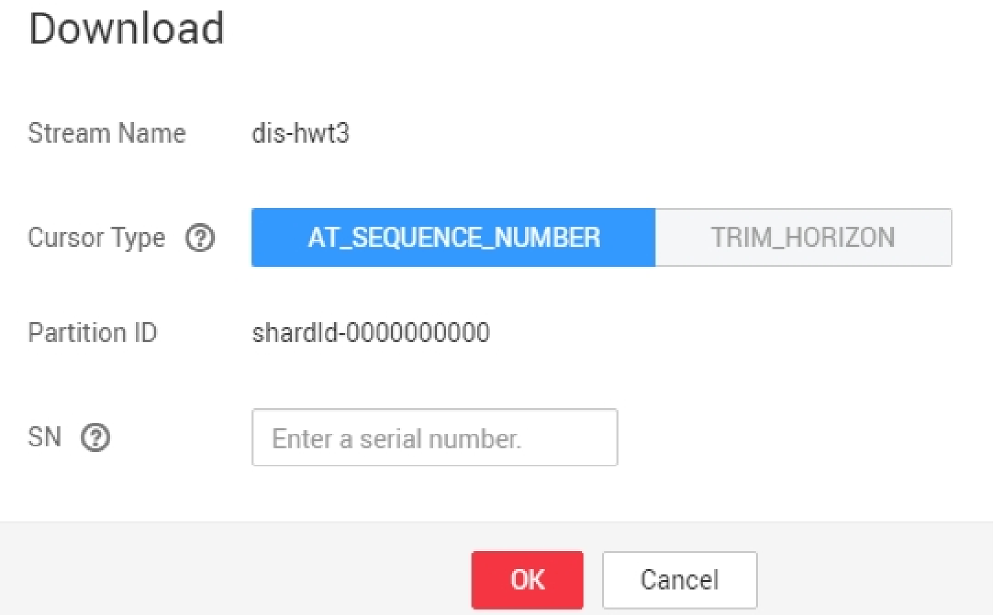
0/2048

Step 7 Click **Download**, select a cursor type, enter a sequence number, and click **OK**. After the download is successful, you can obtain the response body in text format.

NOTE

The entered sequence number must be within the valid range of the partition data. The valid range of the partition data can be obtained by calling the **describeStream** API. **sequenceNumberRange** indicates the data validity range. The first value is the sequence number of the earliest data, the last value is the sequence number of the next uploaded data, and the sequence number of the latest data is one less than the last value.

Figure 3-12 Downloading data



Download

Stream Name dis-hwt3

Cursor Type AT_SEQUENCE_NUMBER TRIM_HORIZON

Partition ID shardId-0000000000

SN Enter a serial number.

----End

3.9 Scaling Up/Down a Stream

After a stream is created successfully, you can scale up or down a stream by adding or reducing partitions to meet capacity change requirements. Streams can be scaled automatically or manually.

Constraints

- A maximum of 10 automatic scale-up, 10 manual scale-up, and 1 scale-down operations can be performed for each stream within one hour.
- After scaling up or down a stream, pay attention to the following:
  - When uploading data, you are not advised to set PartitionKey. DIS automatically uploads data to multiple partitions based on the number of partitions in the stream. If you set PartitionKey, data skew and throttling may occur in the stream.
  - When downloading data, you need to periodically use the descriptStream API to check the partition quantity change so that DIS can download data from all partitions.

Auto Scaling

Auto Scaling Mechanism

- If throttling is triggered in the last minute (that is, the maximum throughput of the partitions in the stream has been reached) and the upload traffic of the stream is greater than 80% of the stream bandwidth, an automatic scale-up is triggered. The number of target partitions is equal to the number of current partitions divided by 0.6 and rounded up.

  For example, a stream has five common partitions, and the upload bandwidth of the stream is 5 Mbit/s. If throttling is triggered in the last minute and the upload traffic of the stream exceeds 4 Mbit/s, an automatic scale-up is triggered to add four more partitions. The number of target partitions is 8.3 (5/0.6), which is rounded up to 9.

- If the upload traffic and download traffic of a stream are both less than 30% of the stream bandwidth, an auto scale-down is triggered. The number of target partitions is equal to the number of current partitions divided by 2 and rounded down.

  For example, a stream has five common partitions, and its upload and download bandwidth is 5 Mbit/s and 10 Mbit/s, respectively. If the upload traffic of the stream is less than 1.5 Mbit/s and the download traffic is less than 3 Mbit/s, an automatic scale-down is triggered to remove three partitions. The number of target partitions is 2.5 (5/2), which is rounded down to 2.

Auto Scaling Rules

- The interval between performing automatic scale-up and scale-down must be greater than 1 minute. Automatic scale-down cannot be triggered within 2 minutes after automatic or manual scale-up is performed, and meanwhile automatic scale-up cannot be triggered within 2 minutes after automatic or manual scale-down is performed.
- When scaling up a stream, restore the **DELETED** and **EXPIRED** partitions to the **ACTIVE** partitions to make the partitions accessible. If the stream capacity is still insufficient, the system will create new partitions.
- The partitions that have been scaled down will not be charged or occupy the quota. Data on those partitions is readable but is not writable during **Data Retention** configured in **Step 1: Creating a DIS Stream**. Once **Data Retention** is expired, the data will become inaccessible.

Enabling Auto Scaling

Step 1 Use the account to log in to the **DIS console**.

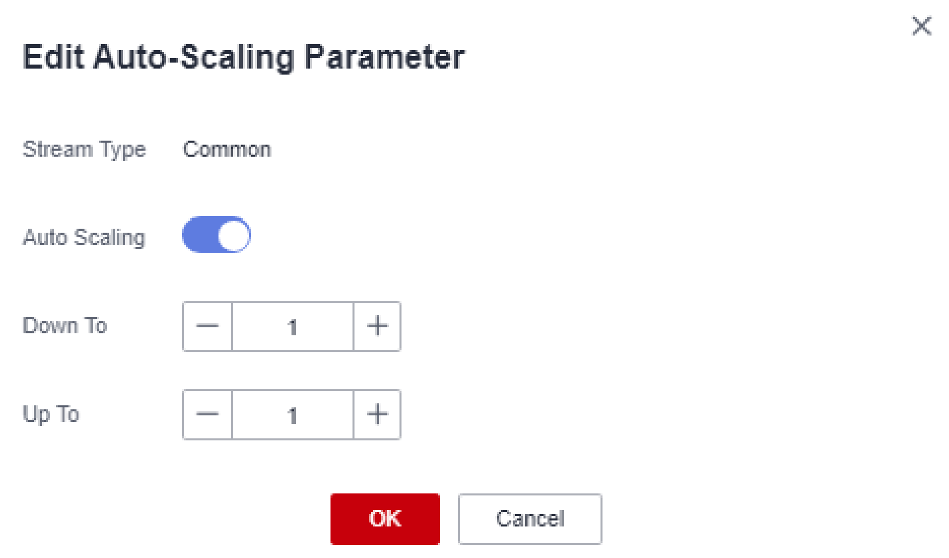
Step 2 Click 📍 in the upper left corner of the page and select a region and project.

Step 3 Perform the following steps to automatically scale up or down a stream:

In the navigation tree on the left, choose **Stream Management**.

1. On the **Stream Management** page, click the name of the stream to be scaled.
2. Click ✎ next to **Auto Scaling**.
3. In the displayed **Edit Auto-Scaling Parameter** dialog box, enable **Auto-Scaling**.

Figure 3-13 Edit Auto-Scaling Parameter



Edit Auto-Scaling Parameter

Stream Type: Common

Auto Scaling: on

Down To: 1

Up To: 1

4. Set the upper and lower thresholds for automatic scaling and click **OK**.

----End

Manual Scaling

Manual Scaling Rules

- When scaling up a stream, ensure that the number of target partitions is greater than the number of the current partitions but is not greater than the total number of the remaining quotas and current partitions.
- When scaling up a stream, restore the **DELETED** and **EXPIRED** partitions to the **ACTIVE** partitions to make the partitions accessible. If the stream capacity is still insufficient, the system will create new partitions.
- When scaling down a stream, ensure that the number of target partitions is smaller than the number of current partitions but is not smaller than **1**.
- The partitions that have been scaled down will not be charged or occupy the quota. Data on those partitions is readable but is not writable during **Data Retention** configured in **Step 1: Creating a DIS Stream**. Once **Data Retention** is expired, the data will become inaccessible.

Manually Scaling Up/Down a Stream

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click 📍 in the upper left corner of the page and select a region and project.

Step 3 Perform either of the following operations to manually scale up or down a stream:

- In the navigation tree on the left, choose **Stream Management**.
  a. On the **Stream Management** page, click the name of the stream to be scaled.
  b. In the upper right corner of the displayed page, click **Scale Up/Down**. The **Scale Up/Down Stream** dialog box is displayed.

   c. Change the number of target partitions and click **Yes**.

- In the navigation tree on the left, choose **Stream Management**.

   a. In the **Operation** column of the stream to be scaled, click **More** and choose **Scale Up/Down** from the drop-down list.

   b. The **Scale Up/Down Partition** dialog box is displayed.

   c. Change the number of target partitions and click **Yes**.

----End

Viewing Scale-Up/Down Logs

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click 📍 in the upper left corner of the page and select a region and project.

Step 3 In the navigation tree on the left, choose **Stream Management**.

Step 4 Click the name of a stream to access its details page.

Step 5 On the displayed page, click the **Scale-Up/Down Logs** tab. View the scaling details of the stream.

----End

3.10 Deleting a Stream

NOTE

A deleted stream will not be charged and cannot be restored. Exercise caution when performing this operation.

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click 📍 in the upper left corner of the page and select a region and project.

Step 3 In the navigation tree on the left, choose **Stream Management**.

Step 4 Click **Delete** in the **Operation** column of the stream that you want to delete. The **Delete Stream** dialog box is displayed.

Step 5 Click **OK** to delete the selected stream.

----End

4 Using DIS

4.1 Checking and Configuring DNS Information

By default, an Elastic Cloud Server (ECS) is configured with two external domain name system (DNS) servers.

```
# Generated by NetworkManager
search openstacklocal
nameserver 114.114.114.114
nameserver 114.114.115.115
```

If the ECS you will use does not have an elastic IP address (EIP) or you do not want to use an EIP to transmit the traffic load generated during the use of AEI_Register.sh and fisclient programs, add a DNS server in configuration file **/etc/resolv.conf**.

XXX.XXX.XXX.XXX is the IP address of the DNS server.

```
# Generated by NetworkManager
search openstacklocal
nameserver XXX.XXX.XXX.XXX
nameserver 114.114.114.114
nameserver 114.114.115.115
```

NOTE

- The new DNS server address must be placed above all the existing DNS server addresses.
- The DNS configuration takes effect immediately after modifications to the **/etc/resolv.conf** file are saved.

Modifications to the **/etc/resolv.conf** file become invalid after the ECS restarts. In this case, you need to reconfigure the file. If you do not want to reconfigure the file each time after restarting the ECS, perform the following steps to modify the subnet information of the Virtual Private Cloud (VPC) to which the ECS belongs and add DNS server addresses to the subnet to which the ECS belongs.

Procedure

Step 1 Log in to the ECS console.

Step 2 In the navigation tree of the ECS console, choose **Elastic Cloud Server**.

Step 3 On the ECS details page, click the **NICs** tab. Expand details of a chosen NIC and view the name of the subnet to which the ECS belongs.

Step 4 On the ECS details page, click the VPC name or ID.

The Network Console is then launched.

Step 5 In the right pane of the Network Console, click the VPC name or ID.

A page with details of the VPC is then displayed.

Step 6 Choose **Subnets** from the left navigation bar and click the subnet name or ID in the subnet list to access the VPC subnet page.

Step 7 On the subnet details page, click **Modify** next to **DNS Server Address** in the **Gateway and DNS Information** area.

Step 8 Enter the IP address of the DNS server in **Edit DNS Server Address** and click **OK**.

Step 9 Restart the ECS and check that the **/etc/resolv.conf** file contains the new DNS server address which is placed in the front of other DNS server addresses.

```
# Generated by NetworkManager
search openstacklocal
nameserver XXX.XXX.XXX.XXX
nameserver 114.114.115.115
```

NOTE

Modifying the subnet information of a VPC affects all ECSs in this subnet.

----End

4.2 Uploading Data by Using Agent

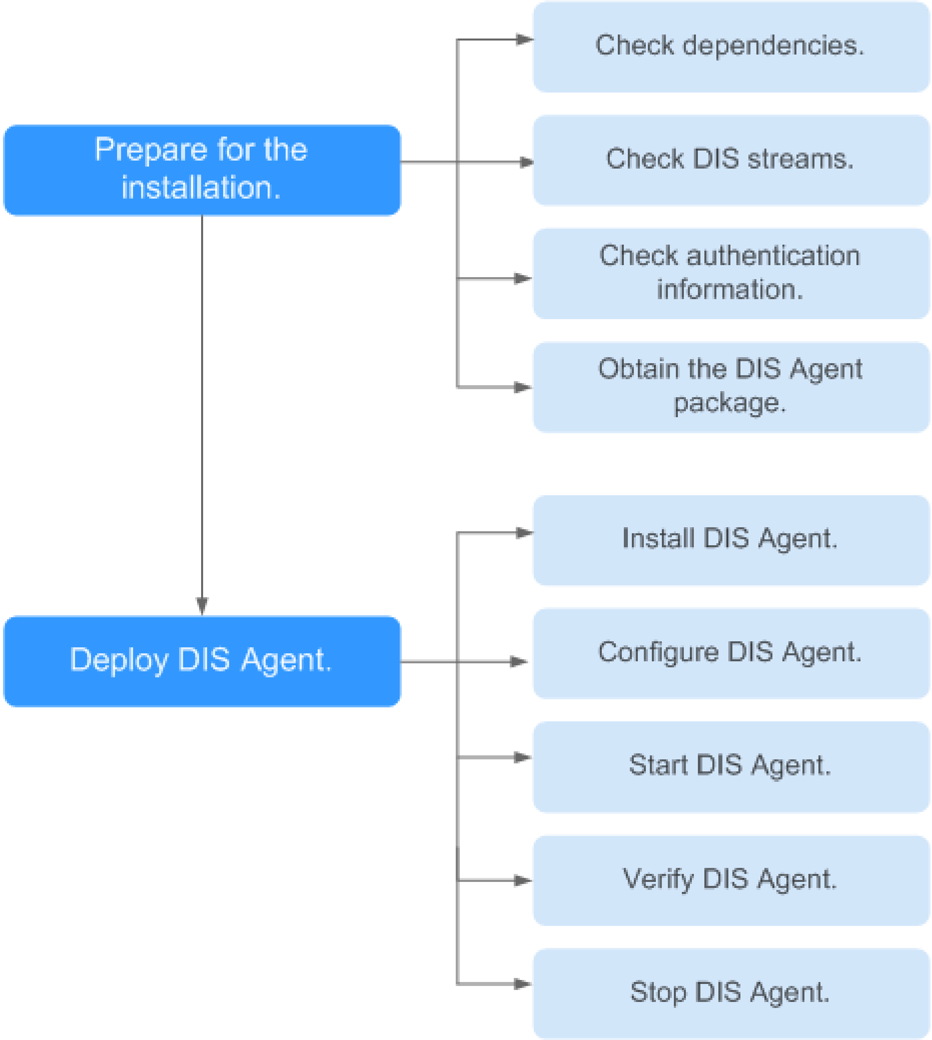
4.2.1 DIS Agent Overview

DIS Agent is a client-side program provided by the Data Ingestion Service (DIS) to fulfill the following tasks:

Monitor text files continuously, collect incremental data in real time, parse the data by delimiter and upload it to DIS streams. The source data type of a stream can be BLOB, JSON, and CSV.

Figure 4-1 depicts the process for installing the DIS agent.

Figure 4-1 Installation flowchart



4.2.2 Preparing for Installing DIS Agent

Checking Dependencies

Step 1 Check the server type.

- Linux x86-64 (64-bit) server, for example, EulerOS, Ubuntu, Debian, CentOS, or OpenSUSE
- Windows 7 or a later version

Step 2 Ensure that Java 1.8.0 or later has been installed.

To download JRE, go to **https://www.java.com/en/download/manual.jsp**.

If Java 1.8.0 or later is not installed, perform the following steps to install it on a Linux server:

1. As the root user, run the following command to navigate to the **/opt** directory:

   cd /opt

2. Run the following command to create the JDK installation directory **jre**:

   mkdir -p jre

3. Run the following command to assign the permission to the JDK installation directory:

   chmod -R 640 jre/

4. Download the Java runtime environment (JRE) installation package from the following website:

   tar -zxvf 'JRE installation package name'.tar.gz

5. Modify the **/etc/profile** file.

   a. Run the following command to open the **/etc/profile** file: **vim /etc/profile**

   b. Add the following JDK installation directory information to the configuration option **JAVA_HOME** in the **/etc/profile** file:

      export JAVA_HOME=path to the **jre** folder

      export PATH=$PATH:$JAVA_HOME/bin

      export CLASSPATH=.:$JAVA_HOME/lib/rt.jar:$JAVA_HOME/lib/ext

   c. Run the following command to save the modification and exit:

      :wq

6. Run the following command to validate the JDK configuration:

   source /etc/profile

----End

Checking DIS Streams

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click 📍 in the upper left corner of the page and select a region and project.

Step 3 In the navigation tree, choose **Stream Management**.

Step 4 Ensure that at least one DIS stream is in the **Running** state and ready to receive incoming data.

----End

Checking Authentication Information

- AK/SK file

  Access Key ID/Secret Access Key (AK/SK) files are created by the Identity and Access Management (IAM) service to authenticate calls to application programming interfaces (APIs) on the cloud. To obtain AK/SK, choose **My Credentials** > **Access Keys**.

- Project ID

  A project is a group of tenant resources. A tenant can have multiple projects, one for each region. Each region has a unique Project ID. To view the project IDs of different regions, choose .

Obtaining DIS Agent Package

Obtain the **dis-agent-*X.X.X*.zip** package from **https://dis-publish.obs-website.cn-north-1.myhuaweicloud.com/**.

4.2.3 Installing DIS Agent

Prerequisites

PuTTY has been installed.

Installing DIS Agent on a Linux Server

Step 1 Start PuTTY and log in to the server on which logs reside, that is, the server that experiences a dependency check in section **Checking Dependencies**.

Step 2 Upload the DIS Agent package **dis-agent-*X.X.X*.zip** obtained in section **Obtaining DIS Agent Package** to the **/opt** directory.

Step 3 Run the following command to decompress the **dis-agent*-X.X.X*** directory from the DIS Agent package **dis-agent*-X.X.X*.zip**:

unzip dis-agent-X.X.X.zip

Step 4 Run the following command to navigate to the **dis-agent*-X.X.X*** directory:

cd dis-agent-X.X.X

----End

Installing DIS Agent on a Windows Server

Step 1 Save the **dis-agent-*X.X.X*.zip** package obtained in **Obtaining DIS Agent Package** to the local host.

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

----End

4.2.4 Configuring DIS Agent

The DIS Agent configuration file is in the YAML format. Configuration parameters and values must be separated by colon (:) and space.

You can obtain the **agent.yml** file template from the **dis-agent** package. The following is an example. **Table 4-1** describes the configuration parameters in the file.

```
---
# cloud region id
region: myregion
## The plaintext storage of the AK and SK used for authentication has great security risks
## You are advised to use the /bin/dis-encrypt.sh script to encrypt the AK and SK before storing them
# you ak (get from 'My Credential')
ak: YOU_AK
# you sk (get from 'My Credential')
sk: YOU_SK
# you_key(encry you ak or sk)
encrypt.key: abc
```

```
# you project id (get from 'My Credential')
projectId: YOU_PROJECTID
# the dis endpoint
endpoint: https://dis.myregion.cloud.com
# config each flow to monitor file.
flows:
  ### DIS Stream
  - DISStream: YOU_DIS_STREAM_1
    ## only support specified directory, filename can use * to match some files. eg. * means match all file, test*.log means match test1.log or test-12.log and so on.
    filePattern: /tmp/*.log
    ## from where to start: 'START_OF_FILE' or 'END_OF_FILE'
    initialPosition: START_OF_FILE
    ## upload max interval(ms)
    maxBufferAgeMillis: 5000

  ### If there are other monitor files, continue to follow the above configuration
  ### another dis stream monitor config, uncomment # if you want to use this feature
  #- DISStream: YOU_DIS_STREAM_2
  #  filePattern: /opt/*.log
  #  initialPosition: START_OF_FILE
  #  maxBufferAgeMillis: 5000

  ### OBS Stream: Upload the matching file to OBS and send the file name to DIS, uncomment # if you want to use this feature
  #- OBSStream: YOU_DIS_STREAM_3
  #  filePattern: /opt/*.log
  #  initialPosition: START_OF_FILE
  #  ## bucket name
  #  OBSBucket: YOU_OBS_BUCKET_NAME
  #  ## OBS endpoint
  #  OBSEndpoint: https://obs.myregion.cloud.com
  #  ## the directory(using / separated) where the files are stored under the bucket, automatically created if it does not exist
  #  dumpDirectory: example/dis/
```

NOTE

After the configuration is complete, delete unnecessary examples from flows in **agent.yml** or use # to comment out them. For example, if only one DISStream is configured, delete or comment out the following CustomFileStream and other DISStream modules.

Configuring DIS Agent on a Linux Server

Step 1 Start PuTTY and log in to the Linux server on which the DIS Agent is installed.

Step 2 Run the **cd /opt/dis-agent-X.X.X/** command to open the **dis-agent-X.X.X** directory.

Step 3 Run the **vim conf/agent.yml** command to open the DIS Agent configuration file **agent.yml**. Modify parameter values in the file to meet specific requirements. **Table 4-1** describes the configuration parameters in the file.

Table 4-1 Parameters in the **agent.yml** file

Parameter	Mandatory	Description	Default Value
region	Yes	Region where DIS is deployed. **NOTE** For details about how to obtain the region where DIS is deployed, see **Regions and Endpoints**.	-
AK	Yes	User's AK. **NOTE** You can encrypt the AK or use a plaintext AK. For details about how to encrypt the AK, see the AK/SK encryption note. For details about how to obtain an AK, see **Checking Authentication Information**.	-
SK	Yes	User's SK. **NOTE** You can encrypt the SK or use a plaintext SK. For details about how to encrypt the SK, see the AK/SK encryption note. For details about how to obtain an SK, see **Checking Authentication Information**.	-
encrypt.key	No	Key used for encryption **NOTE** If you want to use an encrypted AK or SK, you must configure this parameter in the **agent.yml** file. Ensure that you use the configured key to encrypt the AK/SK. Otherwise, the AK/SK cannot be decrypted.	-
projectId	Yes	Project ID specific to your region. For details about how to obtain a project ID, see **Checking Authentication Information**.	-

Parameter	Mandatory	Description	Default Value
endpoint	Yes	DIS gateway address. Format: https://DIS endpoint **NOTE** For details about how to obtain the DIS endpoint, see **Regions and Endpoints**.	-
body.serialize.type	No	Format of the DIS data package to be uploaded (non-original data format). ● **json**: The DIS data packet is encapsulated in the format of JSON. ● **protobuf**: The DIS data packet is encapsulated in the 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

Parameter	Mandatory	Description	Default Value
PROXY_PROTOCOL	No	Proxy protocol. http and https are supported.	http
PROXY_USERNAME	No	Proxy username.	-
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 4-2** describes configuration parameters.

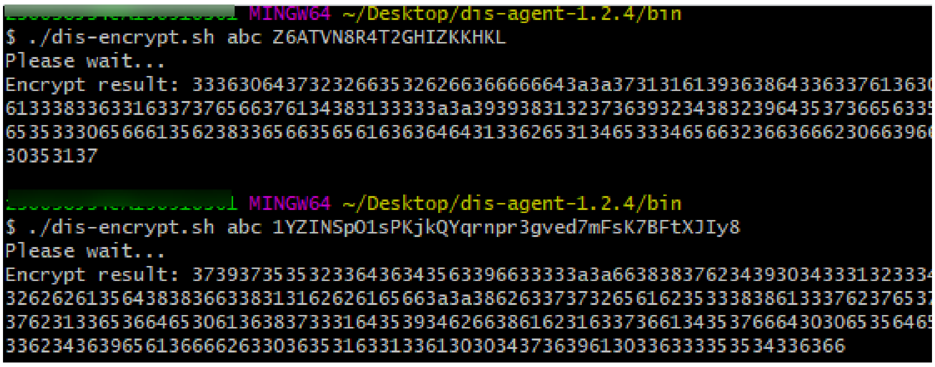
The **agent.yml** file provides example parameter settings.

To encrypt the AK/SK, perform the following steps:

Download and install the DIS Agent by referring to **Installing DIS Agent** and use the script in the **bin** directory of the DIS Agent package to encrypt the AK/SK. The detailed operations are as follows (Windows):

1. Go to the **bin** directory of DIS Agent and right-click git bash here to run the script, for example, **./dis-encrypt.sh {key} {ak}** to obtained the encrypted AK. Then configure the encrypted AK in the **agent.yml** file.
2. Encrypt the SK in the same way. Then configure the encrypted AK/SK and key in the **agent.yml** file.

Figure 4-2 Encryption example



```
[redacted] MINGW64 ~/Desktop/dis-agent-1.2.4/bin
$ ./dis-encrypt.sh abc Z6ATVN8R4T2GHIZKKHKL
Please wait...
Encrypt result: 3336306437323266353262663666666643a3a373131613936386433633761363
6133383363316333737656637613438313333 3a3a393938313237363932343832396435373665633 5
653533306566613562383365663565616363646431336265313465333465663236636662306639 6
30353137

[redacted] MINGW64 ~/Desktop/dis-agent-1.2.4/bin
$ ./dis-encrypt.sh abc 1YZINSpO1sPKjkQYqrnpr3gved7mFsK7BFtXJIy8
Please wait...
Encrypt result: 37393735353233364363435633966333333a3a6638383762343930343331323333 4
32626261356438383663383131626261656 63a3a3862633373732656162353338386133376237653 7
376231336533664653061363837333164353934626663861623163337366134353766643030653564 6
33623436396561366662633036353163313361303034373639613033363333353534336366
```

Table 4-2 DISStream configuration parameters

Parameter	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**.	-
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

Parameter	Mandatory	Description	Default Value
maxBufferAgeMillis	No	The maximum number of milliseconds that must elapse before data can be uploaded to the DIS. Unit: ms ● If the buffer is full with data waiting to be uploaded, data will be immediately uploaded to the 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
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"

Parameter	Mandatory	Description	Default Value
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
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: ms **NOTE** This parameter is available only when **isFileAppendable** is set to **false**.	5000

Parameter	Mandatory	Description	Default Value
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
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	The number of sender threads. By default, there is only one sender thread. **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 each time when 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

----End

Configuring DIS Agent on a Windows Server

Step 1 Use a file manager to open the directory (for example, **C:\dis-agent-X.X.X**) where the installation package is decompressed.

Step 2 Open the **agent.yml** file using an editor and modify parameter values in the file to meet specific requirements.

NOTE

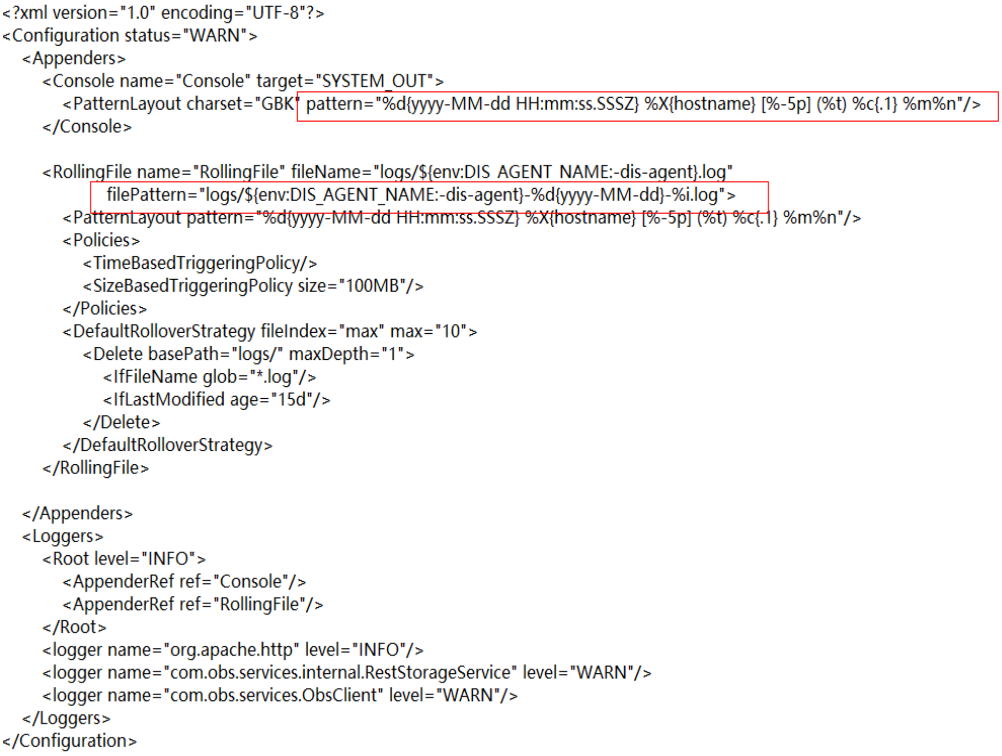
The **agent.yml** file is in the Linux format. You are advised to use the general-purpose text editor to edit the file.

About log files:

In the installation path of DIS Agent, the **logs** directory stores the log files generated during DIS Agent running. The **dis-agent.log** file records the running status of DIS Agent, and the log files with dates, such as **dis-agent-2022-10-28.log**, record file upload records. One log file is generated every day.

You can also customize the storage path of log files in the **log4j2.xml** file in the **conf** folder in the DIS Agent installation path.

Figure 4-3 log4j2



```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN">
  <Appenders>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout charset="GBK" pattern="%d{yyyy-MM-dd HH:mm:ss.SSSZ} %X{hostname} [%-5p] (%t) %c{.1} %m%n"/>
    </Console>

    <RollingFile name="RollingFile" fileName="logs/${env:DIS_AGENT_NAME:-dis-agent}.log"
          filePattern="logs/${env:DIS_AGENT_NAME:-dis-agent}-%d{yyyy-MM-dd}-%i.log">
      <PatternLayout pattern="%d{yyyy-MM-dd HH:mm:ss.SSSZ} %X{hostname} [%-5p] (%t) %c{.1} %m%n"/>
      <Policies>
        <TimeBasedTriggeringPolicy/>
        <SizeBasedTriggeringPolicy size="100MB"/>
      </Policies>
      <DefaultRolloverStrategy fileIndex="max" max="10">
        <Delete basePath="logs/" maxDepth="1">
          <IfFileName glob="*.log"/>
          <IfLastModified age="15d"/>
        </Delete>
      </DefaultRolloverStrategy>
    </RollingFile>

  </Appenders>
  <Loggers>
    <Root level="INFO">
      <AppenderRef ref="Console"/>
      <AppenderRef ref="RollingFile"/>
    </Root>
    <logger name="org.apache.http" level="INFO"/>
    <logger name="com.obs.services.internal.RestStorageService" level="WARN"/>
    <logger name="com.obs.services.ObsClient" level="WARN"/>
  </Loggers>
</Configuration>
```

----End

4.2.5 Starting DIS Agent

Starting DIS Agent on a Linux Server

Step 1 Start PuTTY and log in to the server on which logs reside.

Step 2 Run the following command to navigate to the DIS Agent installation directory:

cd /opt/dis-agent-x.x.x/*x.x.x* indicates the version number.

Step 3 Run the following script to start the DIS Agent:

bash bin/start-dis-agent.sh

If multiple DIS Agent processes need to be started, you need to use **-c** to specify the configuration file and **-n** to specify the name of the new Agent process.

bash bin/start-dis-agent.sh -c config/anotherAgent.yml -n anotherAgent

NOTICE

Run the bash script to start the DIS Agent. Running another script, such as sh or ./ may fail to start the DIS Agent.

If information similar to the following appears, the DIS Agent has started successfully:

```
Success to start DIS Agent [xxxxx].
```

If no Java variables can be found after the DIS Agent starts, run the following command to restart the DIS Agent:

source /etc/profile

----End

Starting DIS Agent on a Windows Server

Step 1 Navigate to the bin directory of DIS Agent, for example, **C:\dis-agent-X.X.X\bin**.

Step 2 Double-click **start-dis-agent.bat** to start the DIS Agent.

If information similar to the following appears in log files, the DIS Agent has started successfully:

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

----End

4.2.6 Testing DIS Agent

Testing DIS Agent on a Linux Server

Step 1 Start PuTTY and log in to the server on which logs reside.

Step 2 Run the following command to navigate to the directory in which the DIS Agent stores logs:

```
cd /opt/dis-agent-X.X.X/logs
```

Step 3 Run the following command to view logs:

```
tail -100f dis-agent.log
```

- If information similar to the following appears, the DIS Agent is running properly.

  ```
  Agent: Startup completed in xx ms
  ```

- If any of the following information appears, the DIS Agent is not running properly and you need to troubleshoot the DIS Agent:
  - **HttpClientErrorException: 400 Bad Request**

    Possible cause: The value of **DISStream** or **projectId** in **Configuring DIS Agent** is incorrect.

    Solution: Stop the DIS Agent process and correct the parameter values.
  - **HttpClientErrorException: 403 Forbidden**

    Possible cause: The DIS gateway blacklists the IP address of the Linux server on which the DIS Agent is installed and consequently the request to view logs is blocked. Usually, the Linux server IP address is blacklisted because a DIS API is repeatedly called using incorrect configuration.

    Solution: Stop the DIS Agent process, modify the **agent.yml** file (see **Configuring DIS Agent**). Restart the DIS Agent process 30 minutes after it stops.
  - **UnknownHttpStatusCodeException: Unknown status code [441]**

    Possible cause: AK/SK configuration is incorrect.

    Solution: Stop the DIS Agent process and correct the AK/SK configuration.
  - **ConnectTimeoutException: Connect to DOMAIN[DOMAIN/IP] failed: connect timed out**

    Possible cause: The connection between the Linux server on which the DIS Agent is installed and the DIS gateway times out.

    Solution: Ensure that the Linux server on which the DIS Agent is installed can connect to the Internet.

Step 4 Check whether the DIS Agent can upload logs.

- If the monitoring directory configured in **agent.yml** contains matched files, logs will record the information similar to the following, indicating that [N1 records (B1 bytes)/N2 files (B2 bytes)] are parsed and [N3 records/N4 files] are successfully uploaded.

  ```
  Agent: Progress: [N1 records (B1 bytes) / N2 files (B2 bytes)] parsed, and [N3 records / N4 files] sent
  successfully to destinations. Uptime: 30146ms
  ```

- If no files match the **filePattern**, run the following command to generate log files required for testing log uploading:

  ```
  echo "`date` Hello world." >> /tmp/test.log
  ```

Step 5 Log in to the DIS console and view monitoring data of the DIS stream specified by the **DISStream** or **CustomFileStream** parameter in **Configuring DIS Agent**. If the stream monitoring data shows that data has been uploaded to the DIS stream, the DIS Agent has been successfully installed.

----End

Testing DIS Agent on a Windows Server

Step 1 Use a file manager to navigate to the **logs** directory.

Step 2 Open the **dis-agent.log** file using an editor. View logs in the file.

- If information similar to the following appears, the DIS Agent is running properly.

  ```
  Agent: Startup completed in xx ms
  ```

- If any of the following information appears, the DIS Agent is not running properly and you need to troubleshoot the DIS Agent:
  - **HttpClientErrorException: 400 Bad Request**

    Possible cause: The value of **DISStream** or **projectId** in **Configuring DIS Agent** is incorrect.

    Solution: Stop the DIS Agent process and correct the parameter values.
  - **HttpClientErrorException: 403 Forbidden**

    Possible cause: The DIS gateway blacklists the IP address of the Linux server on which the DIS Agent is installed and consequently the request to view logs is blocked. Usually, the Linux server IP address is blacklisted because a DIS API is repeatedly called using incorrect configuration.

    Solution: Stop the DIS Agent process, modify the **agent.yml** file (see **Configuring DIS Agent**). Restart the DIS Agent process 30 minutes after it stops.
  - **UnknownHttpStatusCodeException: Unknown status code [441]**

    Possible cause: AK/SK configuration is incorrect.

    Solution: Stop the DIS Agent process and correct the AK/SK configuration.
  - **ConnectTimeoutException: Connect to DOMAIN[DOMAIN/IP] failed: connect timed out**

    Possible cause: The connection between the Linux server on which the DIS Agent is installed and the DIS gateway times out.

    Solution: Ensure that the Linux server on which the DIS Agent is installed can connect to the Internet.

Step 3 Check whether the DIS Agent can upload logs.

- If the monitoring directory configured in **agent.yml** contains matched files, logs will record the information similar to the following, indicating that [N1 records (B1 bytes)/N2 files (B2 bytes)] are parsed and [N3 records/N4 files] are successfully uploaded.

  ```
  Agent: Progress: [N1 records (B1 bytes) / N2 files (B2 bytes)] parsed, and [N3 records / N4 files] sent
  successfully to destinations. Uptime: 30146ms
  ```

- If no files match the **filePattern**, run the following command to generate log files required for testing log uploading:

  echo %date%time%Hello world. >> C:\test.log

Step 4 Log in to the DIS console and view monitoring data of the DIS stream specified by the **DISStream** or **CustomFileStream** parameter in **Configuring DIS Agent**. If the stream monitoring data shows that data has been uploaded to the DIS stream, the DIS Agent has been successfully installed.

----End

4.2.7 Stopping DIS Agent

Stopping DIS Agent on a Linux Server

Step 1 Start PuTTY and log in to the server on which logs reside.

Step 2 Run the following command to navigate to the DIS Agent installation directory:

cd /opt/dis-agent-X.X.X/

Step 3 Run the following script to stop the DIS Agent:

bash bin/stop-dis-agent.sh

NOTICE

Run the bash script to stop the DIS Agent. Running another script, such as sh or ./ may fail to stop the DIS Agent.

If information similar to the following appears, the DIS Agent is stopping. *xxxxx* indicates the process ID.

```
Stopping Agent [xxxxx].....
```

If information similar to the following appears, the DIS Agent has stopped:

```
Stopping Agent [xxxxx]............ Successfully.
```

To forcibly stop the DIS Agent process, perform the following steps:

1. Run the command to obtain the DIS Agent process ID (PID):

   ps -ef | grep dis-agent | grep -v grep

   The second field in the command output indicates the PID.

2. Run the following command to forcibly stop the DIS Agent process:

   kill -9 PID

----End

Stopping DIS Agent on a Windows Server

Step 1 Press **Ctrl+C** on the CMD terminal. The following information appears:

```
[INFO ] (Agent STOPPING) com.bigdata.dis.agent.Agent Agent: Shutting down...
```

Step 2 When the following information appears, it indicates that the DIS Agent has stopped. Enter **Y** and then press **Enter** to exit.

```
Terminate batch job (Y/N)?
```

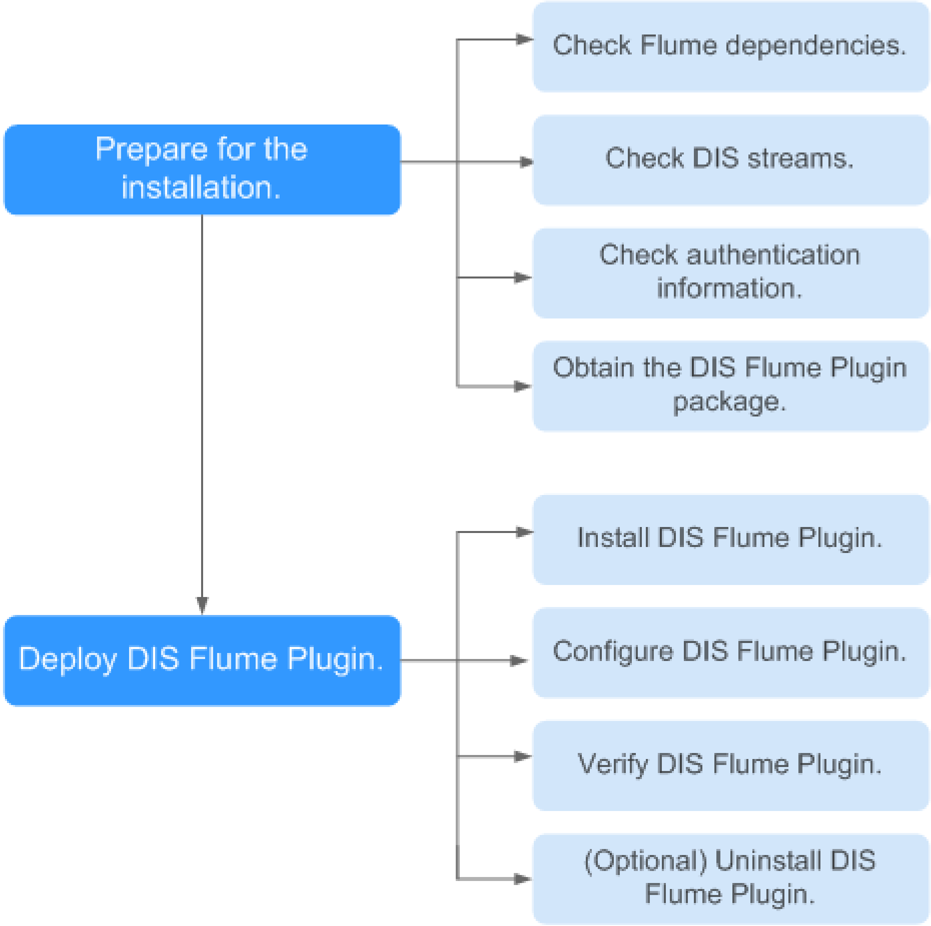
----End

4.3 Using DIS Flume Plugin to Upload and Download Data

4.3.1 Overview

A DIS Flume Plugin is a Flume plugin provided by DIS and consists of DIS Source and DIS Sink. DIS Source is used to download data from DIS to Flume Channel. DIS Sink is used to upload data from Flume Channel to DIS. **Figure 4-4** shows the process for installing a DIS Flume Plugin.

Figure 4-4 Process for installing a DIS Flume Plugin


Prepare for the installation.
Check Flume dependencies.
Check DIS streams.
Check authentication information.
Obtain the DIS Flume Plugin package.
Deploy DIS Flume Plugin.
Install DIS Flume Plugin.
Configure DIS Flume Plugin.
Verify DIS Flume Plugin.
(Optional) Uninstall DIS Flume Plugin.

4.3.2 Preparing for Installing a DIS Flume Plugin

Checking Dependencies

Step 1 Check that Flume is running properly.

Step 2 In the Flume installation directory, run the following command to check that the Flume version is 1.4.0 or later:

```
$ bin/flume-ng version | grep Flume
```

Step 3 Run the following command to check that the Java runtime environment (JRE) version is 1.8.0 or later:

```
java -version
```

----End

Checking DIS Streams

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click 📍 in the upper left corner and select a region and project.

Step 3 In the navigation tree, choose **Stream Management**.

Step 4 Ensure that at least one DIS stream is in the **Running** state and ready to receive incoming data.

----End

Checking Authentication Information

- AK/SK file

  Access Key ID/Secret Access Key (AK/SK) files are created by the Identity and Access Management (IAM) service to authenticate calls to application programming interfaces (APIs) on the public cloud. To obtain AK/SK, choose .

- ProjectID

  A project is a group of tenant resources. A tenant can have multiple projects, one for each region. Each region has a unique Project ID. To view the project IDs of different regions, choose **My Credentials** > **API Credentials**.

Obtaining a DIS Flume Plugin Package

Obtain the **dis-flume-Plugin-*X.X.X*.zip** package from **https://dis-publish.obs-website.cn-north-1.myhuaweicloud.com/**.

4.3.3 Installing a DIS Flume Plugin

Prerequisites

PuTTY has been installed.

Procedure

Step 1 Log in to the remote Flume server using PuTTY or another remote login tool.

Step 2 Run the following command to navigate to the Flume installation directory:

```
cd ${FLUME_HOME}
```

Step 3 Upload the **dis-flume-plugin-*X.X.X*.zip** installation package to this directory.

Step 4 Run the following command to decompress the installation package:

```
unzip dis-flume-plugin-X.X.X.zip
```

Step 5 Run the following command to navigate to the directory where the package is decompressed:

```
cd dis-flume-plugin
```

Step 6 Run the following command to execute the installation program:

bash install.sh

If the following information is displayed, the ***DIS Flume Plugin*** has been successfully installed in the **${FLUME_HOME}/plugin.d/dis-flume-plugin** directory:

```
Install dis-flume-plugin successfully.
```

----End

4.3.4 Configuring the DIS Flume Plugin

A DIS Flume Plugin consists of DIS Source and DIS Sink. The **dis-flume-plugin.conf.template** file in the installation package lists the configuration methods. This section describes the configuration items of DIS Source and DIS Sink.

NOTE

dis-flume-plugin.conf.template is a configuration sample for DIS plug-in and is not a configuration file that will be accessed when Flume is run. Flume provides a configuration sample file in *FLUME_HOME***/conf/flume-conf.properties.template**, where *FLUME_HOME* is the installation path of Flume. You can modify the configuration file based on site requirements.

SK is sensitive information. To encrypt the SK, perform the following steps:

Step 1 Run the following command to go to the **dis-flume-plugin/** directory:

cd /dis-flume-plugin

Step 2 Run the encryption script, enter the password, and press **Enter**.

bash dis-encrypt.sh

Step 3 View the encryption result. The character string following "Encrypt result:" displayed on the console is the encryption result. Use this method to encrypt the MySQL password and SK, respectively and record the ciphertext in the configuration file.

----End

Configuring DIS Source

Table 4-3 DIS Source configuration parameters

Parameter	Mandatory	Description	Default Value
channels	Yes	Name of the Flume channel.	-
type	Yes	DIS Source type.	com.cloud.dis.adapter.flume.source.DISSource

Parameter	Mandatory	Description	Default Value
streams	Yes	DIS stream name.	The entered DIS stream name must be the same as the stream name specified when you are creating a DIS stream on the DIS console.
ak	Yes	User's AK. For details about how to obtain an AK, see **Checking Authentication Information**.	-
sk	Yes	User's SK. For details about how to obtain an SK, see **Checking Authentication Information**.	-
region	Yes	Region in which the DIS is located.	-
projectId	Yes	Project ID specific to your region. For details about how to obtain a project ID, see **Checking Authentication Information**.	-
endpoint	Yes	Data API address of the region where DIS resides.	-
group.id	Yes	Application name, which is used to identify a consumer group and consists of letters, digits, hyphens (-), and underscores (_).	-

Configuring DIS Sink

Table 4-4 DIS Sink configuration parameters

Parameter	Mandatory	Description	Default Value
channel	Yes	Name of the Flume channel.	-
type	Yes	Sink type.	com.cloud.dis.adapter.flume.sink.DISSink
streamName	Yes	Name of the DIS stream.	The entered DIS stream name must be the same as the stream name specified when you are creating a DIS stream on the DIS console.
ak	Yes	User's AK. For details about how to obtain an AK, see **Checking Authentication Information**.	-
sk	Yes	User's SK. For details about how to obtain an SK, see **Checking Authentication Information**.	-
region	Yes	Region in which the DIS is located.	-
projectId	Yes	Project ID specific to your region. For details about how to obtain a project ID, see **Checking Authentication Information**.	-
endpoint	Yes	Data API address of the region where DIS resides.	-

Parameter	Mandatory	Description	Default Value
partitionNumber	No	Number of partitions that the chosen DIS stream has. The value is used to calculate batchSize.	1
batchSize	No	Number of data records that can be batch processed in a single Flume transaction.	batchSize = partitionNumber * 250
sendingThreadSize	No	The number of sender threads. By default, there is only one sender thread. **NOTE** If multiple sender threads are used, the following situations will occur: ● There is no guarantee on the order in which data will be sent. ● Certain data will be resent if the Flume application restarts after it stops abruptly.	1

Parameter	Mandatory	Description	Default Value
sendingRecordSize	No	Number of data records that can be sent in a single call to the DIS API that is used to put data into DIS streams. **NOTE** **batchSize** indicates the number of data records that can be batch processed in a single Flume transaction, whereas **sendingRecordSize** indicates the number of data records that can be batch processed in a single API call. For example, if **batchSize** is 1000 and **sendingRecordSize** is 250, it indicates that four API calls will be made to complete the Flume transaction. A Flume transaction is completed and submitted only after the **batchSize** amount of data is successfully sent. If the application restarts before a Flume transaction is submitted, data will be resent. If **sendingThreadSize** is set to **1**, it indicates that **sendingRecordSize** and **batchSize** will have the same value. This prevents unnecessary data resending.	250

Parameter	Mandatory	Description	Default Value
retrySize	No	The maximum number of times that the DIS Flume Sink can retry to call a DIS API when the initial call to the DIS API fails. The default value 2147483647 is recommended, indicating that Sink can retry the API call for an unlimited number of times. Exponential backoff is used to incrementally increase the wait between retry attempts in order to reduce server load and increase the likelihood that repeated requests will succeed.	2147483647

Parameter	Mandatory	Description	Default Value
resultLogLevel	No	The level of logs generated to print out the latest sequenceNumber at the end of each DIS API call. Log levels are listed in the order of from low to high: OFF < DEBUG < INFO < WARN < ERROR. The value **OFF** indicates that no logs will be generated. If the log level of Flume log4j is higher than **resultLogLevel**, no logs will be generated.	OFF
maxBufferAgeMillis	No	The maximum number of milliseconds that must elapse before data can be uploaded to the DIS. ● If the buffer is full with data waiting to be uploaded, data will be immediately uploaded to the DIS. ● If the buffer is not full, data will be uploaded to the DIS after the specified number of milliseconds elapses.	5000

Parameter	Mandatory	Description	Default Value
connectionTimeOutSeconds	No	The amount of time that must elapse before a DIS API call times out. Unit: second	30
socketTimeOutSeconds	No	The amount of time that must elapse before a response to a DIS API call times out. Unit: second	60
dataEncryptEnabled	No	An indicator of whether data is encrypted using the Advanced Encryption Standard (AES) algorithm. ● true ● false	false
dataPassword	No	Password used to encrypt or decrypt data. This parameter is mandatory if **dataEncryptEnabled** is set to **true**.	-

Parameter	Mandatory	Description	Default Value
bodySerializeType	No	Upload format of the DIS data packet (not original data format). Possible values: ● **json**: The DIS data packet is encapsulated in the format of JSON. ● **protobuf**: The DIS data packet is encapsulated in the 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

4.3.5 Testing a DIS Flume Plugin

Testing DIS Source

Step 1 Start PuTTY and log in to the server on which Flume is installed.

Step 2 Ensure that the configuration file containing information about the DIS Source is ready.

You can modify the configuration file based on the **flume-conf.properties.template** provided by Flume. The following provides a file sample:

```
agent.sources = dissource
agent.channels = memoryChannel
agent.sinks = loggerSink
#Define DIS Source (which is used to obtain data from DIS).
agent.sources.dissource.channels = memoryChannel
agent.sources.dissource.type = com.cloud.dis.adapter.flume.source.DISSource
agent.sources.dissource.streams = YOU_DIS_STREAM_NAME
agent.sources.dissource.ak = YOU_ACCESS_KEY_ID
agent.sources.dissource.sk = YOU_SECRET_KEY_ID
agent.sources.dissource.region = YOU_Region
agent.sources.dissource.projectId = YOU_PROJECT_ID
agent.sources.dissource.endpoint = https://dis.${region}.cloud.com
agent.sources.dissource.group.id = YOU_APP_NAME
#Define a stream.
agent.channels.memoryChannel.type = memory
agent.channels.memoryChannel.capacity = 10000
#Define Logger Sink (which is used to output data to the console).
agent.sinks.loggerSink.type = logger
agent.sinks.loggerSink.channel = memoryChannel
```

Step 3 Start Flume. For details about the startup command, see the instructions at the Apache Flume official website.

To start Flume from the Flume installation directory, run the following sample command:

```
bin/flume-ng agent --conf-file conf/flume-conf.properties.template --name agent --conf conf/ -
Dflume.root.logger=INFO,console
```

In the preceding information, **bin/flume-ng agent** indicates that Flume Agent will be started; **conf-file** indicates the path of the configuration file written by the user; **name** indicates the agent name in the configuration file; **conf** indicates the **conf/** path of Flume.

View the log file. If the log file contains information similar to "source disSource started", DIS Source starts normally. In the preceding information, **disSource** indicates the DIS Source name configured by the user.

Step 4 Check that DIS Source can successfully download data from DIS.

Upload data to a stream to which DIS Source points. If Flume does not report an error and DIS Sink can obtain data, the download is successful.

NOTE

If the sample configuration in **Step 2** is used, the data obtained from DIS is output to the console in byte array format.

Step 5 Log in to the DIS console. Two minutes later, check monitoring data of the DIS stream specified in **Table 4-3**. If data download (blue lines) is displayed, DIS Source is running successfully.

----End

Testing DIS Sink

Step 1 Start PuTTY and log in to the server on which Flume is installed.

Step 2 Ensure that the configuration file containing information about DIS Sink is ready.

You can modify the configuration file based on the **flume-conf.properties.template** provided by Flume. The following provides a file sample:

```
agent.sources = exec
agent.channels = memoryChannel
agent.sinks = dissink
#Define EXEC Source (which is used to monitor the dis.txt file in the tmp directory).
agent.sources.exec.type = exec
agent.sources.exec.command = tail -F /tmp/dis.txt
agent.sources.exec.shell = /bin/bash -c
agent.sources.exec.channels = memoryChannel
#Define a stream.
agent.channels.memoryChannel.type = memory
agent.channels.memoryChannel.capacity = 10000
# Define DIS Sink (which is used to output data to the DIS stream).
agent.sinks.dissink.channel = memoryChannel
agent.sinks.dissink.type = com.cloud.dis.adapter.flume.sink.DISSink
agent.sinks.dissink.streamName = YOU_DIS_STREAM_NAME
agent.sinks.dissink.ak = YOU_ACCESS_KEY_ID
agent.sinks.dissink.sk = YOU_SECRET_KEY_ID
agent.sinks.dissink.region = YOU_Region
agent.sinks.dissink.projectId = YOU_PROJECT_ID
agent.sinks.dissink.endpoint = https://dis.${region}.myhuaweicloud.com
agent.sinks.dissink.resultLogLevel = INFO
```

Step 3 Start Flume. For details about the startup command, see the instructions at the Apache Flume official website.

To start Flume from the Flume installation directory, run the following sample command:

```
bin/flume-ng agent --conf-file conf/flume-conf.properties.template --name agent --conf conf/ -
Dflume.root.logger=INFO,console
```

In the preceding information, **bin/flume-ng agent** indicates that Flume Agent will be started; **conf-file** indicates the path of the configuration file written by the user; **name** indicates the agent name in the configuration file; **conf** indicates the **conf/** path of Flume.

View the log file. If the log file contains information similar to "Dis flume sink [dissink] start", it indicates that DIS Sink starts normally. The value of **dissink** is the DIS Sink name configured by the user.

Step 4 Check that the DIS Sink can successfully upload data to DIS.

Ingest data into the Flume source. Set the **resultLogLevel** of the DIS Sink to a value that is not **OFF** and higher than the log level of log4j. If logs similar to the following are displayed, the DIS Flume Sink has successfully uploaded data to the DIS.

```
CurrentPut 5 events[success 5 / failed 0] spend 131 ms.
```

NOTE

If the sample configuration described in **Step 2** is used, you can create a **dis.txt** file in the **tmp** directory and append content to the file. After Flume is started, each row of the appended content is read by Flume and sent to the DIS stream through DIS Sink.

Step 5 Log in to the DIS console. Two minutes later, check monitoring data of the DIS stream specified in **Table 4-4**. If data upload (indicated in green lines) is displayed, DIS Sink is running successfully.

----End

4.3.6 (Optional) Uninstalling a DIS Flume Plugin

Procedure

Step 1 Start PuTTY and log in to the server on which Flume is installed.

Step 2 Stop the Flume program.

Step 3 Go to the directory where the DIS Flume Plugin is located.

cd ${FLUME_HOME}

cd dis-flume-plugin

Step 4 Run the following commands to uninstall the DIS Flume Plugin:

dos2unix install.sh

bash install.sh uninstall

If information similar to the following appears, the DIS Flume Sink has been successfully uninstalled:

```
Uninstall dis-flume-plugin successfully.
```

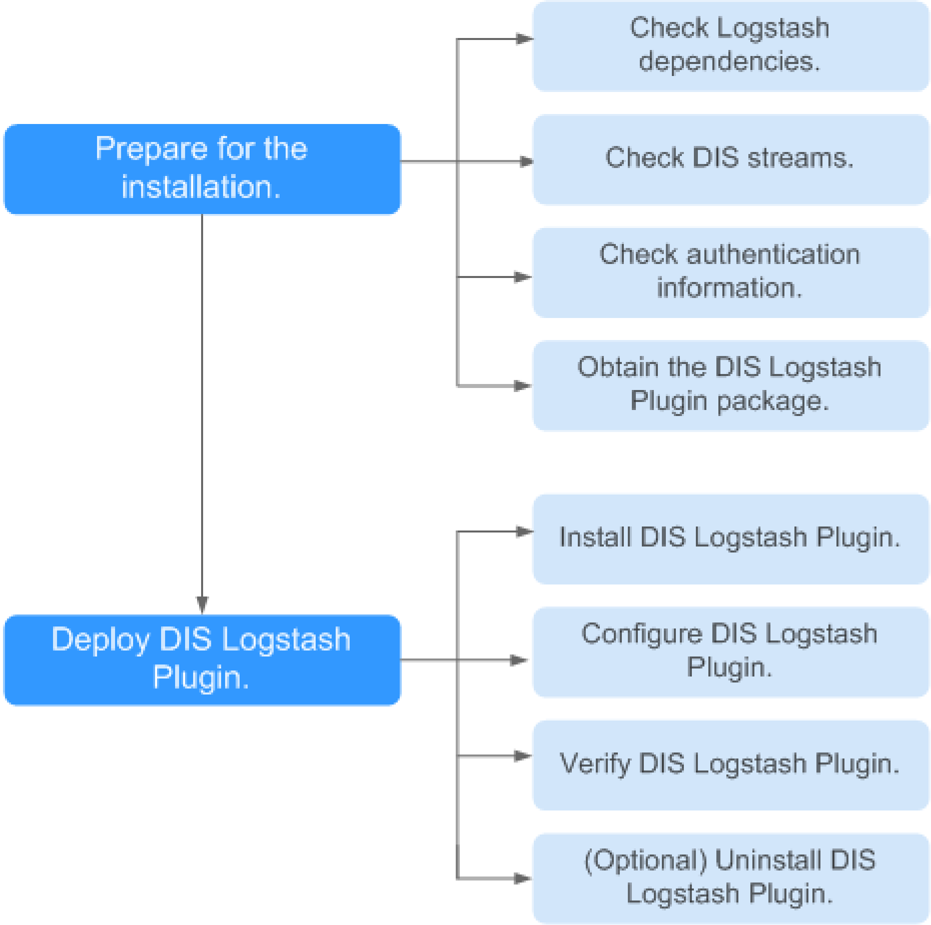
----End

4.4 Using a DIS Logstash Plugin to Upload and Download Data

4.4.1 Overview

A DIS Logstash Plugin is a Logstash plugin provided by DIS and consists of DIS Input and DIS Output. DIS Input is used to download data from DIS to Logstash; DIS Output is used to upload data from Logstash to DIS. **Figure 4-5** depicts the process for installing a DIS Logstash Plugin.

Figure 4-5 Installation flowchart


Prepare for the installation.
Check Logstash dependencies.
Check DIS streams.
Check authentication information.
Obtain the DIS Logstash Plugin package.
Deploy DIS Logstash Plugin.
Install DIS Logstash Plugin.
Configure DIS Logstash Plugin.
Verify DIS Logstash Plugin.
(Optional) Uninstall DIS Logstash Plugin.

4.4.2 Preparing for Installing a DIS Logstash Plugin

Checking Dependencies

Step 1 Check that Logstash is running properly.

Step 2 Ensure that the Java version is 1.8.0 or later. Run the following command to view the Java version:

```
java -version
```

Step 3 Ensure that the JRuby version is 9.0.0.0 or later. Run the following command to view the JRuby version:

```
$ bin/jruby -v
```

----End

Checking DIS Streams

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click ⊙ in the upper left corner and select a region and project.

Step 3 In the navigation tree, choose **Stream Management**.

Step 4 Ensure that at least one DIS stream is in the **Running** state and ready to receive incoming data.

----End

Checking Authentication Information

- AK/SK file

  Access Key ID/Secret Access Key (AK/SK) files are created by the Identity and Access Management (IAM) service to authenticate calls to application programming interfaces (APIs) on the cloud. To obtain AK/SK, choose **My Credentials** > **Access Keys**.

- Project ID

  A project is a group of tenant resources. A tenant can have multiple projects, one for each region. Each region has a unique Project ID. To view the project IDs of different regions, choose .

Obtaining a DIS Logstash Plugin Package

Obtain the **dis-logstash-Plugin-***X.X.X***.zip** package from **https://dis-publish.obs-website.cn-north-1.myhuaweicloud.com/**.

4.4.3 Installing DIS Logstash Plugin Online

DIS Logstash Plugin can be installed online or offline.

For online installation, you do not need to download the DIS Logstash Plugin package but directly connect to the public network.

Prerequisites

PuTTY has been installed.

Installing logstash-input-dis

Step 1 Start PuTTY or another remote login tool to log in to the server where Logstash is installed.

Step 2 Run the following command to navigate to the Logstash installation directory:

cd ${LOGSTASH_HOME}

Step 3 Run the following command to install **logstash-input-dis**:

bin/logstash-plugin install logstash-input-dis

If information similar to the following is displayed, the installation is successful:

```
Validating logstash-input-dis
Installing logstash-input-dis
Installation successful
```

----End

Installing logstash-output-dis

Step 1 Start PuTTY or another remote login tool to log in to the server where Logstash is installed.

Step 2 Run the following command to navigate to the Logstash installation directory:

cd ${LOGSTASH_HOME}

Step 3 Run the following command to install **logstash-output-dis**:

bin/logstash-plugin install logstash-output-dis

If information similar to the following is displayed, the installation is successful:

```
Validating logstash-output-dis
Installing logstash-output-dis
Installation successful
```

----End

4.4.4 Installing DIS Logstash Plugin Offline

DIS Logstash Plugin can be installed online or offline.

For offline installation, you need to obtain the DIS Logstash Plugin package and execute the installation script.

Prerequisites

PuTTY has been installed.

Procedure

Step 1 Start PuTTY or another remote login tool to log in to the server where Logstash is installed.

Step 2 Run the following command to navigate to the Logstash installation directory:

cd ${LOGSTASH_HOME}

Step 3 Upload the **dis-logstash-plugins-***X.X.X***.zip** installation package to this directory.

Step 4 Decompress the installation package.

unzip dis-logstash-plugins-X.X.X.zip

Step 5 Run the following command to navigate to the directory where the package is decompressed:

cd logstash-plugins

Step 6 Run the following command to run the installation program:

bash install.sh –p ${LOGSTASH_HOME}

If information similar to the following is displayed, the installation is successful:

```
Install dis-logstash-plugins successfully.
```

----End

4.4.5 Configuring the DIS Logstash Plugin

DIS Logstash Plugins consist of the Input and Output plugins. This section describes the configuration items of the plugins.

Configuring DIS Logstash Input

The configuration template (used to download data from a DIS stream to a local file) is as follows:

```
input
{
  dis {
      streams => ["YOUR_DIS_STREAM_NAME"]
      endpoint => "https://dis.${region}.myhuaweicloud.com"
      ak => "YOUR_ACCESS_KEY_ID"
      sk => "YOUR_SECRET_KEY_ID"
      region => "YOUR_Region"
      project_id => "YOUR_PROJECT_ID"
      group_id => "YOUR_APP_ID"
      client_id => "YOUR_CLIENT_ID"
      auto_offset_reset => "earliest"
   }
}
output
{
   file {
     path => ["/tmp/test.log"]
  }
}
```

Table 4-5 DIS Logstash Input configuration parameters

Parameter	Mandatory	Description	Default Value
stream	Yes	DIS stream name.	The entered DIS stream name must be the same as the stream name specified when you are creating a DIS stream on the DIS console.
ak	Yes	User's AK. For details about how to obtain an AK, see **Checking Authentication Information**.	-
sk	Yes	User's SK. For details about how to obtain an SK, see **Checking Authentication Information**.	-

Parameter	Mandatory	Description	Default Value
region	Yes	Region in which DIS is located.	-
project_id	Yes	Project ID specific to your region. For details about how to obtain a project ID, see **Checking Authentication Information**.	-
client_id	No	Client ID, which identifies a consumer in a consumer group. If multiple pipelines or Logstash instances are started for consumption, set this parameter to different values. For example, the value of instance 1 is **client1**, and the value of instance 2 is **client2**.	logstash
endpoint	Yes	Data API address of the region where DIS resides.	-
group_id	Yes	DIS App name, used to identify a consumer group. The value can be any character string.	-

Parameter	Mandatory	Description	Default Value
auto_offset_reset	No	Position where data starts to be consumed from the stream. The options are as follows: **earliest**: Data is consumed from the earliest one. **latest**: Data is consumed from the latest one.	latest

Configuring DIS Logstash Output

The configuration template (used to read data from a local file and upload it to a DIS stream) is as follows:

```
input
{
   file {
       path => ["/tmp/test.log"]
       type => "log4j"
       start_position => "beginning"
   }
}
output
{
    dis {
        stream => ["YOUR_DIS_STREAM_NAME"]
        endpoint => "https://dis.${region}.myhuaweicloud.com"
        ak => "YOUR_ACCESS_KEY_ID"
        sk => "YOUR_SECRET_KEY_ID"
        region => "YOUR_Region"
        project_id => "YOUR_PROJECT_ID"
    }
}
```

Table 4-6 DIS Logstash Output configuration parameters

Parameter	Mandatory	Description	Default Value
stream	Yes	DIS stream name.	The entered DIS stream name must be the same as the stream name specified when you are creating a DIS stream on the DIS console.

Parameter	Mandatory	Description	Default Value
ak	Yes	User's AK. For details about how to obtain an AK, see **Checking Authentication Information**.	-
sk	Yes	User's SK. For details about how to obtain an SK, see **Checking Authentication Information**.	-
region	Yes	Region in which DIS is located.	-
project_id	Yes	Project ID specific to your region. For details about how to obtain a project ID, see **Checking Authentication Information**.	-
body_compress_enabled	No	Specifies whether to enable data compression.	No

Parameter	Mandatory	Description	Default Value
body_compress_type	No	Data compression type. The following compression algorithms are supported: lz4: a compression algorithm with a fast compression speed and high compression efficiency snappy: a compression algorithm with a high compression speed and a reasonable compression rate, not for achieving maximum compression or being compatible with other compression formats zstd: a new lossless compression algorithm with a fast compression speed and high compression ratio	lz4

4.4.6 Testing the DIS Logstash Plugin

Testing DIS Logstash Input

Step 1 Start PuTTY and log in to the server on which Logstash is installed.

Step 2 Start the Logstash program.

```
bin/logstash -f dis_to_local.conf
```

In the preceding command, **-f** indicates the path of the configuration file.

Step 3 Check that DIS Input can successfully download data from DIS.

Upload data to a stream to which DIS Input points. If Logstash does not report an error and DIS Output can obtain data, the download is successful.

Step 4 Log in to the DIS console. Two minutes later, check monitoring data of the DIS stream specified in **Table 4-5**. If data download (blue lines) is displayed, DIS Logstash Input is running successfully.

----End

Testing DIS Logstash Output

Step 1 Start PuTTY and log in to the server on which Logstash is installed.

Step 2 Start the Logstash program.

```
bin/logstash -f local_to_dis.conf
```

In the preceding command, **-f** indicates the path of the configuration file.

Step 3 Check that DIS Logstash Output can successfully upload data to DIS.

Ingest data to the input end of Logstash. If Logstash does not report an error and can upload data to the specified stream, the upload is successful.

Step 4 Log in to the DIS console. Two minutes later, check monitoring data of the DIS stream specified in **Table 4-6**. If data upload (green lines) is displayed, DIS Logstash Output is running successfully.

----End

4.4.7 (Optional) Uninstalling the DIS Logstash Plugin

Procedure

Step 1 Start PuTTY and log in to the server on which Logstash is installed.

Step 2 Stop the Logstash program.

Step 3 Go to the directory where the DIS Logstash Plugin is located.

```
cd ${LOGSTASH_HOME}
```

```
cd logstash-plugins
```

Step 4 Run the following command to uninstall the DIS Logstash Plugin:

```
bash uninstall.sh –p ${LOGSTASH_HOME}
```

If information similar to the following appears, the DIS Logstash Plugin has been successfully uninstalled:

```
Uninstall dis-logstash-plugins successfully.
```

----End

4.5 Using Kafka Adapter to Upload and Download Data

4.5.1 Kafka Adapter Overview

dis-kafka-adapter is a software development kit (SDK) provided by DIS. Users who originally use Kafka Client can use dis-kafka-adapter instead to upload data to DIS in the similar way.

4.5.2 Preparations

Configuring the pom.xml File

If a maven project exists, use the following dependency in **pom.xml**:

```
<dependency>
   <groupId>com.huaweicloud.dis</groupId>
   <artifactId>huaweicloud-dis-kafka-adapter</artifactId>
   <version>1.2.18</version>
</dependency>
```

Using the DIS Sample Project

Download the package from **https://dis-publish.obs-website.cn-north-1.myhuaweicloud.com/**.

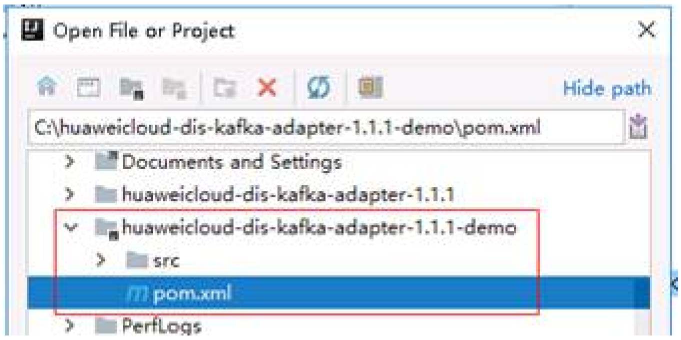
The .zip package contains two directories.

- The **huaweicloud-dis-kafka-adapter-***X.X.X* directory contains all JAR packages. If a non-maven project is used, import all JAR packages in the lib directory to the environment.
- **huaweicloud-dis-kafka-adapter-**-*X.X.X***-demo** is a sample project and is compiled using maven.

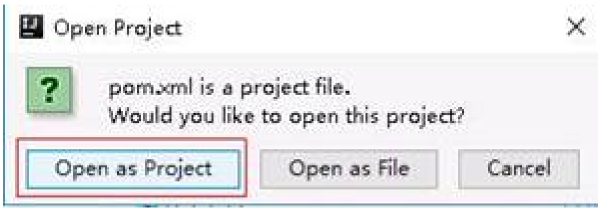
You can use IntelliJ IDEA to import the sample project as follows:

Step 1 Run IntelliJ IDEA and choose **File** > **Open**.

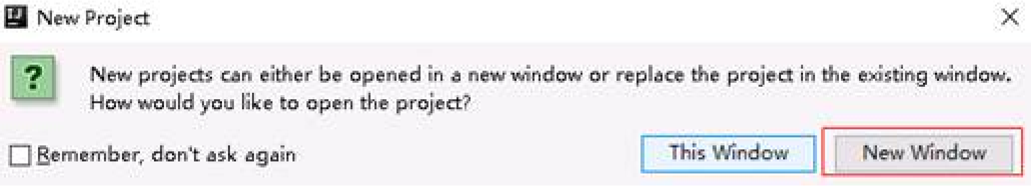
In the displayed dialog box, expand **huaweicloud-dis-kafka-adapter-***X.X.X***-demo** and double-click **pom.xml**.


Open File or Project
Hide path
C:\huaweicloud-dis-kafka-adapter-1.1.1-demo\pom.xml
Documents and Settings
huaweicloud-dis-kafka-adapter-1.1.1
huaweicloud-dis-kafka-adapter-1.1.1-demo
src
pom.xml
PerfLogs

Step 2 When the following dialog box is displayed, select **Open as Project**.


Open Project
pom.xml is a project file.
Would you like to open this project?
Open as Project
Open as File
Cancel

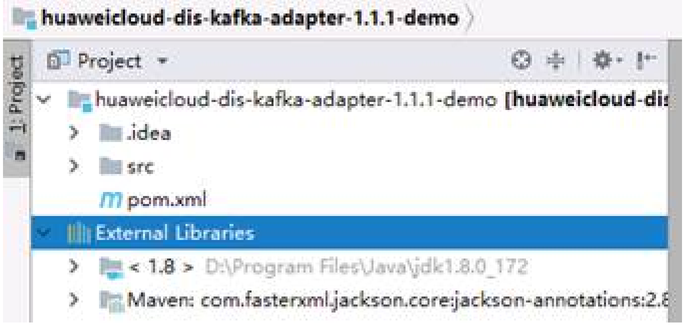
Step 3 Click **New Window** to open the project in a new window.



New Project

New projects can either be opened in a new window or replace the project in the existing window. How would you like to open the project?

Step 4 Wait when IntelliJ IDEA is building the project. After the project is built, the directory and files are displayed.



huaweicloud-dis-kafka-adapter-1.1.1-demo

```
Project
huaweicloud-dis-kafka-adapter-1.1.1-demo [huaweicloud-di
  .idea
  src
  pom.xml
External Libraries
  < 1.8 > D:\Program Files\Java\jdk1.8.0_172
  Maven: com.fasterxml.jackson.core:jackson-annotations:2.8
```

----End

Checking Authentication Information

- AK/SK file

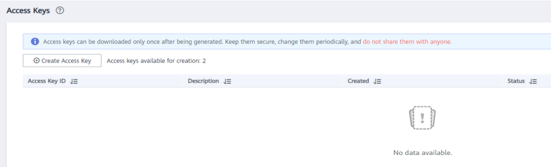
  Access Key ID/Secret Access Key (AK/SK) files are created by the Identity and Access Management (IAM) service to authenticate calls to application programming interfaces (APIs) on the public cloud.

  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 4-6**.

  Figure 4-6 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.

- Project 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 ID from the project list.

4.5.3 Uploading Data

Sample Code

For details about how to obtain the AK, SK, and project ID, see **Checking Authentication Information**.

```
package com.huaweicloud.dis.demo.adapter;
import com.huaweicloud.dis.DISConfig;
import com.huaweicloud.dis.adapter.kafka.clients.producer.*;
import com.huaweicloud.dis.adapter.kafka.common.serialization.StringSerializer;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import java.util.Properties;
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.Future;
import java.util.concurrent.ThreadLocalRandom;

public class DISKafkaProducerDemo
{
    private static final Logger LOGGER = LoggerFactory.getLogger(DISKafkaProducerDemo.class);


    public static void main(String[] args)
    {

        // There will be security risks if the AK/SK used for authentication is directly written into code. Encrypt the AK/SK in the configuration file or environment variables for storage;
       // In this example, the AK and SK stored in the environment variables are used for identity authentication. Before running this example, configure environment variables HUAWEICLOUD_SDK_AK and HUAWEICLOUD_SDK_SK in the local environment.
       String ak = System.getenv("HUAWEICLOUD_SDK_AK");
       String sk = System.getenv("HUAWEICLOUD_SDK_SK");
       // YOU ProjectId
       String projectId = "YOU_PROJECT_ID";
      // YOU DIS Stream
       String streamName = "YOU_STREAM_NAME";
      // Consumption group ID, which is used to record the offset.
      String groupId = "YOU_GROUP_ID";
     // DIS region
      String region = "your region";
```

```
        Properties props = new Properties();
        props.setProperty(DISConfig.PROPERTY_AK, ak);
        props.setProperty(DISConfig.PROPERTY_SK, sk);
        props.setProperty(DISConfig.PROPERTY_PROJECT_ID, projectId);
        props.setProperty(DISConfig.PROPERTY_REGION_ID, region);
        props.setProperty(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class.getName());
        props.setProperty(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
StringSerializer.class.getName());

//By default, the domain name is automatically used for access instead of configuring an endpoint. If an
endpoint is required, remove the following comments and set the endpoint:
        // props.setProperty(DISConfig.PROPERTY_ENDPOINT, "https://dis-${region}.myhuaweicloud.com");

        // Create dis producer
        Producer<String, String> producer = new DISKafkaProducer<>(props);

                    //Send data synchronously.
        synchronousSendDemo(producer, streamName);

                //Send data asynchronously.
        asynchronousSendDemo(producer, streamName);

// Disable the producer to prevent resource leakage.
        producer.close();
    }

    public static void synchronousSendDemo(Producer<String, String> producer, String streamName)
    {
        LOGGER.info("===== synchronous send =====");
        for (int i = 0; i < 5; i++)
        {
 //If the key is set to Random or Null, data is evenly distributed to all partitions.
            String key = String.valueOf(ThreadLocalRandom.current().nextInt(1000000));
            String value = "Hello world[sync]. " + i;

            Future<RecordMetadata> future = producer.send(new ProducerRecord<>(streamName, key, value));

            try
            {
 //Calling future.get will block waiting until sending is complete.
                RecordMetadata recordMetadata = future.get();
//Data is successfully sent.
                LOGGER.info("Success to send [{}], Partition [{}], Offset [{}].",
                        value, recordMetadata.partition(), recordMetadata.offset());
            }
            catch (Exception e)
            {
//Data failed to be sent.
                LOGGER.error("Failed to send [{}], Error [{}]", value, e.getMessage(), e);
            }
        }
    }

    public static void asynchronousSendDemo(Producer<String, String> producer, String streamName)
    {
        LOGGER.info("===== asynchronous send =====");
        int totalSendCount = 5;
        CountDownLatch countDownLatch = new CountDownLatch(totalSendCount);
        for (int i = 0; i < totalSendCount; i++)
        {
 //If the key is set to Random or Null, data is evenly distributed to all partitions.
            String key = String.valueOf(ThreadLocalRandom.current().nextInt(1000000));
            String value = "Hello world[async]. " + i;

            try
            {
//Data is sent in callback mode and is not blocked.
                producer.send(new ProducerRecord<>(streamName, key, value), new Callback()
                {
                    @Override
                    public void onCompletion(RecordMetadata recordMetadata, Exception e)
                    {
                        countDownLatch.countDown();
                        if (e == null)
                        {
//Data is successfully sent.
                            LOGGER.info("Success to send [{}], Partition [{}], Offset [{}].",
                                    value, recordMetadata.partition(), recordMetadata.offset());
                        }
                        else
                        {
//Data failed to be sent.
                            LOGGER.error("Failed to send [{}], Error [{}]", value, e.getMessage(), e);
                        }
                    }
                });
            }
            catch (Exception e)
            {
                countDownLatch.countDown();
                LOGGER.error(e.getMessage(), e);
            }
        }

        try
        {
// Wait until all data is sent.
            countDownLatch.await();
        }
        catch (InterruptedException e)
        {
            LOGGER.error(e.getMessage(), e);
        }
    }
}
```

After running the preceding program, if the data is successfully sent, the following log is generated:

```
09:32:52.001 INFO  c.h.d.d.a.DISKafkaProducerDemo - ===== synchronous send =====
09:32:53.523 INFO  c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[sync]. 0], Partition [0],
Offset [114].
09:32:53.706 INFO  c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[sync]. 1], Partition [0],
Offset [115].
09:32:53.956 INFO  c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[sync]. 2], Partition [0],
Offset [116].
09:32:54.160 INFO  c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[sync]. 3], Partition [0],
Offset [117].
09:32:54.450 INFO  c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[sync]. 4], Partition [0],
Offset [118].
09:32:54.450 INFO  c.h.d.d.a.DISKafkaProducerDemo - ===== asynchronous send =====
09:32:54.673 INFO  c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[async]. 0], Partition [0],
Offset [119].
09:32:54.674 INFO  c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[async]. 1], Partition [0],
Offset [120].
09:32:54.674 INFO  c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[async]. 2], Partition [0],
Offset [121].
09:32:54.674 INFO  c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[async]. 3], Partition [0],
Offset [122].
09:32:54.674 INFO  c.h.d.d.a.DISKafkaProducerDemo - Success to send [Hello world[async]. 4], Partition [0],
Offset [123].
```

Adaptation to the Native KafkaProducer API

DISKafkaProducer is implemented in a different way from KafkaProducer. The DISKafkaProducer client and server are implemented through the REST API, whereas KafkaProducer is implemented based on TCP. Their API compatibility differences are as follows:

Table 4-7 Adaptation description

Native KafkaProducer	Type	DISKafkaProducer	Description
Future<RecordMetadata> send(ProducerRecord<K, V> record)	API	Supported	Send a single data record.
Future<RecordMetadata> send(ProducerRecord<K, V> record, Callback callback)	API	Supported	Send a single data record and set the callback processing function.
void close()	API	Supported	Disable Producer.
void close(long timeout, TimeUnit timeUnit)	API	Supported	Disable Producer and set the timeout period.
List<PartitionInfo> partitionsFor(String topic)	API	Supported	Obtain the partition information of the stream.
void flush(long timeout, TimeUnit unit)	API	Not supported	Forcibly send the current cached data.
Map<MetricName, ? extends Metric> metrics()	API	Not supported	Obtain statistics.

Native KafkaProducer	Type	DISKafkaProducer	Description
key.serializer	Parameter	Supported	The meaning of this parameter is the same as that in Kafka. The default value is **StringSerializer**. In Kafka, this parameter has no default value, and you must configure a value for it.
value.serializer	Parameter	Supported	The meaning of this parameter is the same as that in Kafka. The default value is **StringSerializer**. In Kafka, this parameter has no default value, and you must configure a value for it.
linger.ms	Parameter	Supported	The meaning of this parameter is the same as that in Kafka. The default value is **50**. In Kafka, the default value is **0**. This parameter is configured for improving the upload efficiency of the REST API.

Native KafkaProducer	Type	DISKafkaProducer	Description
batch.size	Parameter	Supported	The meaning of this parameter is the same as that in Kafka. The default value is 1 MB. In Kafka, the default value is 16 KB. This parameter is configured for matching the flow control.
buffer.memory	Parameter	Supported	Same as the default setting in Kafka, which is 32 MB.
max.in.flight.requests.per.connection	Parameter	Supported	Limit the maximum number of not-responded requests that can be sent by a client on a single connection. The default value is **100**. In Kafka, the default value is 5. This parameter is configured for improving the sending performance. However, the data sequence may be inconsistent. You are advised to set this parameter to 1 to ensure data sequence.

Native KafkaProducer	Type	DISKafkaProducer	Description
block.on.buffer.full	Parameter	Supported	Same as the default setting in Kafka, which is **false**. ● true: When the sending buffer is full, sending is blocked and does not time out. ● false: After the sending buffer is full, data is blocked based on max.block.ms. If the time is exceeded, an exception is issued.
max.block.ms	Parameter	Supported	Same as the default setting in Kafka, which is **60000**. When the sending buffer is full and **block.on.buffer.full** is **false**, control the block time (ms) of send ().
retries	Parameter	Supported, but the parameter name is changed to **exception.retries**.	The default value in Kafka is **0**, and the default value in DIS is **8**. Number of retry attempts that are made when the network or server is abnormal.
Others	Parameter	Not supported	-

4.5.4 Consumption Modes

Similar to Kafka, dis kafka adapter supports three consumption modes.

assign Mode

Users specify the partitions to be consumed by the consumer instance are manually. In this case, the group management mechanism is not used. That is, when the number of consumers in the group changes or the stream scaling is performed, the partitions will not be reallocated. A code example is provided as follows:

```
package com.huaweicloud.dis.demo.adapter;

import com.huaweicloud.dis.DISConfig;
import com.huaweicloud.dis.adapter.kafka.clients.consumer.*;
import com.huaweicloud.dis.adapter.kafka.common.PartitionInfo;
import com.huaweicloud.dis.adapter.kafka.common.TopicPartition;
import com.huaweicloud.dis.adapter.kafka.common.serialization.StringDeserializer;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.Properties;

public class DISKafkaConsumerAssignDemo
{
    private static final Logger LOGGER = LoggerFactory.getLogger(DISKafkaConsumerAssignDemo.class);

    public static void main(String[] args)
  {

        // There will be security risks if the AK/SK used for authentication is directly written into code. Encrypt
the AK/SK in the configuration file or environment variables for storage;
       // In this example, the AK and SK stored in the environment variables are used for identity
authentication. Before running this example, configure environment variables HUAWEICLOUD_SDK_AK and
HUAWEICLOUD_SDK_SK in the local environment.
       String ak = System.getenv("HUAWEICLOUD_SDK_AK");
       String sk = System.getenv("HUAWEICLOUD_SDK_SK");
       // YOU ProjectId
       String projectId = "YOU_PROJECT_ID";
      // YOU DIS Stream
       String streamName = "YOU_STREAM_NAME";
//Consumption group ID, which is used to record the offset.
      String groupId = "YOU_GROUP_ID";
     // DIS region
      String region = "your region";

        Properties props = new Properties();
        props.setProperty(DISConfig.PROPERTY_AK, ak);
        props.setProperty(DISConfig.PROPERTY_SK, sk);
        props.setProperty(DISConfig.PROPERTY_PROJECT_ID, projectId);
        props.setProperty(DISConfig.PROPERTY_REGION_ID, region);
        props.setProperty(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
        props.setProperty(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
        props.setProperty(ConsumerConfig.GROUP_ID_CONFIG, groupId);
        props.setProperty(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "false");
        props.setProperty(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG,
OffsetResetStrategy.LATEST.name());


//By default, the domain name is automatically used for access instead of configuring an endpoint. If an
endpoint is required, remove the following comments and set the endpoint:
        // props.setProperty(DISConfig.PROPERTY_ENDPOINT, "https://dis-${region}.myhuaweicloud.com");

        Consumer<String, String> consumer = new DISKafkaConsumer<>(props);
        List<TopicPartition> topicPartitions = new ArrayList<>();
        for (PartitionInfo partitionInfo : consumer.partitionsFor(streamName))
        {
            topicPartitions.add(new TopicPartition(partitionInfo.topic(), partitionInfo.partition()));
        }

//Use the assign mode to specify the partition to be consumed.
        consumer.assign(topicPartitions);

        while (true)
        {
            try
            {
                ConsumerRecords<String, String> records = consumer.poll(Long.MAX_VALUE);

                if (!records.isEmpty())
                {
                    for (TopicPartition partition : records.partitions())
                    {
                        List<ConsumerRecord<String, String>> partitionRecords = records.records(partition);
                        for (ConsumerRecord<String, String> record : partitionRecords)
                        {
                            LOGGER.info("Value [{}], Partition [{}], Offset [{}], Key [{}]",
                                    record.value(), record.partition(), record.offset(), record.key());
                        }
                    }
//Submit the current offset asynchronously after data processing is complete or submit commitSync
synchronously.
                    consumer.commitAsync(new OffsetCommitCallback()
                    {
                        @Override
                        public void onComplete(Map<TopicPartition, OffsetAndMetadata> map, Exception e)
                        {
                            if (e == null)
                            {
                                LOGGER.debug("Success to commit offset [{}]", map);
                            }
                            else
                            {
                                LOGGER.error("Failed to commit offset [{}]", e.getMessage(), e);
                            }
                        }
                    });
                }
            }
            catch (Exception e)
            {
                LOGGER.info(e.getMessage(), e);
            }
        }
    }
}
```

After running the preceding program, if the data is sent to the stream, the following log is generated:

```
09:36:45.071 INFO  c.h.d.a.k.c.DISKafkaConsumer - create DISKafkaConsumer successfully
09:36:49.842 INFO  c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[sync]. 0], Partition [0],
Offset [134], Key [769066]
09:36:49.963 INFO  c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[sync]. 1], Partition [0],
Offset [135], Key [700005]
09:36:50.145 INFO  c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[sync]. 2], Partition [0],
Offset [136], Key [338044]
09:36:51.093 INFO  c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[sync]. 3], Partition [0],
Offset [137], Key [537495]
09:36:51.093 INFO  c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[sync]. 4], Partition [0],
Offset [138], Key [980016]
09:36:51.093 INFO  c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[async]. 0], Partition [0],
Offset [139], Key [182772]
09:36:51.093 INFO  c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[async]. 1], Partition [0],
Offset [140], Key [830271]
09:36:51.093 INFO  c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[async]. 2], Partition [0],
Offset [141], Key [927054]
09:36:51.093 INFO  c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[async]. 3], Partition [0], 
Offset [142], Key [268122]
09:36:51.093 INFO  c.h.d.d.a.DISKafkaConsumerAssignDemo - Value [Hello world[async]. 4], Partition [0], 
Offset [143], Key [992787]
```

subscribe Mode

Users only need to specify stream names without specifying specific partitions. The server will trigger the group management mechanism based on the number of consumers or stream scaling changes to automatically allocate partitions to each consumer. This ensures that a partition is consumed by only one consumer.

A code example is provided as follows:

```
package com.huaweicloud.dis.demo.adapter;

import com.huaweicloud.dis.DISConfig;
import com.huaweicloud.dis.adapter.kafka.clients.consumer.*;
import com.huaweicloud.dis.adapter.kafka.common.TopicPartition;
import com.huaweicloud.dis.adapter.kafka.common.serialization.StringDeserializer;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.Collections;
import java.util.List;
import java.util.Map;
import java.util.Properties;

public class DISKafkaConsumerSubscribeDemo
{
    private static final Logger LOGGER = LoggerFactory.getLogger(DISKafkaConsumerSubscribeDemo.class);

    public static void main(String[] args)
    {

        // There will be security risks if the AK/SK used for authentication is directly written into code. Encrypt 
the AK/SK in the configuration file or environment variables for storage;
       // In this example, the AK and SK stored in the environment variables are used for identity 
authentication. Before running this example, configure environment variables HUAWEICLOUD_SDK_AK and 
HUAWEICLOUD_SDK_SK in the local environment.
       String ak = System.getenv("HUAWEICLOUD_SDK_AK");
       String sk = System.getenv("HUAWEICLOUD_SDK_SK");
       // YOU ProjectId
       String projectId = "YOU_PROJECT_ID";
      // YOU DIS Stream
       String streamName = "YOU_STREAM_NAME";
//Consumption group ID, which is used to record the offset.
      String groupId = "YOU_GROUP_ID";
     // DIS region
      String region = "your region";

        Properties props = new Properties();
        props.setProperty(DISConfig.PROPERTY_AK, ak);
        props.setProperty(DISConfig.PROPERTY_SK, sk);
        props.setProperty(DISConfig.PROPERTY_PROJECT_ID, projectId);
        props.setProperty(DISConfig.PROPERTY_REGION_ID, region);
        props.setProperty(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, 
StringDeserializer.class.getName());
        props.setProperty(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, 
StringDeserializer.class.getName());
        props.setProperty(ConsumerConfig.GROUP_ID_CONFIG, groupId);
        props.setProperty(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "false");
        props.setProperty(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, 
OffsetResetStrategy.LATEST.name());
//By default, the domain name is automatically used for access instead of configuring an endpoint. If an
endpoint is required, remove the following comments and set the endpoint:
        // props.setProperty(DISConfig.PROPERTY_ENDPOINT, "https://dis-${region}.myhuaweicloud.com");

        Consumer<String, String> consumer = new DISKafkaConsumer<>(props);
      //When using the subscribe mode, specify the name of the stream to be consumed.
        consumer.subscribe(Collections.singleton(streamName));

        while (true)
        {
            try
            {
                ConsumerRecords<String, String> records = consumer.poll(Long.MAX_VALUE);

                if (!records.isEmpty())
                {
                    for (TopicPartition partition : records.partitions())
                    {
                        List<ConsumerRecord<String, String>> partitionRecords = records.records(partition);
                        for (ConsumerRecord<String, String> record : partitionRecords)
                        {
                            LOGGER.info("Value [{}], Partition [{}], Offset [{}], Key [{}]",
                                    record.value(), record.partition(), record.offset(), record.key());
                        }
                    }
//Submit the current offset asynchronously after data processing is complete or submit commitSync
synchronously.
                    consumer.commitAsync(new OffsetCommitCallback()
                    {
                        @Override
                        public void onComplete(Map<TopicPartition, OffsetAndMetadata> map, Exception e)
                        {
                            if (e == null)
                            {
                                LOGGER.debug("Success to commit offset [{}]", map);
                            }
                            else
                            {
                                LOGGER.error("Failed to commit offset [{}]", e.getMessage(), e);
                            }
                        }
                    });
                }
            }
            catch (Exception e)
            {
                LOGGER.info(e.getMessage(), e);
            }
        }
    }
}
```

When the program runs, it sends a heartbeat request (Heartbeat) every 10 seconds and then sends a (JoinGroup) request to join the consumer group. The server starts to allocate partitions to consumers in the consumer group. This process takes about 20s. After the operation is complete, the consumers send synchronization requests (SyncGroup) to obtain allocation results. If the information about heartbeat {"state": "STABLE"} is recorded in the log, it indicates that allocation to the entire consumer group has been completed, data is ready to be consumed.

The key logs in this process are described as follows:

- Heartbeat {"state":"JOINING"}

Heartbeat: indicates a heartbeat request. The heartbeat request is initiated every 10 seconds and is used to keep a connection with the server. If the server does not receive the heartbeat message within 1 minute, it considers that the consumer is offline and the partitions will be reallocated to the consumers in the consumer group. If the heartbeat result is JOINING, the consumer needs to join the consumer group again. If the heartbeat result is STABLE, the consumer group is stable.

- JoinGroup

If the heartbeat result is not STABLE, the consumer sends a joinGroup request to notify the server that it needs to join the consumer group. After receiving the join request from the client, the server will reallocate partitions for the consumer group. In this case, syncDelayedTimeMs is returned, indicating the allocation duration. After allocation is completed, the client sends a synchronization request (SyncGroup) to obtain the allocation result.

- SyncGroup

A SyncGroup request is used to obtain the allocation result. The returned assignment contains the stream name and partition to be consumed.

Run the sample program. After allocation is completed, send data to the stream. The complete log is as follows:

```
09:42:37.296 INFO  c.h.d.a.k.c.DISKafkaConsumer - create DISKafkaConsumer successfully
09:42:37.354 INFO  c.h.d.a.k.c.Coordinator - Heartbeat {"state":"JOINING"}
09:42:37.363 INFO  c.h.d.a.k.c.Coordinator - joinGroupRequest {"groupId":"ding","clientId":"consumer-
c2d43144-0823-4eea-aaa8-7af95c536144","interestedStream":["liuhao12"]}
09:42:37.406 INFO  c.h.d.a.k.c.Coordinator - joinGroupResponse {"state":"OK","syncDelayedTimeMs":21000}
09:42:58.408 INFO  c.h.d.a.k.c.Coordinator - syncGroup {"groupId":"ding","clientId":"consumer-
c2d43144-0823-4eea-aaa8-7af95c536144","generation":-1}
09:42:58.451 INFO  c.h.d.a.k.c.Coordinator - syncGroup {"state":"OK","generation":33,"assignment":{"dis-
test":[0]}}
09:42:58.488 INFO  c.h.d.a.k.c.Coordinator - Heartbeat {"state":"STABLE"}
09:43:08.960 INFO  c.h.d.a.k.c.Coordinator - Heartbeat {"state":"STABLE"}
09:46:56.227 INFO  c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[sync]. 0], Partition [0],
Offset [144], Key [799704]
09:46:56.327 INFO  c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[sync]. 1], Partition [0],
Offset [145], Key [683757]
09:46:56.449 INFO  c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[sync]. 2], Partition [0],
Offset [146], Key [439062]
09:46:56.535 INFO  c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[sync]. 3], Partition [0],
Offset [147], Key [374939]
09:46:56.654 INFO  c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[sync]. 4], Partition [0],
Offset [148], Key [321528]
09:46:56.749 INFO  c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[async]. 0], Partition
[0], Offset [149], Key [964841]
09:46:56.749 INFO  c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[async]. 1], Partition
[0], Offset [150], Key [520262]
09:46:56.749 INFO  c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[async]. 2], Partition
[0], Offset [151], Key [619119]
09:46:56.749 INFO  c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[async]. 3], Partition
[0], Offset [152], Key [257094]
09:46:56.749 INFO  c.h.d.d.a.DISKafkaConsumerSubscribeDemo - Value [Hello world[async]. 4], Partition
[0], Offset [153], Key [310331]
```

subscribePattern Mode

Users specify a wildcard instead of specifying a stream name. For example, stream.* indicates that stream1, stream2, stream_123, and so on are consumed. An existing, added, or deleted stream can be consumed by a consumer group as long as it matches the wildcard.

A code example is provided as follows:

```
package com.huaweicloud.dis.demo.adapter;

import com.huaweicloud.dis.DISConfig;
import com.huaweicloud.dis.adapter.kafka.clients.consumer.*;
import com.huaweicloud.dis.adapter.kafka.common.TopicPartition;
import com.huaweicloud.dis.adapter.kafka.common.serialization.StringDeserializer;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.Collection;
import java.util.List;
import java.util.Map;
import java.util.Properties;
import java.util.regex.Pattern;

public class DISKafkaConsumerSubscribePatternDemo
{
    private static final Logger LOGGER =
LoggerFactory.getLogger(DISKafkaConsumerSubscribePatternDemo.class);

    public static void main(String[] args)
    {
        // There will be security risks if the AK/SK used for authentication is directly written into code. Encrypt
the AK/SK in the configuration file or environment variables for storage;
        // In this example, the AK and SK stored in the environment variables are used for identity
authentication. Before running this example, configure environment variables HUAWEICLOUD_SDK_AK and
HUAWEICLOUD_SDK_SK in the local environment.
       String ak = System.getenv("HUAWEICLOUD_SDK_AK");
       String sk = System.getenv("HUAWEICLOUD_SDK_SK");
       // YOU ProjectId
       String projectId = "YOU_PROJECT_ID";
      // YOU DIS Stream
       String streamName = "YOU_STREAM_NAME";
//Consumption group ID, which is used to record the offset.
      String groupId = "YOU_GROUP_ID";
      // DIS region
      String region = "your region";

        Properties props = new Properties();
        props.setProperty(DISConfig.PROPERTY_AK, ak);
        props.setProperty(DISConfig.PROPERTY_SK, sk);
        props.setProperty(DISConfig.PROPERTY_PROJECT_ID, projectId);
        props.setProperty(DISConfig.PROPERTY_REGION_ID, region);
        props.setProperty(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
        props.setProperty(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
        props.setProperty(ConsumerConfig.GROUP_ID_CONFIG, groupId);
        props.setProperty(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "false");
        props.setProperty(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG,
OffsetResetStrategy.LATEST.name());

//By default, the domain name is automatically used for access instead of configuring an endpoint. If an
endpoint is required, remove the following comments and set the endpoint:
        // props.setProperty(DISConfig.PROPERTY_ENDPOINT, "https://dis-${region}.myhuaweicloud.com");

        Consumer<String, String> consumer = new DISKafkaConsumer<>(props);
//With the subscribePattern mode, you only need to specify a wildcard.
        consumer.subscribe(Pattern.compile(streamNamePattern), new ConsumerRebalanceListener()
        {
            @Override
            public void onPartitionsRevoked(Collection<TopicPartition> collection)
            {
                LOGGER.info("onPartitionsRevoked [{}]", collection);
            }

            @Override
            public void onPartitionsAssigned(Collection<TopicPartition> collection)
            {
                LOGGER.info("onPartitionsAssigned [{}]", collection);
            }
        });

        while (true)
        {
            try
            {
                ConsumerRecords<String, String> records = consumer.poll(Long.MAX_VALUE);

                if (!records.isEmpty())
                {
                    for (TopicPartition partition : records.partitions())
                    {
                        List<ConsumerRecord<String, String>> partitionRecords = records.records(partition);
                        for (ConsumerRecord<String, String> record : partitionRecords)
                        {
                            LOGGER.info("Value [{}], Partition [{}], Offset [{}], Key [{}]",
                                    record.value(), record.partition(), record.offset(), record.key());
                        }
                    }
//Submit the current offset asynchronously after data processing is complete or submit commitSync
synchronously.
                    consumer.commitAsync(new OffsetCommitCallback()
                    {
                        @Override
                        public void onComplete(Map<TopicPartition, OffsetAndMetadata> map, Exception e)
                        {
                            if (e == null)
                            {
                                LOGGER.debug("Success to commit offset [{}]", map);
                            }
                            else
                            {
                                LOGGER.error("Failed to commit offset [{}]", e.getMessage(), e);
                            }
                        }
                    });
                }
            }
            catch (Exception e)
            {
                LOGGER.info(e.getMessage(), e);
            }
        }
    }
}
```

After the program runs, wait about 20s until the allocation is completed. The data can be consumed. The following is an example code execution log:

```
10:10:36.420 INFO  c.h.d.a.k.c.DISKafkaConsumer - create DISKafkaConsumer successfully
10:10:36.481 INFO  c.h.d.a.k.c.Coordinator - Heartbeat {"state":"JOINING"}
10:10:36.486 INFO  c.h.d.a.k.c.Coordinator - joinGroupRequest {"groupId":"ding","clientId":"consumer-
cad967ba-70ab-4e02-b184-f60b95fe3256","streamPattern":"stream.*"}
10:10:36.697 INFO  c.h.d.a.k.c.Coordinator - joinGroupResponse {"state":"OK","subscription":
["stream_hello","stream_world"],"syncDelayedTimeMs":21000}
10:10:57.699 INFO  c.h.d.a.k.c.Coordinator - syncGroup {"groupId":"ding","clientId":"consumer-
cad967ba-70ab-4e02-b184-f60b95fe3256","generation":-1}
10:10:57.746 INFO  c.h.d.a.k.c.Coordinator - syncGroup {"state":"OK","generation":34,"assignment":
{"stream_hello":[0],"stream_world":[0]}}
10:10:57.770 INFO  c.h.d.d.a.DISKafkaConsumerSubscribePatternDemo - onPartitionsAssigned
[[stream_hello-0, stream_world-0]]
10:10:57.770 INFO  c.h.d.a.k.c.Coordinator - Heartbeat {"state":"STABLE"}
10:11:08.466 INFO  c.h.d.a.k.c.Coordinator - Heartbeat {"state":"STABLE"}
10:11:09.992 INFO  c.h.d.d.a.DISKafkaConsumerSubscribePatternDemo - Value [Hello world[sync]. 0],
Partition [0], Offset [154], Key [181881]
10:11:09.993 INFO  c.h.d.d.a.DISKafkaConsumerSubscribePatternDemo - Value [Hello world[sync]. 1],
Partition [0], Offset [155], Key [483023]
10:11:09.993 INFO  c.h.d.d.a.DISKafkaConsumerSubscribePatternDemo - Value [Hello world[sync]. 2],
Partition [0], Offset [156], Key [32453]
10:11:10.093 INFO  c.h.d.d.a.DISKafkaConsumerSubscribePatternDemo - Value [Hello world[sync]. 3], 
Partition [0], Offset [157], Key [111948]
10:11:10.180 INFO  c.h.d.d.a.DISKafkaConsumerSubscribePatternDemo - Value [Hello world[sync]. 4], 
Partition [0], Offset [158], Key [822860]
```

4.5.5 Consumption Offset

Two offset committing policies are available: automatic and manual. When creating a DISKafkaConsumer object, set the **enable.auto.commit** parameter to specify a desired offset committing policy. If the value is set to **true**, automatic offset committing is used. If the value is set to **false**, manual offset committing is used.

With automatic offset committing, the consumer enables the coordinator to commit offsets every *auto.commit.interval.ms*. With manual offset committing, instead of relying on the consumer to periodically commit consumed offsets, users can control when records should be considered as consumed and hence commit their offsets.

- Automatic

  When a consumer is created, automatic offset committing is set by default. The default committing interval is 5,000 ms. Parameters for automatic offset committing are as follows:

```
Props.setProperty("enable.auto.commit", "true");//Automatic offset committing is used.
Props.setProperty("auto.commit.interval.ms", "5000");//Offsets are committed at an interval of 5,000 ms.
```

- Automatic

In some scenarios, offsets need to be more strictly managed to ensure that messages are not repeatedly consumed or are not lost. For example, the pulled messages need to be written to the database for processing, or are used for complex service processing such as processing other network access requests. In such scenarios, the messages are regarded as successfully consumed only after all services are processed. In this case, you must manually control offset committing. Parameters for manual offset committing are as follows:

```
props.put("enable.auto.commit", "false");//Manual offset committing is used.
```

After the services are successfully processed, call the commitAsync() or commitSync() method to commit offsets. commitAsync() is used to commit offsets asynchronously. With this method, the consumer threads will not be blocked, and the next pull operation may be started before the offset committing result is returned. To obtain the committing result, add the OffsetCommitCallback method. After the offsets are committed, the onComplete() method is automatically called for processing of different logics based on the callback results.

CommitSync() is used to commit offsets synchronously. With this method, the consumer thread will be blocked until the offset committing result is returned.

In addition, the specific offset data of the specific partition may be further controlled. The confirmed offset is the maximum offset of the accepted data plus 1. For example, when a batch of data is consumed and the offset of the last record is 100, commit offset 101. In this case, consumption starts from the record whose offset is 101 and data will not be consumed repeatedly. The code sample is as follows:

```
ConsumerRecords<String, String> records = consumer.poll(Long.MAX_VALUE);

```

```
if (!records.isEmpty())
{
    for (TopicPartition partition : records.partitions())
    {
        List<ConsumerRecord<String, String>> partitionRecords = records.records(partition);
        for (ConsumerRecord<String, String> record : partitionRecords)
        {
            LOGGER.info("Value [{}], Partition [{}], Offset [{}], Key [{}]",
                    record.value(), record.partition(), record.offset(), record.key());
        }
        if (!partitionRecords.isEmpty())
        {
// Confirm the specific offset of a partition synchronously.
            long lastOffset = partitionRecords.get(partitionRecords.size() - 1).offset();
            consumer.commitSync(Collections.singletonMap(partition, new OffsetAndMetadata(lastOffset + 
1)));
        }
    }
}
```

4.5.6 Adaptation to the Native KafkaConsumer API

Table 4-8 API adaptation description

Native KafkaConsumer	Type	DISKafkaConsumer	Description
Set<TopicPartition> assignment()	API	Supported	Obtain information about the consumed stream and partition.
Set<String> subscription()	API	Supported	Obtain the name of the stream that has been subscribed to by the consumer.
void assign(Collection<TopicPartition> var1)	API	Supported	Allocate a specified partition.
void subscribe(Collection<String> var1)	API	Supported	Subscribe to a specified stream.
void subscribe(Collection<String> var1, ConsumerRebalanceListener var2)	API	Supported	Subscribe to a specified stream and supports callback of ConsumerRebalanceListener.

Native KafkaConsumer	Type	DISKafkaConsumer	Description
void subscribe(Pattern var1, ConsumerRebalanceListener var2)	API	Supported	Subscribe to all streams that match wildcards and supports callback of ConsumerRebalanceListener.
void unsubscribe()	API	Supported	Cancels all subscription.
ConsumerRecords <K, V> poll(long var1)	API	Supported	The message is obtained. Checksum (CRC32 check value of the message), serializedKeySize (byte length after key serialization), and serializedValue-Size (byte length after key serialization).
void commitSync()	API	Supported	Commit the consumed offsets synchronously.
void commitSync(final Map<TopicPartition, OffsetAndMetadata> offsets)	API	Supported	Commit the specified offset synchronously.
void commitAsync()	API	Supported	Commit the consumed offsets asynchronously.
public void commitAsync(OffsetCommitCallback callback)	API	Supported	Commit the current consumed offset asynchronously and supports callback of OffsetCommitCall-back.

Native KafkaConsumer	Type	DISKafkaConsumer	Description
void commitAsync(Map<TopicPartition, OffsetAndMetadata> offsets, OffsetCommitCallback callback)	API	Supported	Commit the specified offset asynchronously and supports callback of OffsetCommitCall-back.
void seek(TopicPartition partition, long offset)	API	Supported	Set the specified offset for a partition.
void seekToBeginning(Collection<TopicPartition> partitions)	API	Supported	Set the earliest value for the partition offset.
void seekToEnd(Collection<TopicPartition> partitions)	API	Supported	Set the latest value for the partition offset.
long position(TopicPartition partition)	API	Supported	Obtain the offset of the current consumed data in the partition.
OffsetAndMetadata committed(TopicPartition partition)	API	Supported	Obtain the committed offset of the partition.
List<PartitionInfo> partitionsFor(String topic)	API	Supported	Obtain the partition information of the stream, but leader, replicas, and inSyncReplicas in PartitionInfo are not implemented.
Map<String, List<PartitionInfo>> listTopics()	API	Supported	Obtain the stream information, but leader, replicas, and inSyncReplicas in PartitionInfo are not implemented.

Native KafkaConsumer	Type	DISKafkaConsumer	Description
void pause(Collection<TopicPartition> partitions)	API	Supported	Suspends the partition consumption.
void resume(Collection<TopicPartition> partitions)	API	Supported	Resumes the partition consumption.
Set<TopicPartition> paused()	API	Supported	Obtain all partitions that stop being consumed.
close()	API	Supported	Disable the consumer.
Map<MetricName, ? extends Metric> metrics()	API	Not supported	Obtain statistics.
wakeup()	API	Not supported	The internal implementation principles are different.
group.id	Parameter	Supported	Consumer group ID.
client.id	Parameter	Supported	**client.id** of each consumer must be unique. If **client.id** is not specified, the dis kafka consumer will generate a UUID as a **client.id**.
key.deserializer	Parameter	Supported	The meaning of this parameter is the same as that in Kafka. The default value is **StringDeserializer**. In Kafka, **StringDeserializer** has no default value, and you must configure a value for it.

Native KafkaConsumer	Type	DISKafkaConsumer	Description
value.deserializer	Parameter	Supported	The meaning of this parameter is the same as that in Kafka. The default value is **StringDeserializer**. In Kafka, this parameter has no default value, and you must configure a value for it.
enable.auto.commit	Parameter	Supported	The default value is the same as that in Kafka, which is **true**. ● **true**: The automatic offset committing is enabled and offsets are automatically committed at an interval of *auto.commit.interval.ms*. ● **false**: Offsets are not automatically committed.
auto.commit.interval.ms	Parameter	Supported	Interval for automatically committing offsets, in milliseconds. The default value is **5000**.

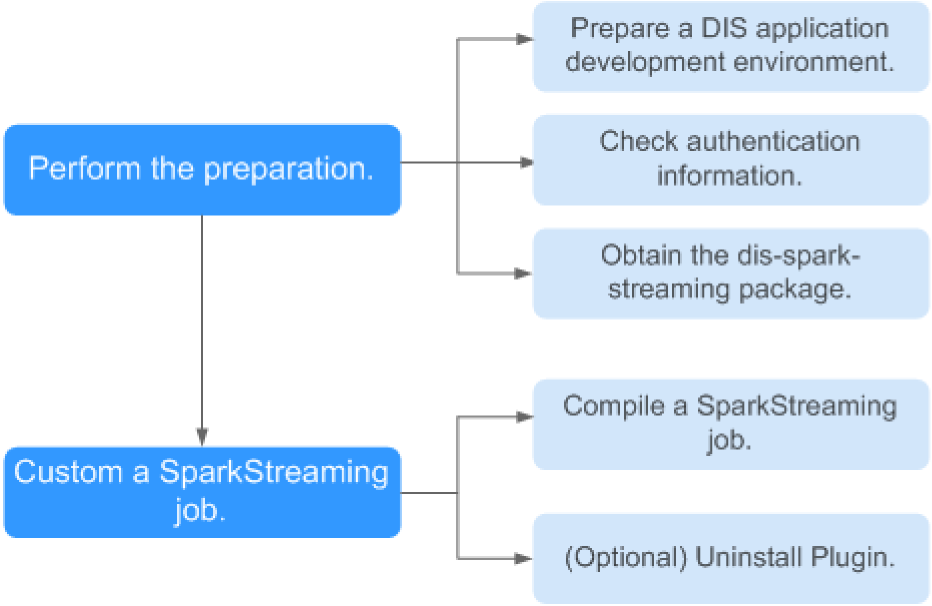
Native KafkaConsumer	Type	DISKafkaConsumer	Description
auto.offset.reset	Parameter	Supported	The default value is the same as that in Kafka, which is **latest**. This parameter is used to automatically set the offset position when there is no initial offset or the offset is incorrect. ● **earliest**: The offset is automatically reset to the earliest value. ● **latest**: The offset is automatically reset to the latest value. ● **none**: If the previous offset is not found in the consumer group, an exception is issued to the consumer.
Others	Parameter	Not supported	-

4.6 Using SparkStreaming SDK to Download Data

4.6.1 Overview

DIS Spark Streaming is a software development kit (SDK) provided by DIS to create a DStream that uses the DIS as a data source and runs with SparkStreaming. **Figure 4-7** depicts the process for using DIS Spark Streaming.

Figure 4-7 DIS Spark Streaming usage process


Perform the preparation.
Prepare a DIS application development environment.
Check authentication information.
Obtain the dis-spark-streaming package.
Custom a SparkStreaming job.
Compile a SparkStreaming job.
(Optional) Uninstall Plugin.

4.6.2 Preparing a DIS Spark Streaming Environment

Preparing a DIS Application Development Environment

Step 1 Set up DIS application development environments by following the procedure in **Step 1: Creating a DIS Stream**.

Step 2 Install Maven. Configure a local repository address.

Step 3 Install scala-sdk.

----End

Configuring the DIS Spark Streaming Dependency

The DIS Spark Streaming dependency can be introduced to the project through the following configuration:

```
<dependency>
   <groupId>com.cloud.dis</groupId>
   <artifactId>cloud-dis-spark-streaming_2.11</artifactId>
   <version>1.2.1</version>
   <scope>compile</scope>
</dependency>
```

Checking Authentication Information

- AK/SK file

  Access Key ID/Secret Access Key (AK/SK) files are created by the Identity and Access Management (IAM) service to authenticate calls to application programming interfaces (APIs) on the cloud. To obtain AK/SK, choose **My Credentials > Access Keys**.

- Project ID

A project is a group of tenant resources. A tenant can have multiple projects, one for each region. Each region has a unique Project ID. To view the project IDs of different regions, choose .

4.6.3 Customizing a SparkStreaming Job

Obtaining the DIS Spark Streaming Demo

Step 1 Obtain the **dis-spark-streaming-***X.X.X***.zip** package from **https://dis-publish.obs-website.cn-north-1.myhuaweicloud.com/**. Decompress the **dis-spark-streaming-***X.X.X***.zip** package to obtain the following directory that contains a Maven project sample:

dis-spark-streaming-demo: The **dis-spark-streaming-demo** directory contains a maven project sample.

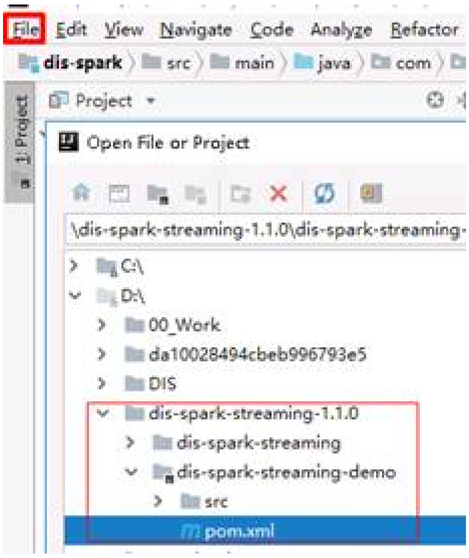
----End

Building a SparkStreaming Job

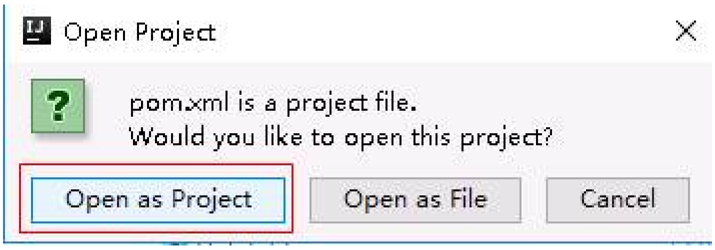
The following uses the IntelliJ IDEA community version as an example to describe how to build a SparkStreaming job. Make sure that the SparkStreaming job has been configured on the IDEA.

- JDK 1.8+
- Scala-sdk-2.11
- Maven 3.3.*

Step 1 Start IntelliJIDEA and choose **File** > **Open**. Select the **dis-spark-streaming-demo** directory and double-click **pom.xml**.



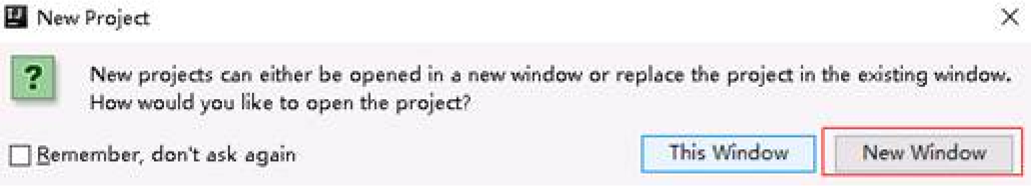
Step 2 When the following dialog box is displayed, select **Open as Project**.



Open Project

pom.xml is a project file.
Would you like to open this project?

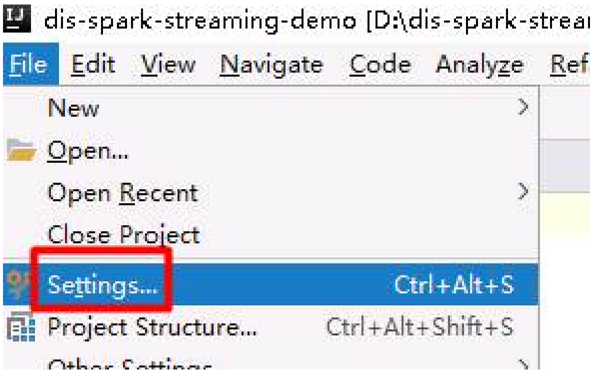
Step 3 Click **New Window** to open the project in a new window.



New Project

New projects can either be opened in a new window or replace the project in the existing window.
How would you like to open the project?

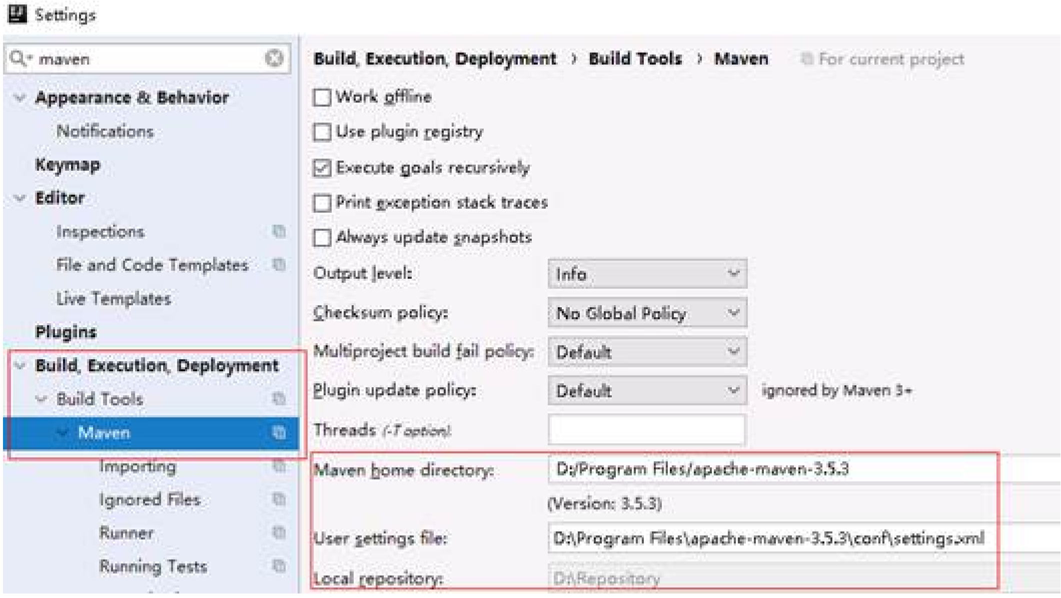
Step 4 In the displayed IDEA window, choose **File** > **Settings**.



Step 5 Enter **maven** in the search box and find the maven configuration. Ensure that **Maven home directory** (maven installation path), **User settings file** (**settings.xml** file location), and **Local repository** (local repository address) are correctly configured.

NOTE

If configurations are incorrect, modify them. Otherwise, the SDK installed in **Step 2** cannot be found.

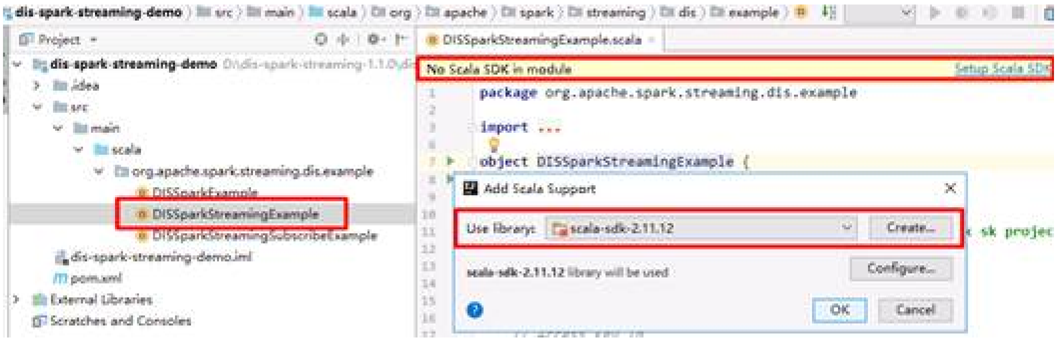


Build, Execution, Deployment › Build Tools › Maven   For current project

□ Work offline
□ Use plugin registry
☑ Execute goals recursively
□ Print exception stack traces
□ Always update snapshots

Output level: Info
Checksum policy: No Global Policy
Multiproject build fail policy: Default
Plugin update policy: Default   ignored by Maven 3+
Threads (-T option):
Maven home directory: D:/Program Files/apache-maven-3.5.3
(Version: 3.5.3)
User settings file: D:\Program Files\apache-maven-3.5.3\conf\settings.xml
Local repository: D:\Repository

Step 6 Open the **DISSparkStreamingExample** file. If the IDEA displays the message "No Scala SDK in module", click **Setup Scala SDK** next to it and select one from the Scala SDK list. If no Scala SDK is available, create one and associate it with the Scala path. Version 2.11 can be selected.


dis-spark-streaming-demo › src › main › scala › org › apache › spark › streaming › dis › example
Project
dis-spark-streaming-demo D:\dis-spark-streaming-1.1.0\d
.idea
src
main
scala
org.apache.spark.streaming.dis.example
DISSparkExample
DISSparkStreamingExample
DISSparkStreamingSubscribeExample
dis-spark-streaming-demo.iml
pom.xml
External Libraries
Scratches and Consoles
DISSparkStreamingExample.scala
Setup Scala SDK
Add Scala Support
Create...
Configure...
OK
Cancel

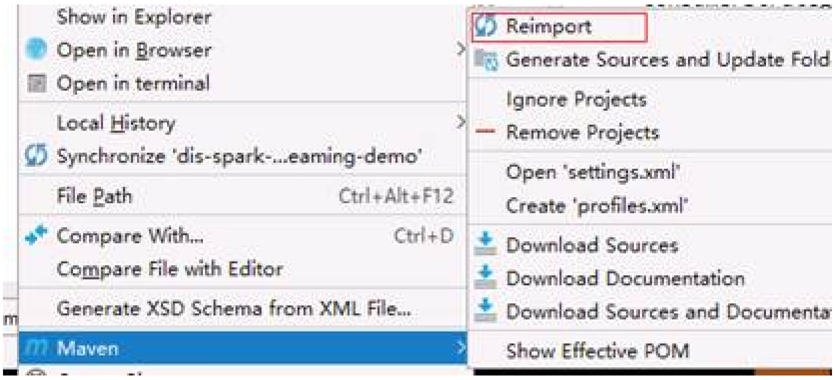
No Scala SDK in module

```
package org.apache.spark.streaming.dis.example

import ...

object DISSparkStreamingExample {
```

Use library: scala-sdk-2.11.12

scala-sdk-2.11.12 library will be used

Step 7 Right-click **pom.xml** and choose **Maven** > **Reimport** from the shortcut menu to import the maven dependency library again.



Step 8 If there is no error in the **DISSparkStreamingExample** file opened by IDEA, the development environment is configured successfully. The logic of this file is to read the data in DIS streams and count the occurrence times of each word.

1. DISSparkStreamingExample: an example of using the Assign mode. It does not have the capability to start from the last stop. The SDK construction method is as follows:

```
ConsumerStrategies.Assign[String, String](streamName, params, startingOffsets)
```

- **streamName**: a DIS stream name.
- **params**: a set of parameter maps. It includes at least **endpoint** (DIS gateway address), **region** (region where DIS resides), **ak** (user AK), **sk** (user SK), and **projectId** (user project ID).
- **startingOffsets**: start position for reading DIS data. LATEST indicates read from the latest data. EARLIEST indicates reading from the oldest data. If the exact start position of each partition needs to be specified, the value can be a JSON character string. For example, {"0" :23, "1" :-1, "2" :-2}, which indicates that the start position of partition 0 is 23, the first partition starts from the location of the latest data, and the second partition starts from the position of the earliest data. If there is no specified location for a partition, the system starts from the latest data position by default.

2. DISSparkStreamingSubscribeExample: an example of using the Subscribe mode. It has the capability to start from the last stop. The SDK construction method is as follows:

```
ConsumerStrategies.Subscribe[String, String](Array(streamName), params)
```

- **streamName**: a DIS stream name.
- **params**: a set of parameter maps. It includes at least **endpoint** (DIS gateway address), **region** (region where DIS resides), **ak** (user AK), **sk** (user SK), **projectId** (user project ID), and **group.id** (app name, indicating a consumer group). It can also include **auto.offset.reset**. The meaning of this parameter is the same as that of **startingOffsets** in Assign mode. If **enable.auto.commit** is set to **true**, the system automatically submits the offset every 5000 ms (which can be modified by setting **auto.commit.interval.ms**). If the parameter is set to **false**, the system does not automatically submit the offset. You can manually invoke **commitAsync** to submit the offset. For details, see the following part in the sample code:

```
stream.foreachRDD { rdd =>
    val offsetRanges = rdd.asInstanceOf[HasOffsetRanges].offsetRanges
    // commit offset to DIS async.
    stream.asInstanceOf[CanCommitOffsets].commitAsync(offsetRanges)
}
```

----End

Testing a SparkStreming Job

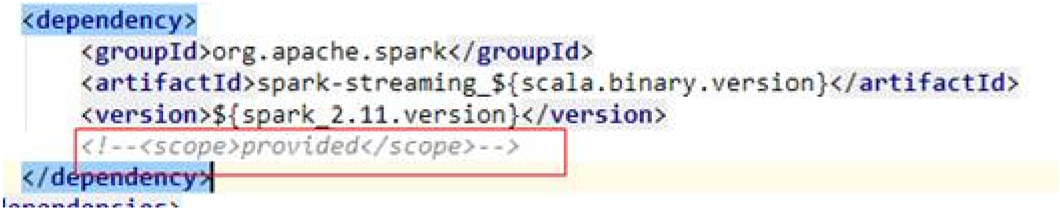
This section describes how to test a SparkStreming job in the local IDE to understand the basic usage of the SDK. In a real-world scenario, the SparkStreming job needs to run on a Spark cluster. After the test is complete, you can create clusters (such as MRS clusters) and submit a job for verification.

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click 📍 in the upper left corner of the page and select a region and project.

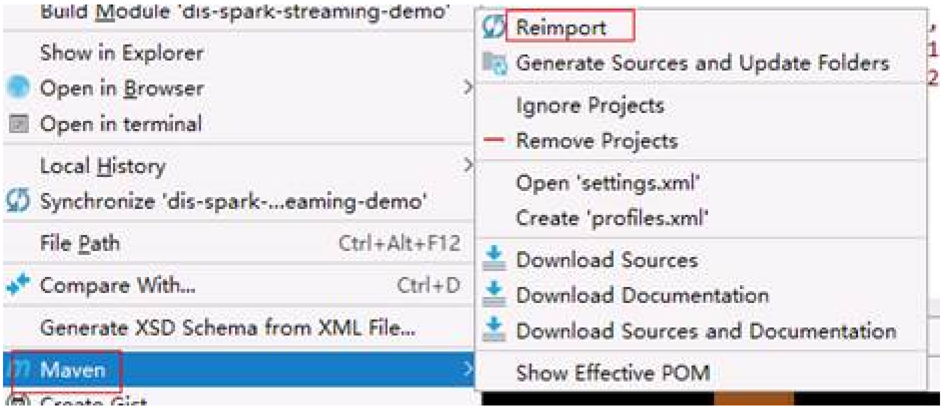
Step 3 Create a DIS stream by referring to **Step 1: Creating a DIS Stream** and continuously upload data to the newly created DIS stream. In this example, the content to be uploaded is **hello world**.

Step 4 Open the **pom.xml** file, press **Ctrl+/** to comment out the **<scope>provided</scope>** row, and save the setting.

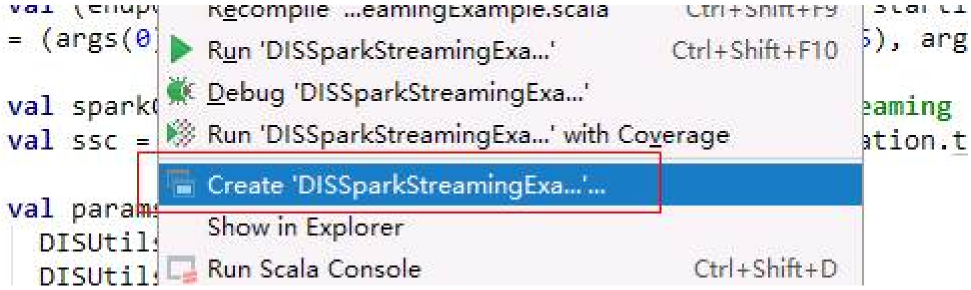


```
<dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-streaming_${scala.binary.version}</artifactId>
    <version>${spark_2.11.version}</version>
    <!--<scope>provided</scope>-->
</dependency>
```

Step 5 Right-click **pom.xml** and choose **Maven** > **Reimport** from the shortcut menu to import the dependency package again.



Step 6 Right-click in the **DISSparkStreamingExample** file and choose **Create 'DISSparkStreamingExample'** from the shortcut menu.

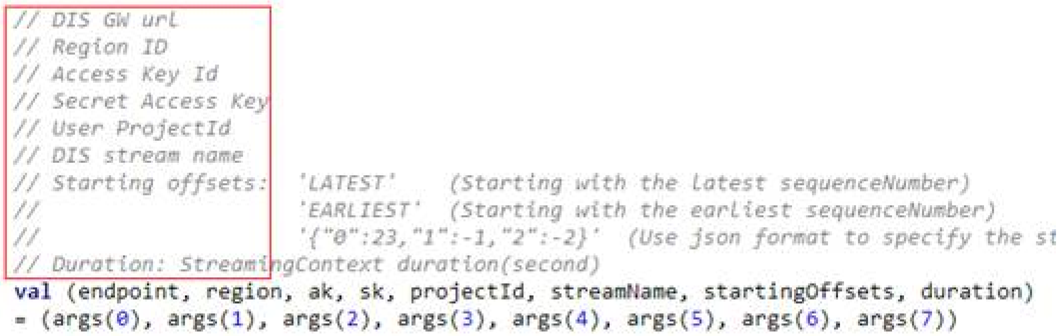

Recompile '...eamingExample.scala' Ctrl+Shift+F9
Run 'DISSparkStreamingExa...' Ctrl+Shift+F10
Debug 'DISSparkStreamingExa...'
Run 'DISSparkStreamingExa...' with Coverage
Create 'DISSparkStreamingExa...'...
Show in Explorer
Run Scala Console Ctrl+Shift+D

```
= (args(0 ...), arg
val spark( ...eaming
val ssc = ...ation.t
val params
  DISUtil
  DISUtil
```

Step 7 On the displayed configuration page, enter **-Dspark.master=local[*]** in **VM options**, indicating that a Spark job runs in local mode. Enter running parameters in **Program arguments** in the following format:

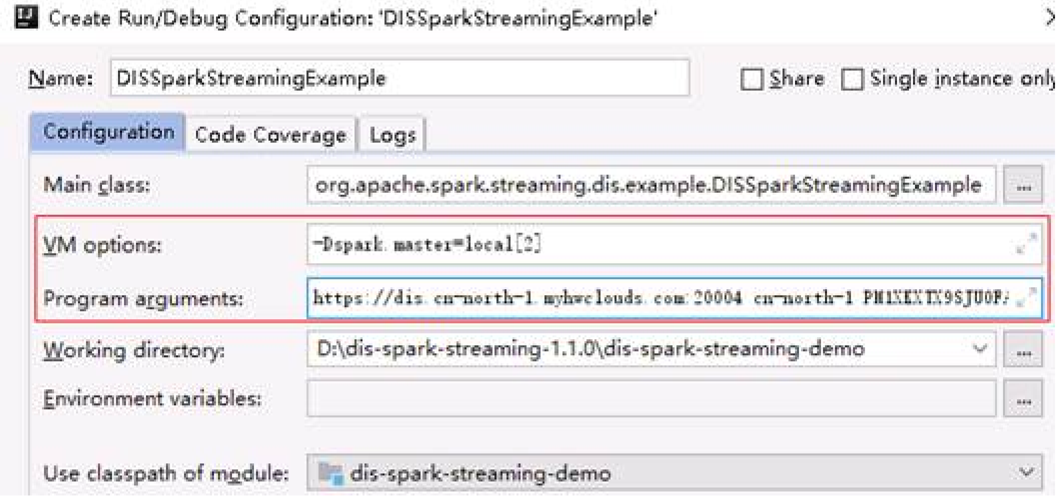
```
DIS gateway address Region name AK SK ProjectID Stream name Start position Streaming batch time

https://dis.${region}.cloud.com ${region} YOU_AK YOU_SK YOU_PROJECTID YOU_STREAM_NAME latest 10
```

The parameter sequence and meaning are available in the sample code. For details, see the following information:



```
// DIS GW url
// Region ID
// Access Key Id
// Secret Access Key
// User ProjectId
// DIS stream name
// Starting offsets:  'LATEST'    (Starting with the latest sequenceNumber)
//                    'EARLIEST'  (Starting with the earliest sequenceNumber)
//                    '{"0":23,"1":-1,"2":-2}'  (Use json format to specify the st
// Duration: StreamingContext duration(second)
val (endpoint, region, ak, sk, projectId, streamName, startingOffsets, duration)
= (args(0), args(1), args(2), args(3), args(4), args(5), args(6), args(7))
```

The final configurations of the IDEA are shown in the following figure. After confirming that all configurations are correct, click **OK** to close the window.



Create Run/Debug Configuration: 'DISSparkStreamingExample'

Name: DISSparkStreamingExample   □ Share   □ Single instance only

Configuration | Code Coverage | Logs

Main class: org.apache.spark.streaming.dis.example.DISSparkStreamingExample

VM options: -Dspark.master=local[2]

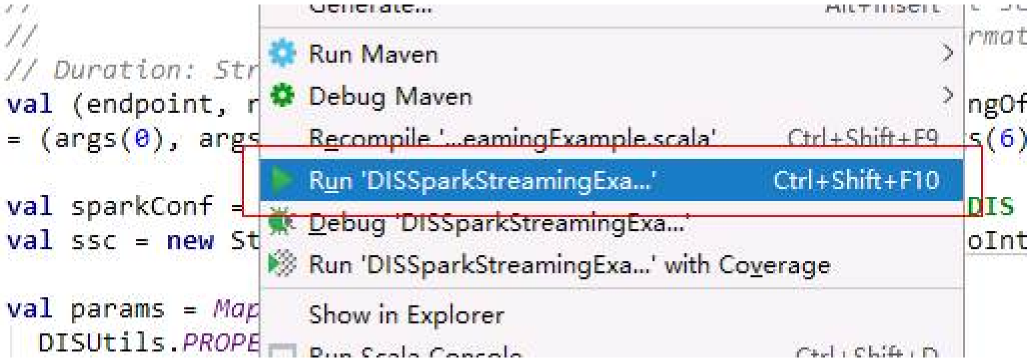
Program arguments: https://dis.cn-north-1.myhwclouds.com:20004 cn-north-1 PM1XEXTX9SJU0F:

Working directory: D:\dis-spark-streaming-1.1.0\dis-spark-streaming-demo

Environment variables:

Use classpath of module: dis-spark-streaming-demo

Step 8 Right-click in the **DISSparkStreamingExample** file and choose **Run 'DISSparkStreamingExample'** from the shortcut menu to start the job.



Generate... Alt+Insert
Run Maven
Debug Maven
Recompile '...eamingExample.scala' Ctrl+Shift+F9
Run 'DISSparkStreamingExa...' Ctrl+Shift+F10
Debug 'DISSparkStreamingExa...'
Run 'DISSparkStreamingExa...' with Coverage
Show in Explorer
Run Scala Console Ctrl+Shift+D

```
//
// Duration: Str
val (endpoint, r
= (args(0), args

val sparkConf =
val ssc = new St

val params = Map
  DISUtils.PROPE
```

Step 9 During the startup, an error message "hadoop binary path" is displayed, which can be ignored.

```
18/08/28 10:26:10 ERROR Shell: Failed to locate the winutils binary in the hadoop binary path
java.io.IOException: Could not locate executable null\bin\winutils.exe in the Hadoop binaries.
```

Step 10 If there are no other errors, the job runs a batch at a fixed interval, reads data from the batch, and exports the read result. The following is an example:

```
-------------------------------------------
Time: 1535423650000 ms
-------------------------------------------
```

```
(hello,30)
(world.,30)
```

Step 11 After verifying that the job can run locally without error, remove the comment tag from the <scope>provided</scope> row in **pom.xml** to prevent the Spark dependency from being packaged. Then stop the data upload program.

----End

4.7 Using a DIS Flink Connector to Upload and Download Data

4.7.1 Introducing DIS Flink Connectors

A DIS Flink Connector is a software development kit (SDK) provided by DIS to create a DStream that uses the DIS as a data source and connect it to Flink.

4.7.2 Preparing a DIS Flink Connector Environment

Preparing a DIS Application Development Environment

Step 1 Set up DIS application development environments by following the procedure in **Step 1: Creating a DIS Stream**.

Step 2 Install Maven and configure a local repository address.

Step 3 Install scala-sdk.

----End

Configuring the DIS Flink Connector Dependency

The DIS Flink Connector dependency can be introduced in the project through the following configuration:

```
<dependency>
   <groupId>com.cloud.dis</groupId>
   <artifactId>cloud-dis-flink-connector_2.11</artifactId>
   <version>1.0.5</version>
   <scope>compile</scope>
</dependency>
```

Checking Authentication Information

- AK/SK file

  Access Key ID/Secret Access Key (AK/SK) files are created by the Identity and Access Management (IAM) service to authenticate calls to application programming interfaces (APIs) on the cloud. To obtain AK/SK, choose **My Credentials > Access Keys**.

- Project ID

  A project is a group of tenant resources. A tenant can have multiple projects, one for each region. Each region has a unique Project ID. To view the project IDs of different regions, choose .

4.7.3 Customizing a Flink Streaming Job

Obtaining a DIS Flink Connector Demo

Step 1 Obtain the **dis-flink-connector-*X.X.X*.zip** package from **https://dis-publish.obs-website.cn-north-1.myhuaweicloud.com/**. Decompress the **dis-flink-connector-*X.X.X*.zip** package to obtain the following directory that contains a Maven project sample:

- **huaweicloud-dis-flink-connector-demo** which contains a sample Maven project

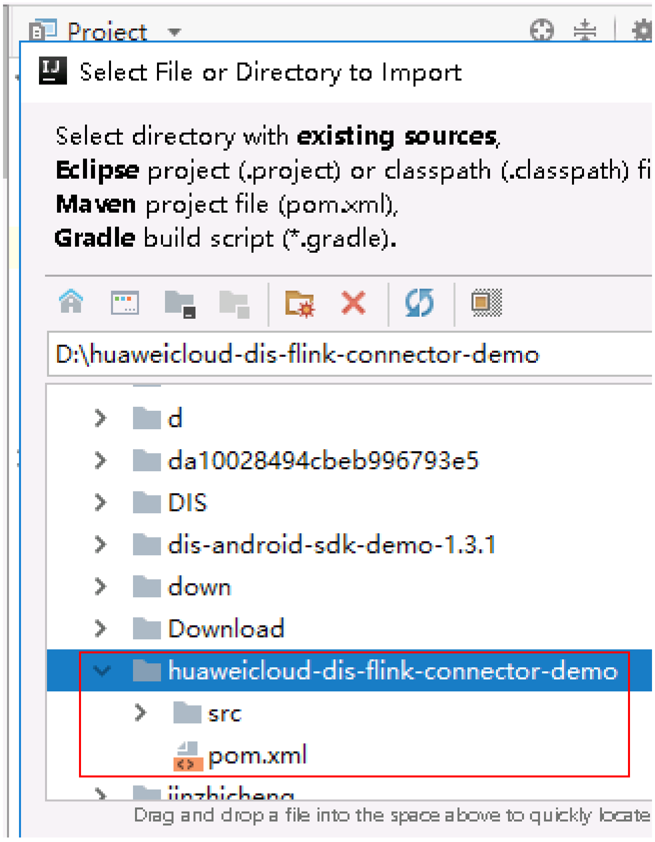
----End

Importing the Demo Project on Intellij IDEA

The following uses the IntelliJ IDEA community version as an example to describe how to compile a Flink job. Make sure that the following components have been configured on IDEA.

- JDK 1.8+
- Scala-sdk-2.11
- Maven 3.3.*

Step 1 Open IntelliJIDEA and choose **File** > **New** > **Project from Existing Sources...**. Select the **huaweicloud-dis-flink-connector-demo** directory and click **OK**.



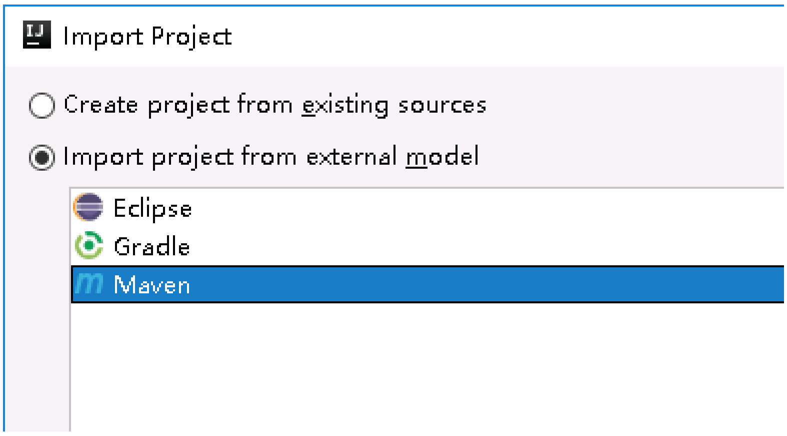
Select File or Directory to Import

Select directory with **existing sources**,
Eclipse project (.project) or classpath (.classpath) fi
Maven project file (pom.xml),
Gradle build script (*.gradle).

D:\huaweicloud-dis-flink-connector-demo

Project
d
da10028494cbeb996793e5
DIS
dis-android-sdk-demo-1.3.1
down
Download
huaweicloud-dis-flink-connector-demo
src
pom.xml
jinzhicheng
Drag and drop a file into the space above to quickly locate

Step 2 Select a Maven project, retain the default settings, and click **Next** until the **New Project** dialog box is displayed.

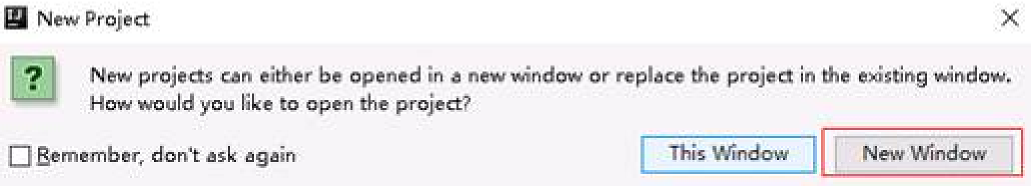


Import Project

○ Create project from existing sources

● Import project from external model

Step 3 Click **New Window** to open the project in a new window.

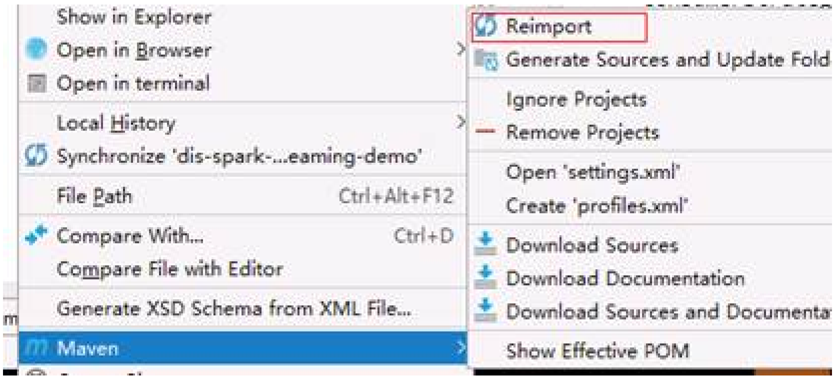


New Project

New projects can either be opened in a new window or replace the project in the existing window.
How would you like to open the project?

□ Remember, don't ask again

Step 4 Right-click **pom.xml** and choose **Maven** > **Reimport** from the shortcut menu to import the maven dependency library again.



----End

Verifying the Flink Streaming Source Job

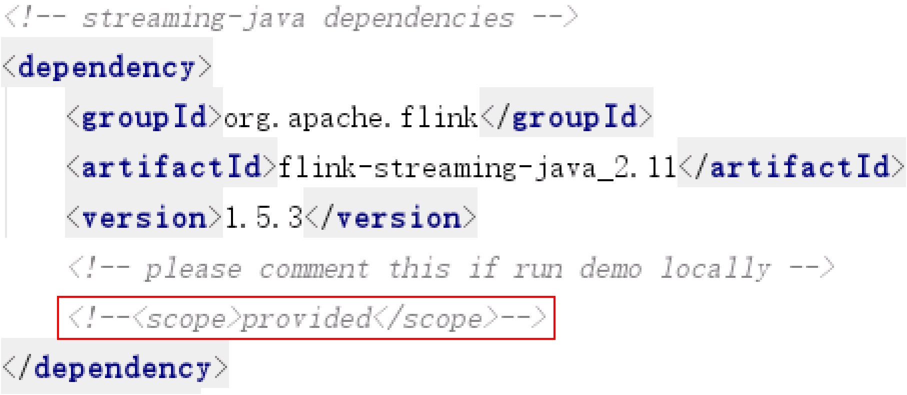
This section describes how to test a Flink Streaming job in the local IDE to understand the basic usage of the SDK. In a real-world scenario, the Flink Streaming job needs to run on a Flink cluster. After the test is complete, you can create clusters (such as MRS clusters) and submit a job for verification.

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click in the upper left corner of the page and select a region and project.

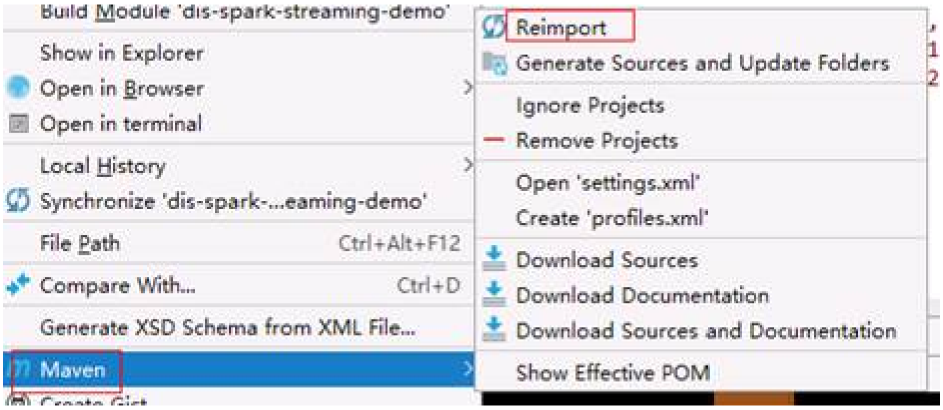
Step 3 Create a DIS stream by referring to **Step 1: Creating a DIS Stream** and continuously upload data to the newly created DIS stream. In this example, the content to be uploaded is **hello world**.

Step 4 Open the **pom.xml** file, press **Ctrl+/** to comment out the **<scope>provided</scope>** row, and save the setting.

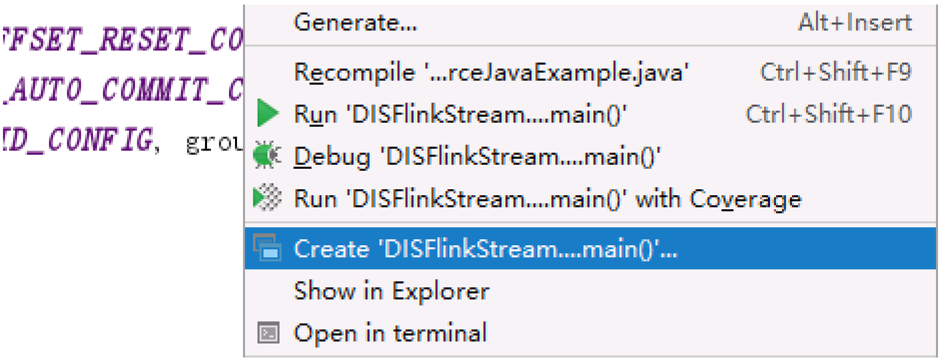


```
<!-- streaming-java dependencies -->
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-streaming-java_2.11</artifactId>
    <version>1.5.3</version>
    <!-- please comment this if run demo locally -->
    <!--<scope>provided</scope>-->
</dependency>
```

Step 5 Right-click **pom.xml** and choose **Maven** > **Reimport** from the shortcut menu to import the dependency package again.



Step 6 Right-click in the **DISFlinkStreamingSourceJavaExample** file and choose **Create 'DISFlinkStreamingSourceJavaExample'** from the shortcut menu.


FSET_RESET_CO
AUTO_COMMIT_C
D_CONFIG, grou
Generate... Alt+Insert
Recompile '...rceJavaExample.java' Ctrl+Shift+F9
Run 'DISFlinkStream....main()' Ctrl+Shift+F10
Debug 'DISFlinkStream....main()'
Run 'DISFlinkStream....main()' with Coverage
Create 'DISFlinkStream....main()'...
Show in Explorer
Open in terminal

Step 7 On the configuration page that is displayed, set **Program arguments** in the following format:

```
DIS gateway address + Region name + AK + SK + Project ID + Stream name + Start position + Consumer ID

https://dis.${region}.myhuaweicloud.com ${region} YOU_AK YOU_SK YOU_PROJECTID YOU_STREAM_NAME
latest GROUP_ID
```

The parameter sequence and meaning are available in the sample code. For details, see the following information:

```
//DIS endpoint.
        String endpoint;
//ID of the region where DIS resides.
        String region;
//AK of the user.
        String ak;
//SK of the user.
        String sk;
//Project ID of the user.
        String projectId;
//DIS stream name.
        String streamName;
//Consumption policy. This policy is used only when the partition has no checkpoint or the checkpoint has
expired. If a valid checkpoint exists, the consumption continues from this checkpoint.
//When the policy is set to LATEST, the consumption starts from the latest data. This policy will ignore the
existing data in the stream.
//When the policy is set to Earliest, the consumption starts from the earliest data. This policy will obtain all
valid data in the stream.
        String startingOffsets;
//Consumer group ID. Different clients in the same consumer group can consume the same stream at the
same time.
        String groupId;
```

NOTE

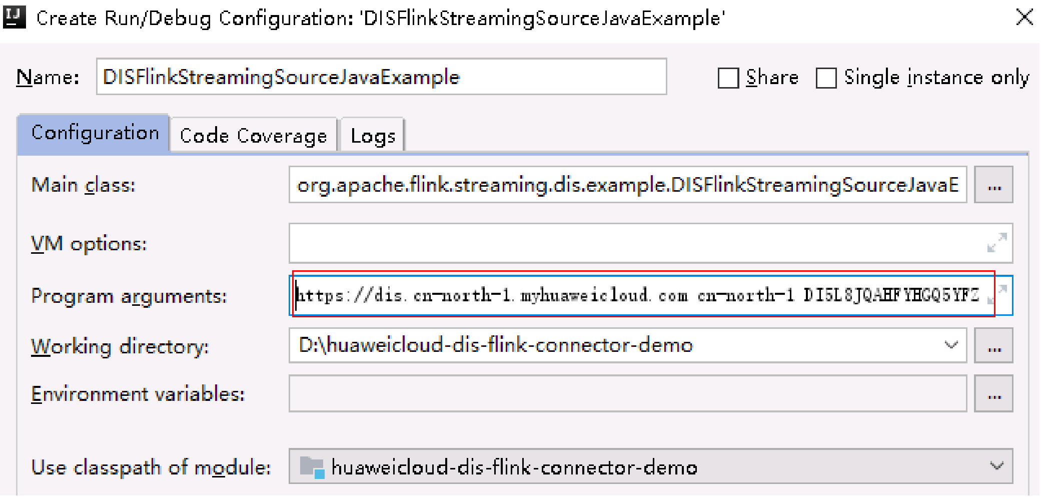
- A checkpoint must be specified or a consumption point is automatically marked as follows:

  disConfig.put(DisConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "true");

  disConfig.put(DisConsumerConfig.AUTO_COMMIT_INTERVAL_MS_CONFIG, "1000");

  disConfig.put(DisConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "LATEST"); //LATEST indicates that the consumption starts from the latest data.
- If none of them is set, the consumption starts from the latest by default.

The final configurations of the IDEA are shown in the following figure. After confirming that all configurations are correct, click **OK** to close the window.



Create Run/Debug Configuration: 'DISFlinkStreamingSourceJavaExample'

Name: DISFlinkStreamingSourceJavaExample   □ Share   □ Single instance only

Main class: org.apache.flink.streaming.dis.example.DISFlinkStreamingSourceJavaE

VM options:

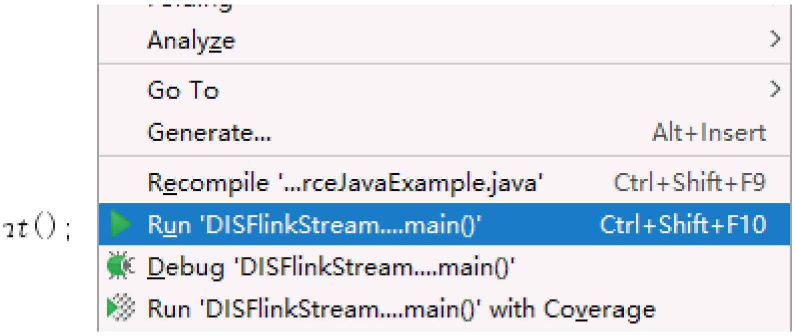
Program arguments: https://dis.cn-north-1.myhuaweicloud.com cn-north-1 DI5L8JQAHFYHGQ5YFZ

Working directory: D:\huaweicloud-dis-flink-connector-demo

Environment variables:

Use classpath of module: huaweicloud-dis-flink-connector-demo

Step 8 Right-click in the **DISFlinkStreamingSourceJavaExample** file and choose **Run 'DISFlinkStreamingSourceJavaExample'** from the shortcut menu to start the job.


nt();
Analyze
Go To
Generate... Alt+Insert
Recompile '...rceJavaExample.java' Ctrl+Shift+F9
Run 'DISFlinkStream....main()' Ctrl+Shift+F10
Debug 'DISFlinkStream....main()'
Run 'DISFlinkStream....main()' with Coverage

Step 9 If no error occurs, data is read from DIS and output to the console. The following is an example:

```
2> hello world
2> hello world
2> hello world
```

Step 10 After verifying that the job can run locally without error, remove the comment tag from the **<scope>provided</scope>** row in **pom.xml** to prevent the Flink dependency from being packaged. Then stop the data upload program.

----End

Verifying the Flink Streaming Sink Job

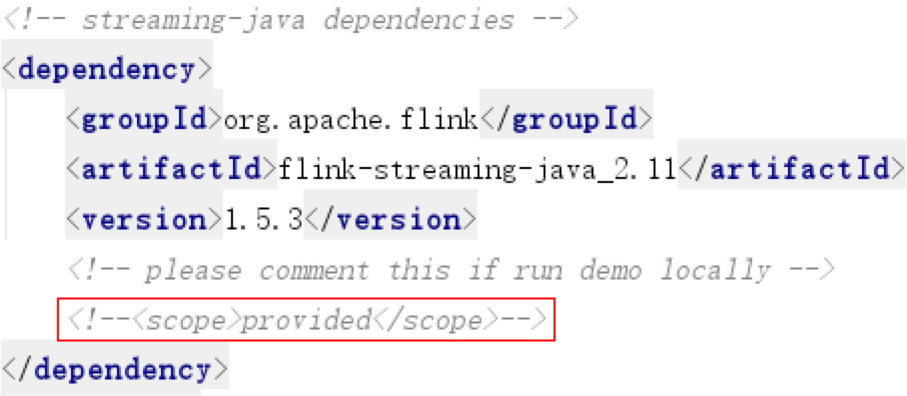
This section describes how to test a Flink job in the local IDE to understand the basic usage of the SDK. In a real-world scenario, the Flink job needs to run on a Flink cluster. After the test is complete, you can create clusters (such as MRS clusters) and submit a job for verification.

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click in the upper left corner of the page and select a region and project.

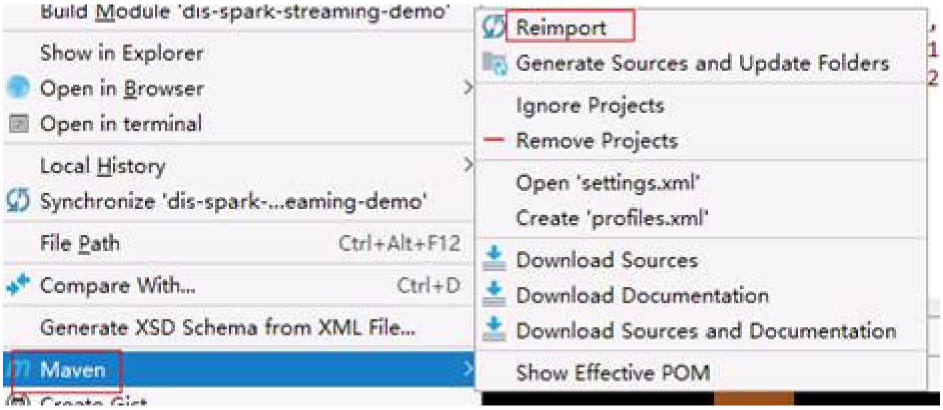
Step 3 Enable DIS by referring to **Step 1: Creating a DIS Stream**.

Step 4 Open the **pom.xml** file, press **Ctrl+/** to comment out the **<scope>provided</scope>** row, and save the setting.

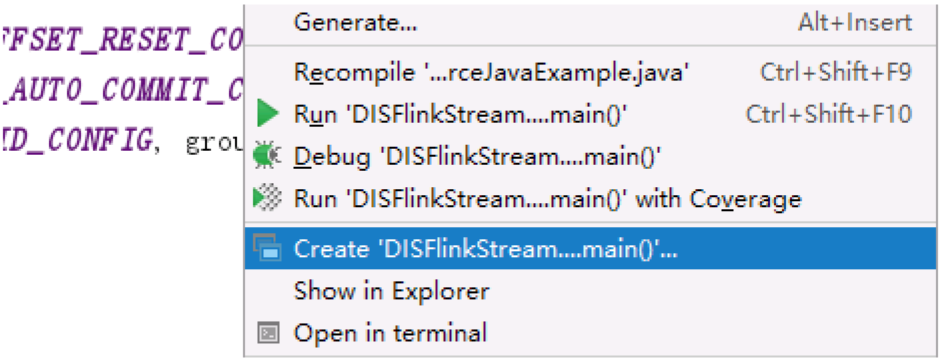


```
<!-- streaming-java dependencies -->
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-streaming-java_2.11</artifactId>
    <version>1.5.3</version>
    <!-- please comment this if run demo locally -->
    <!--<scope>provided</scope>-->
</dependency>
```

Step 5 Right-click **pom.xml** and choose **Maven** > **Reimport** from the shortcut menu to import the dependency package again.



Step 6 Right-click in the **DISFlinkStreamingSinkJavaExample** file and choose **Create 'DISFlinkStreamingSinkJavaExample'** from the shortcut menu.


FSET_RESET_CO
AUTO_COMMIT_C
D_CONFIG, grou
Generate... Alt+Insert
Recompile '...rceJavaExample.java' Ctrl+Shift+F9
Run 'DISFlinkStream....main()' Ctrl+Shift+F10
Debug 'DISFlinkStream....main()'
Run 'DISFlinkStream....main()' with Coverage
Create 'DISFlinkStream....main()'...
Show in Explorer
Open in terminal

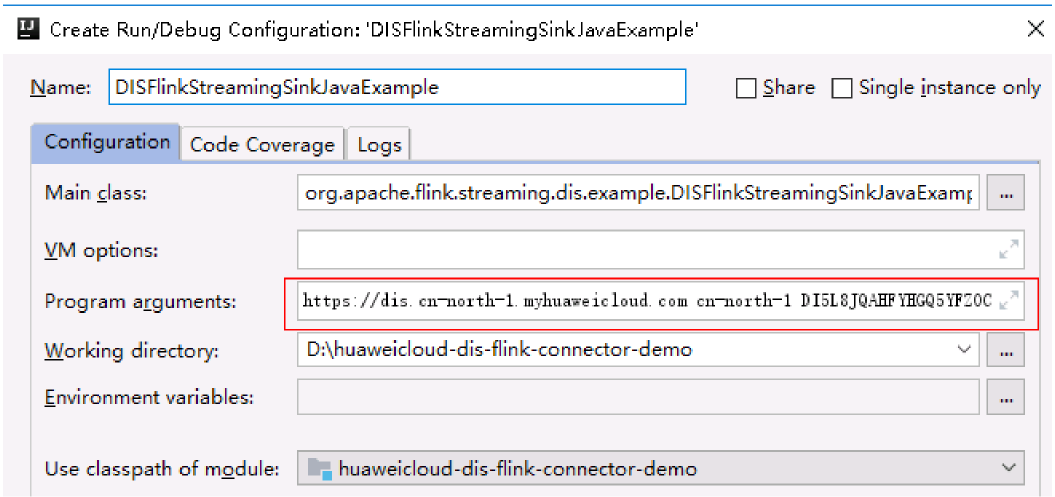
Step 7 On the configuration page that is displayed, set **Program arguments** in the following format:

```
DIS gateway address + Region name + AK + SK + Project ID + Stream name
https://dis.${region}.myhuaweicloud.com ${region} YOU_AK YOU_SK YOU_PROJECTID YOU_STREAM_NAME
```

The parameter sequence and meaning are available in the sample code. For details, see the following information:

```
//DIS endpoint.
        String endpoint;
//ID of the region where DIS resides.
        String region;
//AK of the user.
        String ak;
//SK of the user.
        String sk;
//Project ID of the user.
        String projectId;
//DIS stream name.
        String streamName;
```

The final configurations of the IDEA are shown in the following figure. After confirming that all configurations are correct, click **OK** to close the window.



Create Run/Debug Configuration: 'DISFlinkStreamingSinkJavaExample'

Name: DISFlinkStreamingSinkJavaExample □ Share □ Single instance only

Main class: org.apache.flink.streaming.dis.example.DISFlinkStreamingSinkJavaExamp

VM options:

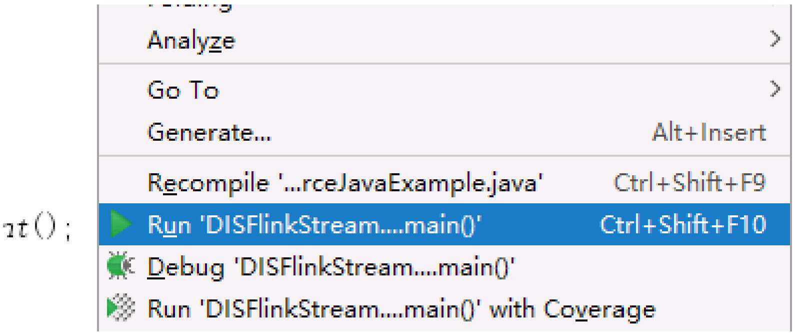
Program arguments: https://dis.cn-north-1.myhuaweicloud.com cn-north-1 DI5L8JQAHFYHGQ5YFZ0C

Working directory: D:\huaweicloud-dis-flink-connector-demo

Environment variables:

Use classpath of module: huaweicloud-dis-flink-connector-demo

Step 8 Right-click in the **DISFlinkStreamingSinkJavaExample** file and choose **Run 'DISFlinkStreamingSinkJavaExample'** from the shortcut menu to start the job.


Analyze
Go To
Generate... Alt+Insert
Recompile '...rceJavaExample.java' Ctrl+Shift+F9
Run 'DISFlinkStream....main()' Ctrl+Shift+F10
Debug 'DISFlinkStream....main()'
Run 'DISFlinkStream....main()' with Coverage

Step 9 Check whether the data is successfully uploaded on the stream monitoring page of the DIS console.

Step 10 After verifying that the job can run locally without error, remove the comment tag from the <scope>provided</scope> row in **pom.xml** to prevent the Flink dependency from being packaged. Then stop the data upload program.

----End

5 Managing a Dump Task

5.1 Creating a Dump Task

If a dump task is created for a DIS stream, data sent to the DIS stream can be automatically dumped to the selected target specified in the dump task.

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click 📍 in the upper left corner of the page and select a region and project.

Step 3 In the left navigation pane, choose .

Step 4 Click the name of a stream that you want to view. On the displayed page, click the **Dump Management** tab. Alternatively, in the **Operation** column of a stream that you want to view, click **More** and choose **View Dump Task** from the drop-down list.

Step 5 Click **Add Dump Task**. On the **Add Dump Task** page, configure dump parameters. For details about the dump task parameters, see .

NOTE

A maximum of five dump tasks can be created for each stream.

Step 6 Click **Create Now**.

Step 7 In the **Operation** column of the corresponding **Task Name**, click **More** > **View Dump Log** to view the dump task details of the stream. **Table 5-1** describes the dump log parameters.

Table 5-1 Dump log parameters

Parameter	Description
Start Time	Time when the dump log is created. Format: YYYY/MM/dd HH:mm:ss GTM ● YYYY: year. ● MM: month. ● dd: date. ● HH: hour. ● mm: minute. ● ss: second. ● GMT: time zone.
End Time	Time when you finish creating the dump log. Format: YYYY/MM/dd HH:mm:ss GTM ● YYYY: year. ● MM: month. ● dd: date. ● HH: hour. ● mm: minute. ● ss: second. ● GMT: time zone.
Status	Dump status. ● Succeeded ● Failed ● Abnormal
Dump File Name	Name of the file that is dumped to the target service. The user records read from the stream are written into the file and then dumped to the target service (such as OBS) in the file format.
Records	Number of the records uploaded between the time when you start to create a dump log to the time when you finish creating it.
Data Amount (bytes)	Amount of the data uploaded between the time when you start to create the dump log to the time when you finish creating it. Unit: byte

Parameter	Description
Operation	Dump failure details. ● If **Status** is **Succeeded**, the column is not operable. ● If **Status** is **Failed**, click **View Details** to view dump details. ● If **Status** is **Abnormal**, click **View Details** to view dump details.

----End

Modifying and Enabling Dump Tasks

After creating a stream and adding a dump task successfully, you can modify the attributes of the created stream.

Step 1 Log in to the DIS console.

Step 2 Click 📍 in the upper left corner of the page and select a region and project.

Step 3 In the navigation tree, choose **Stream Management**.

Step 4 Click the name of a stream that you want to view. On the displayed page, click the **Dump Management** tab. Alternatively, in the **Operation** column of a stream that you want to view, click **More** and choose **View Dump Task** from the drop-down list.

Step 5 In the **Operation** column of the stream for which a dump task has been added, perform the following operations:

1. Choose **More** > **Modify** to modify the dump task.
2. Choose **More** > **Start** to start the dump task.
3. Choose **More** > **Pause** to pause the dump task.

----End

5.2 Dumping Data to OBS

Dumping the JSON, BLOB, and CSV Data to the Text Data

Table 5-2 Parameters for configuring a Text dump file

Parameter	Description	Value
Task Name	Name of the dump task. The names of dump tasks created for the same stream must be unique. A dump task name is 1 to 64 characters long. Only letters, digits, hyphens (-), and underscores (_) are allowed.	-
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 in OBS.	-
File Directory	Directory created in OBS to store files from the DIS stream. This directory name is 0 to 50 characters long. By default, this parameter is left unspecified.	-
Time Directory Format	Data will be saved according to the time format in the file directory of the OBS bucket. For example, if the time directory is accurate to day, the save path will be in the format of bucket name/file directory/year/month/day. Possible values are as follows: ● N/A: If this field is left unspecified, 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 are as follows: ● 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 read. ● **Earliest**: Minimum offset, indicating that the earliest data will be read.	Latest
Dump Interval (s)	Interval at which data from the DIS stream will be imported into dump destination, such as OBS, MRS, DLI, and DWS. If no data was pushed to the DIS stream during the time specified here, the dump file will not be generated. Value range: 30s to 900s Unit: second Default value: 300s	-

Dumping the JSON Data to the CSV Data

Table 5-3 Parameters for configuring a CSV dump file

Parameter	Description	Value
Task Name	Name of the dump task. The names of dump tasks created for the same stream must be unique. A dump task name is 1 to 64 characters long. Only letters, digits, hyphens (-), and underscores (_) are allowed.	-
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 in OBS.	-

Parameter	Description	Value
File Directory	Directory created in OBS to store files from the DIS stream. This directory name is 0 to 50 characters long. By default, this parameter is left unspecified.	-
Time Directory Format	Data will be saved according to the time format in the file directory of the OBS bucket. For example, if the time directory is accurate to day, the save path will be in the format of bucket name/file directory/year/month/day. Possible values are as follows: ● N/A: If this field is left unspecified, 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.	-
Offset	● **Latest**: Maximum offset, indicating that the latest data will be read. ● **Earliest**: Minimum offset, indicating that the earliest data will be read.	Latest
Dump Interval (s)	User-defined interval at which data is imported from the current DIS stream into the target place. 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: 300s	-

Dumping the JSON and CSV Data to the Parquet Data

NOTE

Table 5-4 lists the differentiated parameters that need to be set when the source data type is JSON or CSV, the dump destination is OBS, and the dump file format is Parquet. For details about how to configure other common parameters, see **Table 5-2**.

Table 5-4 Parameters for configuring a Parquet dump file

Parameter	Description	Value
Source Data Schema	JSON or CSV data example, used to describe the JSON or CSV data format. DIS can generate an Avro schema based on the JSON or CSV data sample and convert the uploaded JSON or CSV data to the Parquet format.	-

Parameter	Description	Value
File Directory	Directory created in OBS to store files from the DIS stream. This directory name is 0 to 100 characters long. By default, this parameter is left unspecified. **NOTE** When the source data type is JSON, EL and built-in functions are supported.	● **EL example** Source data: {"name":"Andy","city":"", "time":1556323141582} The dump file directory is configured as follows: dis/basePath/app_key_p=${name} The final directory structure is as follows: {$Bucket name}/dis/basePath/app_key_p=Andy ● **Example of built-in functions** Source data: {"name":"Andy","city":"", "time":1556323141582} The dump file directory is configured as follows: dis/basePath/date_p=toDate(${time}, "yyMMdd") The final directory structure is as follows: {$Bucket name}/dis-basePath/date_p=20190427 **Supported built-in functions** toDate(timestamp, format): Convert the time stamp to a specified time format. For example: toDate (1556323141582, 'yymmdd')

Parameter	Description	Value
Custom Time Directory	You can click or to disable or enable the custom time directory. ● If the custom timestamp is disabled, the directory where the object file written to OBS resides is named after the creation time of the dump file. For example, if a dump file was created on October 16, 2018 and the time directory format is accurate to day, the file will be saved in *OBS bucket name*/*dump file directory*/**2018**/**10**/**16**. ● If the custom timestamp is disabled, the directory where the object file written to OBS resides is named in the time format specified in the source data. For example, if a dump file was created on October 16, 2018, the time directory format is accurate to day, and the time format specified in the data source is 2017/09/08 11:01:01, the file will be saved in *OBS bucket name*/*dump file directory*/**2017**/**09**/**08**. The storage directory is defined based on the time format defined in the source data instead of the time when the dump file is created.	● Example 1: Dump simple JSON data. Source data: { "id":"1", "date":"2018/10/16 11:00:05"} The configuration is as follows: Set the timestamp attribute name to **date**, data type to **String**, and timestamp format to **yyyy/MM/dd HH:mm:ss** based on the source data type of the data to be dumped. After the data is dumped successfully, the storage directory structure depends on the source data timestamp and the time directory format. In this example, the time directory format is accurate to day. Therefore, the final data storage directory is *OBS bucket name*/*dump file directory*/**2018**/**10**/**16**. ● Example 2: Dump multiply layers of nested JSON data. Source data: { "id":"1", "detail": { "detID":"05790110000000000103#567fd3cb13a4493eaa43076953253eed", "endTime":"2018/10/07 13:26:35" }} The configuration is as follows: Set the timestamp attribute name to **detail.endTime**, data type to **String**, and timestamp format to **yyyy/MM/dd HH:mm:ss**
Source Data Timestamp	● Attribute name of the timestamp. **NOTE** Enter the field name corresponding to the timestamp defined in the source data to be uploaded. ● Timestamp format. Possible values can be: yyyy/MM/dd HH:mm:ss MM/dd/yyyy HH:mm:ss dd/MM/yyyy HH:mm:ss yyyy-MM-dd HH:mm:ss MM-dd-yyyy HH:mm:ss	

Parameter	Description	Value
	dd-MM-yyyy HH:mm:ss ● Data type. Possible values can be: – String – Timestamp **NOTE** If the type of the source data to be uploaded is **Timestamp**, the value must be accurate to milliseconds.	based on the source data type of the data to be dumped. After the data is dumped successfully, the storage directory structure depends on the source data timestamp and the time directory format. In this example, the time directory format is accurate to day. Therefore, the final data storage directory is *OBS bucket name*/*dump file directory*/**2018**/**10**/**07**. ● Example 3: Dump CSV data. Source data: a,2010-10-12 11:00:00,b,2011-10-12 11:00:10 The configuration is as follows: Select timestamp **2010-10-12 11:00:00** based on the source data to be dumped. After data is converted to the Parquet format, the corresponding attribute field name is **field_1**. When creating a dump task, set the timestamp attribute to **field_1**, data type to **String**, and timestamp format to **yyyy/MM/dd HH:mm:ss**. After the data is dumped successfully, the storage directory structure depends on the source data timestamp and the time directory format. In this example, the time directory format is accurate to day. Therefore, the final data storage directory is *OBS*

Parameter	Description	Value
		bucket name/*dump file directory*/**2010**/**10**/**12**.

Dumping the JSON and CSV Data to the CarbonData Data

NOTE

Table 5-5 lists the differentiated parameters that need to be set when the source data type is JSON or CSV, the dump destination is OBS, and the dump file format is CarbonData. For details about how to configure other common parameters, see **Table 5-2**.

Table 5-5 Parameters for configuring a CarbonData dump file

Parameter	Description	Value
Source Data Schema	JSON or CSV data sample, used to describe the JSON or CSV data format. DIS can generate an Avro schema based on the JSON or CSV data sample and convert the uploaded JSON or CSV data to the CarbonData format.	-

Parameter	Description	Value
CarbonData Retrieval Attribute	Attribute of the carbon table, used to create a carbon writer. The following keys are supported: ● **table_blocksize**: Size of a table block. The value ranges from 1 MB to 2048 MB. The default value is 1024 MB. ● **table_blocklet_size**: Size of the blocklet in a file. The default value is 64 MB. ● **local_dictionary_enable**: Possible values can be **true** or **false**. The default value is **false**. ● **sort_columns**: Specifies the index column. Multi-level index columns are separated by commas (,). ● **sort_scope**: Specifies the scope where data is sorted during loading. Currently, the following types are supported: – **local_sort**: Default value, indicating that data is sorted in a node. – **no_sort**: Data is not sorted. It is used when data needs to be saved to a database quickly. After the data is saved to the database, you can use the **Compaction** command to create an index when the system is idle. – **batch_sort**: A CarbonData file is generated after the memory is sorted in a node and no full sorting is performed on the node. This configuration improves the loading speed, but the query performance is inferior to that of **LOCAL_SORT**.	-

5.3 Dumping Data to DLI

Source Data Type: JSON and CSV

Table 5-6 Dump parameters

Parameter	Description	Value
Task Name	Name of the dump task. The names of dump tasks created for the same stream must be unique. A dump task name is 1 to 64 characters long. Only letters, digits, hyphens (-), and underscores (_) are allowed.	-
DLI Database	Click **Select**. In the **Select DLI Database** dialog box, select a database. You can only select but not enter a value in this field.	-
DLI Table	Click **Select**. In the **Select DLI Table** dialog box, select a data table. Only tables in DLI are supported. Before selecting a DLI table, ensure that you have the permission to insert data into the table. You can only select but not enter a value in this field.	This parameter is available only after you select a DLI database.
Offset	● **Latest**: Maximum offset, indicating that the latest data will be read. ● **Earliest**: Minimum offset, indicating that the earliest data will be read.	Latest
Dump Interval (s)	Interval at which data from the DIS stream will be imported into dump destination, such as OBS, MRS, DLI, and DWS. If no data was pushed to the DIS stream during the time specified here, the dump file will not be generated. Value range: 30s to 900s Unit: second Default value: 300s	-
Temporary Bucket	OBS bucket in which a directory is created for temporarily storing user data. The data in the directory is deleted after being dumped to a specific destination.	-

Parameter	Description	Value
Temporary Directory	Directory in the chosen **Temporary Bucket** for temporarily storing data. The data in the directory is deleted after being dumped to a specific destination. If this field is left blank, the data is stored directly to the **Temporary Bucket**.	-

5.4 Dumping Data to DWS

Source Data Type: JSON and CSV

Table 5-7 Dump parameters

Parameter	Description	Value
Task Name	Name of the dump task. The names of dump tasks created for the same stream must be unique. A dump task name is 1 to 64 characters long. Only letters, digits, hyphens (-), and underscores (_) are allowed.	-
DWS Cluster	Name of the DWS cluster that stores the data in the stream. Click **Select**. In the **Select DWS Cluster** dialog box, select a DWS cluster. You can only select but not enter a value in this field.	-
DWS Database	Name of the DWS database that stores the data in the stream. The value must be manually entered and cannot be left blank.	-
Database Schema	A database contains one or more named schemas, and schemas contain tables. Schemas also contain other named objects, including data types, functions, and operators. The same object name can be used in different modes without causing conflicts.	-
DWS Table	Name of the DWS table that stores the data in the stream.	-

Parameter	Description	Value
Delimiter	Delimiter used to separate the columns in the DWS tables. This parameter cannot be left blank.	-
Offset	● **Latest**: Maximum offset, indicating that the latest data will be read. ● **Earliest**: Minimum offset, indicating that the earliest data will be read.	Latest
Dump Interval (s)	Interval at which data from the DIS stream will be imported into dump destination, such as OBS, MRS, DLI, and DWS. If no data was pushed to the DIS stream during the time specified here, the dump file will not be generated. Value range: 30s to 900s Unit: second Default value: 300s	-
Username	Username for logging in to the DWS cluster.	-
Password	Password for logging in to the DWS cluster.	-
KMS Key	Database encryption key of the cluster	-
Temporary Bucket	OBS bucket in which a directory is created for temporarily storing user data. The data in the directory is deleted after being dumped to a specific destination.	-
Temporary Directory	Directory in the chosen **Temporary Bucket** for temporarily storing data. The data in the directory is deleted after being dumped to a specific destination. If this field is left blank, the data is stored directly to the **Temporary Bucket**.	-

Parameter	Description	Value
Fault Processing	You can click or to disable or enable Fault Processing. ● fill_missing_fields Specifies whether to generate an error message when the last field in one row of data in the source file is missing during data import. Value: **true/on** or **false/off** Default value: **false/off** – **true/on**: The value of the last field is set to **NULL** and no error message is reported. – **false/off**: The following error message is reported: ● ignore_extra_data Specifies whether to ignore the extra columns when the number of fields in the data source file is greater than the number of columns defined in the external table. This parameter is used only during data import. Value: **true/on** or **false/off** Default value: **false/off** – **true/on**: The extra columns at the end of a row are ignored. – **false/off**: The following error message is reported: **NOTE** If a linefeed at the end of a row is missing so that the row and another row are integrated into one, data in another row is ignored after the parameter is set to **true**. ● compatible_illegal_chars Specifies whether to tolerate invalid characters during data import. This parameter is valid only for READ ONLY foreign tables. Value: **true/on** or **false/off** Default value: **false/off** – **true/on**: Invalid characters are converted into valid ones before being imported to the database. No error message is reported and data import is not interrupted. – **false/off**: Data import is interrupted.	-

Parameter	Description	Value
	NOTICE In the Windows operating system, if OBS reads data files in text format, 0x1A is used as the EOF symbol to end data reading. As a result, a parsing error occurs. This is a restriction on the Windows platform. OBS can read BINARY data files on Linux only. **NOTE** ● The rules for converting invalid characters are as follows: 1. \0 is converted to a space. 2. Others invalid characters are converted into question marks. 3. When **compatible_illegal_chars** is set to **true/on**, after invalid characters such as **NULL**, **DELIMITER**, **QUOTE**, and **ESCAPE** are converted to spaces or question marks, an error message stating "illegal chars conversion may confuse COPY escape 0x20" will be displayed to remind you of possible parameter confusion caused by the conversion. ● **PER NODE REJECT LIMIT 'value'** Specify the number of data format errors allowed in each DN instance during data import. If the number of errors in a DN instance is greater than the preset value, the import fails and an error message is displayed. Value range: integer, unlimited. The default value is 0, indicating that error information is returned immediately. **NOTE** This parameter specifies the error tolerance of a single node. Data format errors refer to missing or redundant field values, data type errors, or coding errors. If a non-data format error occurs, the entire data import fails.	

5.5 Dumping Data to MRS

Prerequisites

DIS cannot dump data to MRS 3.x or later versions. Kerberos authentication must be disabled for the MRS cluster to which data is to be dumped.

Source Data Type: JSON, BLOB, and CSV; Dump File Format: Text

Table 5-8 Parameters for configuring a Text dump file

Parameter	Description	Value
Task Name	Name of the dump task. The names of dump tasks created for the same stream must be unique. A dump task name is 1 to 64 characters long. Only letters, digits, hyphens (-), and underscores (_) are allowed.	-
MRS Cluster	Click **Select**. In the **Select MRS Cluster** dialog box, select an MRS cluster. Data is dumped only to an MRS cluster that is not authenticated by Kerberos. You can only select but not enter a value in this field.	-
HDFS Path	Click **Select**. In the **Select HDFS Path** dialog box, select an HDFS path. You can only select but not enter a value in this field.	This parameter is available only after you select an MRS cluster.
File Directory	Directory created in MRS to store files from the DIS stream. This directory name is 0 to 50 characters long. By default, this parameter is left unspecified.	-
Offset	● **Latest**: Maximum offset, indicating that the latest data will be read. ● **Earliest**: Minimum offset, indicating that the earliest data will be read.	Latest
Dump Interval (s)	Interval at which data from the DIS stream will be imported into dump destination, such as OBS, MRS, DLI, and DWS. If no data was pushed to the DIS stream during the time specified here, the dump file will not be generated. Value range: 30s to 900s Unit: second Default value: 300s	-
Temporary Bucket	OBS bucket in which a directory is created for temporarily storing user data. The data in the directory is deleted after being dumped to a specific destination.	-

Parameter	Description	Value
Temporary Directory	Directory in the chosen **Temporary Bucket** for temporarily storing data. The data in the directory is deleted after being dumped to a specific destination. If this field is left blank, the data is stored directly to the **Temporary Bucket**.	-

Source Data Type: JSON and CSV; Dump File Format: Parquet

NOTE

Table 5-9 lists the differentiated parameters that need to be set when the source data type is JSON or CSV, the dump destination is MRS, and the dump file format is Parquet. For details about how to configure other common parameters, see **Table 5-8**.

Table 5-9 Parameters for configuring a Parquet dump file

Parameter	Description	Value
Source Data Schema	JSON or CSV data example, used to describe the JSON or CSV data format. DIS can generate an Avro schema based on the JSON or CSV data sample and convert the uploaded JSON or CSV data to the Parquet format.	-

Source Data Type: JSON and CSV; Dump File Format: CarbonData

NOTE

Table 5-10 lists the differentiated parameters that need to be set when the source data type is JSON or CSV, the dump destination is OBS, and the dump file format is CarbonData. For details about how to configure other common parameters, see **Table 5-8**.

Table 5-10 Parameters for configuring a CarbonData dump file

Parameter	Description	Value
Source Data Schema	JSON or CSV data example, used to describe the JSON or CSV data format. DIS can generate an Avro schema based on the JSON or CSV data sample and convert the uploaded JSON or CSV data to the CarbonData format.	-

Parameter	Description	Value
CarbonData Retrieval Attribute	Attribute of the carbon table, used to create a carbon writer. The following keys are supported: ● **table_blocksize**: Size of a table block. The value ranges from 1 MB to 2048 MB. The default value is 1024 MB. ● **table_blocklet_size**: Size of the blocklet in a file. The default value is 64 MB. ● **local_dictionary_enable**: Possible values can be **true** or **false**. The default value is **false**. ● **sort_columns**: Specifies the index column. Multi-level index columns are separated by commas (,). ● **sort_scope**: Specifies the scope where data is sorted during loading. Currently, the following types are supported: – **local_sort**: Default value, indicating that data is sorted in a node. – **no_sort**: Data is not sorted. It is used when data needs to be saved to a database quickly. After the data is saved to the database, you can use the **Compaction** command to create an index when the system is idle. – **batch_sort**: A CarbonData file is generated after the memory is sorted in a node and no full sorting is performed on the node. This configuration improves the loading speed, but the query performance is inferior to that of **LOCAL_SORT**.	-

6 Managing Enterprise Projects

The enterprise project is a cloud resource management mode. Enterprise Management provides comprehensive management for cloud-based resources, personnel, permissions, and finance. Different from common management consoles that feature independent control and configuration of cloud products, the Enterprise Management console is oriented to resource management, helping enterprises manage cloud-based resources, personnel, permissions, and finance in a hierarchical manner, such as management of companies, departments, and projects.

Users who have enabled Enterprise Management can use it to manage cloud service resources on HUAWEI CLOUD.

Binding an Enterprise Project

When creating a DIS stream, you can select an enterprise project for the stream to associate the stream with the enterprise project. For details, see **Step 1: Creating a DIS Stream**. The **Enterprise Project** drop-down list displays the projects you created. In addition, the system has a built-in enterprise project (**default**). If you do not select an enterprise project, the **default** project is used.

During stream creation, if the stream is successfully bound to an enterprise project, the stream is successfully created. If the binding fails, the system sends an alarm and the stream fails to be created.

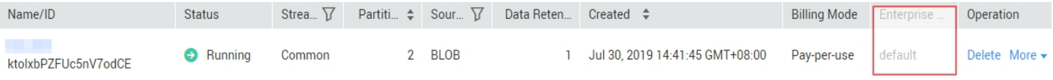
After you delete a DIS stream, the association between the stream and its enterprise project is automatically deleted.

Viewing Enterprise Projects

After a stream is created, you can view the enterprise projects associated with the stream on the stream list and basic stream information page. You can query only the stream resources of the project on which you have the access permission.

In the **Enterprise Project** column of the **Streams** page, view enterprise projects to which streams belong.

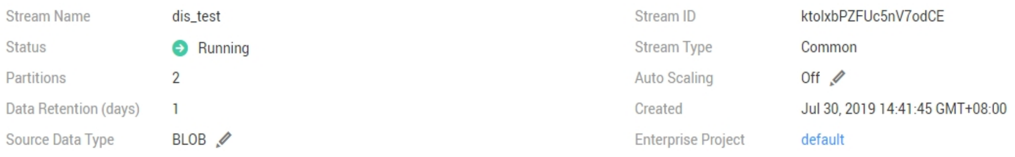
Figure 6-1 Viewing enterprise projects



Name/ID	Status	Strea...	Partiti...	Sour...	Data Reten...	Created	Billing Mode	Enterprise ...	Operation
[redacted] ktolxbPZFUc5nV7odCE	Running	Common	2	BLOB	1	Jul 30, 2019 14:41:45 GMT+08:00	Pay-per-use	default	Delete More

On the **Streams** page, click a stream name. In the stream details area, view the enterprise project associated with the stream. Click the enterprise project name to view and edit it on the Enterprise Management console.

Figure 6-2 Viewing enterprise projects



Stream Name	dis_test	Stream ID	ktolxbPZFUc5nV7odCE
Status	Running	Stream Type	Common
Partitions	2	Auto Scaling	Off
Data Retention (days)	1	Created	Jul 30, 2019 14:41:45 GMT+08:00
Source Data Type	BLOB	Enterprise Project	default

When querying the resource list of a specified project on the Enterprise Management console, you can also query the DIS resources.

Searching for Stream by Enterprise Project

Log in to the DIS console and choose **Stream Management**. In the drop-down list above the streams, select a required project name to view all streams associated with the project.

Migrating Streams into or out of Enterprise Projects

One DIS stream can be associated with only one enterprise project. After a stream is created, you can migrate it from its current enterprise project to another one on the Enterprise Management console, or migrate the stream from another enterprise project to a specified enterprise project. After the migration, the stream is associated with the new enterprise project. The association between them the stream and the original enterprise project is automatically released. For details, see **Resource Management > Managing Enterprise Project Resources** in the *Enterprise Management User Guide*.

7 Notifying Events

7.1 Event Notification Overview

Overview

DIS uses Simple Message Notification (SMN) to send notifications of DIS events. In a subscription, you need to specify one or more event filtering conditions. When an event that matches all filtering conditions occurs, DIS sends a notification based on the subscription. The filter conditions include the **Event Type** (for example, **Management**, **Monitoring**, or **Security**), **Event Level** (for example, **Normal** or **Warning**), and **Event Source** (for example, **Stream**, **Dump task**, or **User**).

Supported Event Types and Events

Events are records of changes in the tenant's stream status. It can be triggered by a user operation (for example, an audit event), or may be caused by a stream status change (for example, a dump task is abnormal or a dump task is recovered). The following tables list the events and event types supported by DIS:

- The following table lists the events whose **Event Source** is **Stream**.

Table 7-1 Events whose **Event Source** is **Stream**

Event Source	Event Level	Event
Stream	Warning	Traffic limited
Stream	Warning	Automatic stream scaling succeeded
Stream	Warning	Automatic stream scaling failed
Stream	Warning	Stream traffic abnormal
Stream	Warning	Stream traffic restored

- The following table lists the events whose **Event Source** is **Event**.

Table 7-2 Events whose **Event Source** is **User**

Event Source	Event Level	Event
User	Warning	Quota insufficient

- The following table lists the events whose **Event Source** is **Dump task**.

Table 7-3 Events whose **Event Source** is **Dump task**

Event Source	Event Level	Event
Dump task	Normal	Dump task restored
Dump task	Warning	Dump task abnormal

7.2 Setting Notification

After enabling Notification for subscribed events, you will receive notifications by email or SMS when management, monitoring, or security events occur in a specific cluster or dump task.

Creating a Subscription

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click **Event Management**.

Step 3 On the **Event Management** page, choose **Subscriptions** > **Create Subscription**.

Step 4 In the **Subscription Settings** area, set basic subscription information and event filtering.

The **Events** area displays the events filtered by the system based on the subscription settings.

Table 7-4 Subscription parameters

Parameter	Description
Notification	Enable or disable event subscription. ● indicates that notification is enabled and ○ indicates that notification is disabled. By default, notification is disabled. After notification is disabled, the system stops sending notifications of subscribed events but does not delete the subscription.

Parameter	Description
Subscription Name	Enter the name of a subscription. ● The name can contain letters (upper or lower case), digits, hyphens (-), and underscores (_) and must start with a letter or digit. ● The name contains 1 to 64 characters.
Subscription Stream	Specifies whether to enable subscription to alarms of a specified stream. ● indicates that subscription to alarms of a specified stream is enabled. ○ indicates that subscription to alarms of a specified channel is disabled. By default, the function is disabled. After this function is enabled, the alarms sent from your specified stream are received, and sent from other streams will not be received.
Subscription Type	Two subscription types are supported: SMN and DIS. **NOTE** ● If **Subscription Type** is set to **SMN**, go to **Step 5**. ● If **Subscription Type** is set to **DIS**, go to **5** to select a stream.

Step 5 Select a message notification topic from the **SMN Topic** drop-down list.

You can perform the following operations to create a message notification topic as required.

1. Click **Create SMN Topic**. The **Topics** page of SMN is displayed. You can click **Create Topic** in the upper right corner to create a topic. For details, see section **Creating a Topic** in the *Simple Message Notification User Guide*.
2. In the **Operation** column of a topic, choose **More** > **Configure Topic Policy** and select **DIS** under **Services that can publish messages to this topic** to enable SMN to publish DIS topics.
3. In the row containing the created topic, click **Add Subscription** to add a subscription to the topic. For details, see **Adding a Subscription** in the *Simple Message Notification User Guide*.

Step 6 Click **OK**.

----End

Modifying the Subscription

Step 1 Choose **Event Management** > **Subscriptions**.

Step 2 In the **Operation** column of a specified subscription name, choose **More** > **Modify**.

Step 3 On the **Subscription Settings** page, set the parameters that you want to modify. For details, see **Step 4** to **Step 6** in section **Creating a Subscription**.

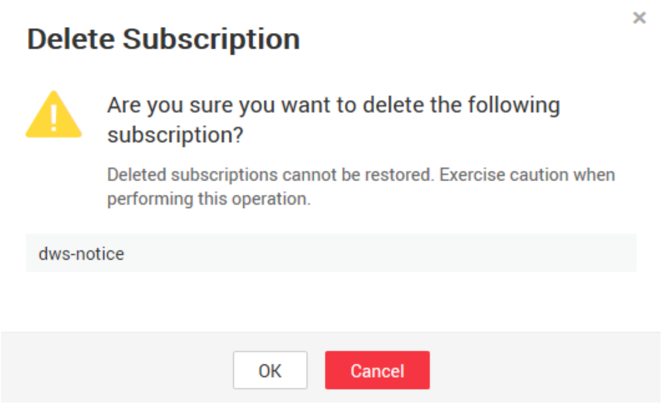
----End

Deleting the Subscription

Step 1 Choose **Event Management** > **Subscriptions**.

Step 2 In the **Operation** column of a specified subscription name, choose **More** > **Delete**. The **Delete Subscription** dialog box is displayed.

Figure 7-1 Confirming deletion



Delete Subscription

Are you sure you want to delete the following subscription?

Deleted subscriptions cannot be restored. Exercise caution when performing this operation.

dws-notice

Step 3 Click **OK**.

----End

7.3 Viewing Events

This section describes how to search for events that occur in a stream or dump task.

Step 1 In the navigation tree, choose **Event Management** > **Events**. All events are displayed by default.

Step 2 Select different filter criteria from the drop-down list in the upper right corner of the event list to search for events. Events can be filtered based on the event level or event source.

- In the **Event Levels** drop-down list, select **All**, **Normal**, or **Warning**
- In the **Event source** drop-down list, select **Event source**. In the search box, enter a stream name or a dump task name, for example, **demo**.

Step 3 Click 🔍. The filtered event query results are displayed.

Step 4 Click ▽ on the right of **Event** and select an event name, for example, **Dump task restored** to filter the corresponding events.

----End

8 Monitoring

8.1 Supported Metrics

Description

This section describes metrics reported by DIS to Cloud Eye as well as their namespaces and dimensions. You can use Cloud Eye to query metric information generated for DIS.

Namespace

SYS.DAYU

Metrics

Table 8-1 lists the DIS metrics.

Table 8-1 DIS metrics

Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
Total Input Traffic	The amount of data uploaded through a stream during a specific period. Unit: byte/s	≥ 0	Stream	1 minute
Total Output Traffic	The amount of data downloaded through a stream during a specific period. Unit: byte/s	≥ 0	Stream	1 minute

Metric Name	Description	Value Range	Monitored Object	Monitoring Period (Raw Data)
Total Input Records	The number of records uploaded through a stream during a specific period. Unit: count/s	≥ 0	Stream	1 minute
Total Output Records	The number of records downloaded through a stream during a specific period. Unit: count/s	≥ 0	Stream	1 minute
Successful Upload Requests	The number of successful requests for uploading data through a stream during a specific period. Unit: count/s	≥ 0	Stream	1 minute
Successful Download Requests	The number of successful requests for downloading data through a stream during a specific period. Unit: count/s	≥ 0	Stream	1 minute
Average Processing Time of Upload Requests	Average upload request delay of a stream during a specific period. Unit: ms	0 to 50 ms	Stream	1 minute
Average Processing Time of Download Requests	Average download request delay of a stream during a specific period. Unit: ms	0 to 50 ms	Stream	1 minute
Throttled Upload Requests	The number of rejected upload requests due to flow control. Unit: count/s	0 to 1	Stream	1 minute
Throttled Download Requests	The number of rejected download requests due to flow control. Unit: count/s	0 to 1	Stream	1 minute

Dimension

Key	Value
stream_id	Real-time data ingestion

8.2 Setting Alarm Rules

Scenario

You can customize the monitored objects and notification policies by setting alarm rules to learn DIS running status in a timely manner.

A DIS alarm rule includes the alarm rule name, monitored object, metric, threshold, monitoring interval, and whether to send a notification. This section describes how to set DIS alarm rules.

Procedure

Step 1 Log in to the management console.

Step 2 Choose **Service List** > **Management & Governance** > **Cloud Eye**.

Step 3 In the left navigation pane, choose **Alarm Management** > **Alarm Rules**. In the right pane, click **Create Alarm Rule**.

Step 4 Set alarm rules for the DIS stream as prompted. Currently, only custom alarm rules are supported.

Step 5 After configuring the required parameters, click **Create**. When an alarm that meets the rule is generated, the system automatically sends a notification.

NOTE

For more information about DIS alarm rules, see the ***Cloud Eye User Guide***.

----End

8.3 Querying Metrics

Scenario

Cloud Eye monitors the running status of the DIS stream. You can obtain the monitoring metrics of CDM on the Cloud Eye management console.

Monitored data requires a period of time for transmission and display. The status of CDM displayed on the Cloud Eye page is the status obtained 5 to 10 minutes before. You can view the monitored data of a newly created DIS stream 5 to 10 minutes later.

Prerequisites

- The DIS stream is running properly.

  NOTE

  Cloud Eye will delete a deleted stream from the monitoring list and will not monitor it any more. However, you need to manually clear its alarm rules.

- Alarm rules have been configured on the Cloud Eye page. For details, see **Setting Alarm Rules**.

Procedure

Step 1 Use the account to log in to the **DIS console**.

Step 2 Click in the upper left corner of the page and select a region and project.

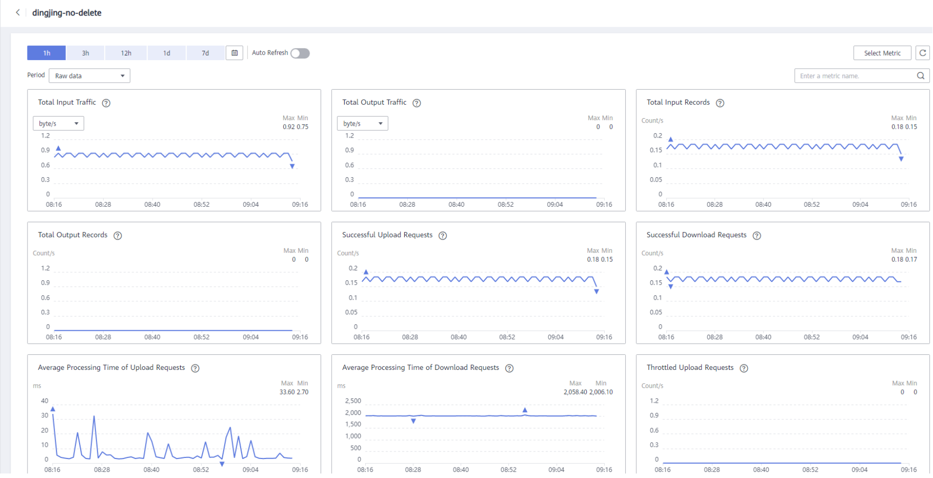
Step 3 In the navigation tree on the left, choose **Stream Management**.

Step 4 In the stream list, click the name of the DIS stream whose monitoring metrics will be viewed. The monitoring page is displayed.

Step 5 On the **Streams** tab page, click **View details** to switch to the Cloud Eye console.

Step 6 On the DIS monitoring page, you can view the graphs of all monitoring metrics.

Figure 8-1 Viewing DIS monitoring metrics



Step 7 Click in the upper right corner of the graphs to zoom out the graphs.

You can view the raw monitoring data curves of different metrics over different periods, such as the last 1 hour, 3 hours, and 12 hours. You can determine whether to enable **Auto Refresh**. Cloud Eye provides an automatic refresh interval of 60 seconds.

----End