DataArts Studio

Best Practices

Issue 01

Date 2022-09-30

HUAWEI TECHNOLOGIES CO., LTD.

Contents

1 Advanced Data Migration Guidance

1.1 Incremental Migration

1.1.1 Incremental File Migration

CDM supports incremental migration of file systems. After full migration is complete, all new files or only specified directories or files can be exported.

Currently, CDM supports the following incremental migration modes:

1. **Exporting the files in a specified directory**
   - Application scenarios: The migration source is a file system (OBS/HDFS/FTP/SFTP). In incremental migration, only the specified files are written to the migration destination. The existing records are not updated or deleted.
   - Key configurations: **File/Path Filter** and Schedule Execution
   - Prerequisites: The source directory or file name contains the time field.
2. **Exporting the files modified after the specified time point**
   - Application scenarios: The migration source is a file system (OBS/HDFS/FTP/SFTP). The specified time point refers to the time when the file is modified. CDM migrates the files modified after the specified time point.
   - Key configurations: **Time Filter** and Schedule Execution
   - Prerequisites: None

NOTE

If you have configured a macro variable of date and time and schedule a CDM job through DataArts Studio DataArts Factory, the system replaces the macro variable of date and time with (*Planned start time of the data development job* – *Offset*) rather than (*Actual start time of the CDM job* – *Offset*).

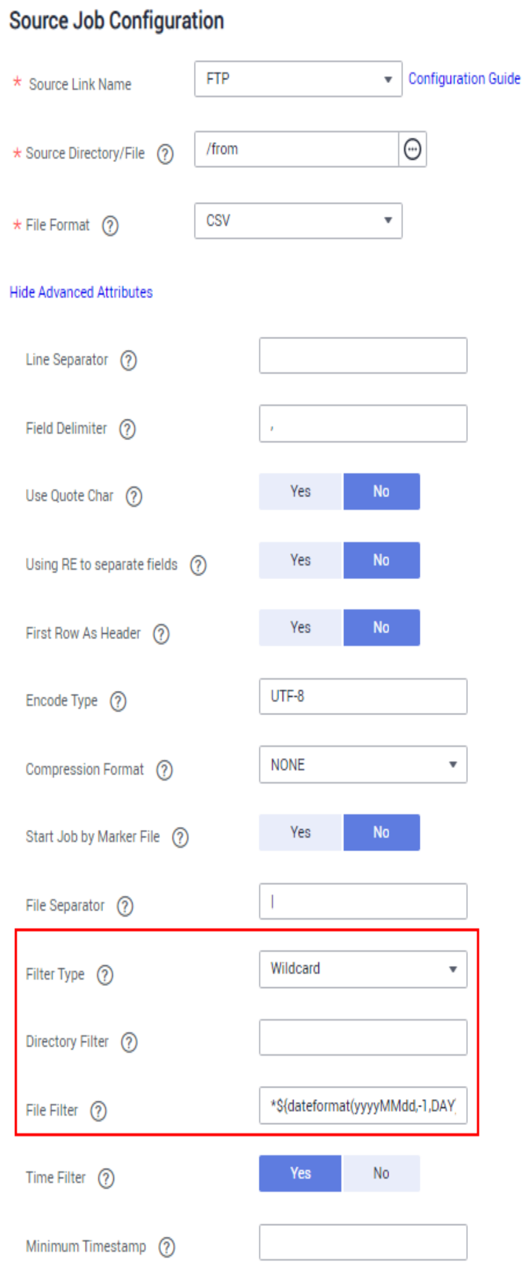
File/Path Filter

- Parameter position: When creating a table/file migration job, if the migration source is a file system, set **Filter Type** in advanced attributes of **Source Job Configuration** to **Wildcard** or **Regular expression**.
- Parameter principle: If you select **Wildcard** for **Filter Type**, CDM filters files or paths based on the configured wildcard character and migrates only files or paths that meet the specified condition.
- Example configurations:

  Suppose that the source file name contains the date and time field, such as **2017-10-15 20:25:26**, the **/opt/data/file_20171015202526.data** file is generated. Set the parameters as follows:

  a. **Filter Type**: Select **Wildcard**.
  b. **File Filter**: Enter **"*${dateformat(yyyyMMdd,-1,DAY)}*"**, which is the format of the macro variables of date and time supported by CDM. For details, see **Using Macro Variables of Date and Time**.

Figure 1-1 Filtering files



Source Job Configuration

* Source Link Name	FTP
* Source Directory/File	/from
* File Format	CSV

Hide Advanced Attributes

Line Separator	
Field Delimiter	,
Use Quote Char	Yes [✓ No]
Using RE to separate fields	Yes [✓ No]
First Row As Header	Yes [✓ No]
Encode Type	UTF-8
Compression Format	NONE
Start Job by Marker File	Yes [✓ No]
File Separator	|
Filter Type	Wildcard
Directory Filter	
File Filter	*${dateformat(yyyyMMdd,-1,DAY
Time Filter	[✓ Yes] No
Minimum Timestamp	

c. Schedule Execution: Set **Cycle (days)** to **1**.

In this way, you can import the files generated in the previous day to the destination directory every day to implement incremental synchronization.

In incremental file migration, **Path Filter** is used in the same way as **File Filter**. The path name must contain the time field. In this case, all files in the specified path can be synchronized periodically.

Time Filter

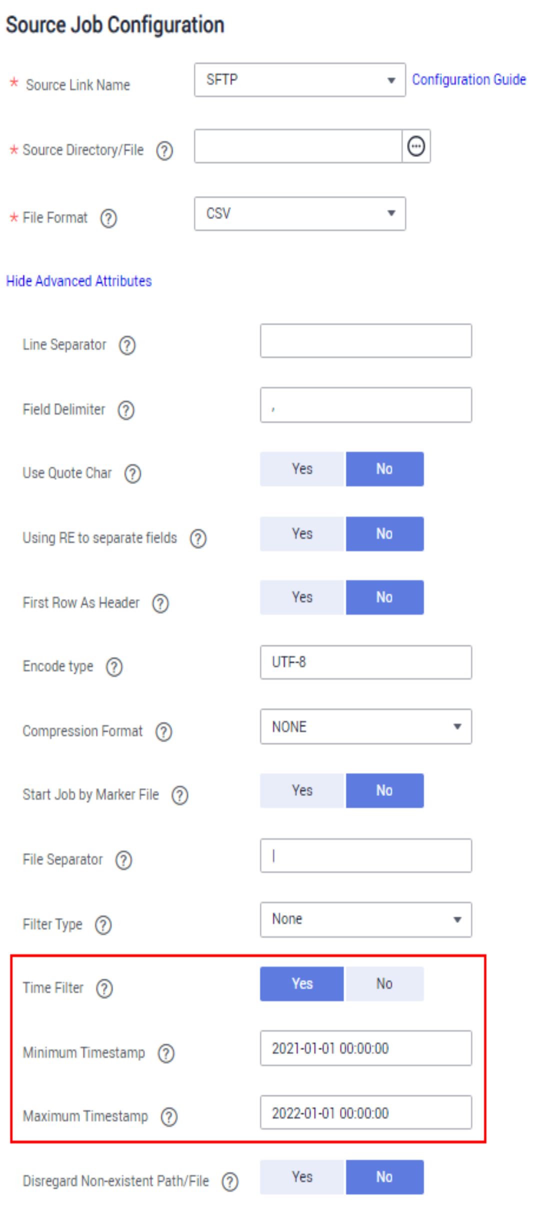
- Parameter position: When creating a table/file migration job, if the migration source is a file system, set select **Yes** for **Time Filter**.
- Parameter principle: Only files generated from the **Minimum Timestamp** to the **Maximum Timestamp** will be migrated by CDM.

- Example configurations:

  For example, if you want CDM to synchronize only the files generated from January 1, 2021 to January 1, 2022 to the destination, configure the following parameters:

  a. **Time Filter**: select **Yes**.
  b. **Minimum Timestamp**: Enter a value in the format of *yyyy-MM-dd HH:mm:ss*, such as **2021-01-01 00:00:00**.
  c. **Maximum Timestamp**: Enter a value in the format of *yyyy-MM-dd HH:mm:ss*, such as **2022-01-01 00:00:00**.

Figure 1-2 Time Filter



Source Job Configuration

* Source Link Name	SFTP
* Source Directory/File	
* File Format	CSV
Line Separator	
Field Delimiter	,
Use Quote Char	Yes [✓ No]
Using RE to separate fields	Yes [✓ No]
First Row As Header	Yes [✓ No]
Encode type	UTF-8
Compression Format	NONE
Start Job by Marker File	Yes [✓ No]
File Separator	|
Filter Type	None
Time Filter	[✓ Yes] No
Minimum Timestamp	2021-01-01 00:00:00
Maximum Timestamp	2022-01-01 00:00:00
Disregard Non-existent Path/File	Yes [✓ No]

In this way, the CDM job migrates only the files generated from January 1, 2021 to January 1, 2022, and performs incremental synchronization next time it is started.

1.1.2 Incremental Migration of Relational Databases

CDM supports incremental migration of relational databases. After a full migration is complete, data in a specified period can be incrementally migrated. For example, data added on the previous day can be exported at 00:00:00 every day.

- **Migrating incremental data within a specified period of time**
  - Application scenarios: The source end is a relational database. The destination end can be of any type.
  - Key configurations: **WHERE Clause** and Schedule Execution
  - Prerequisites: The data table contains a date and time field or timestamp field.

In incremental migration, only the specified data is written to the data table. The existing records are not updated or deleted.

NOTE

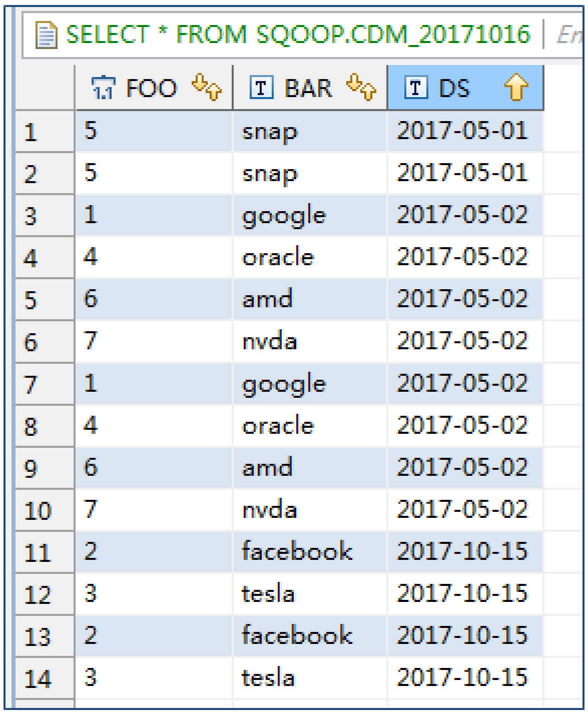
If you have configured a macro variable of date and time and schedule a CDM job through DataArts Studio DataArts Factory, the system replaces the macro variable of date and time with (*Planned start time of the data development job* - *Offset*) rather than (*Actual start time of the CDM job* - *Offset*).

WHERE Clause

- Parameter position: When creating a table/file migration job, if the source end is a relational database, the **Where Clause** parameter is available in the advanced attributes of **Source Job Configuration**.
- Parameter principle: Set **WHERE Clause** to an SQL statement, for example, **age > 18 and age <= 60**, CDM exports only the data that meets the SQL statement requirement. If **WHERE Clause** is not specified, the entire table is exported.

  Where Clause can be set to **macro variables of date and time**. When the data table contains the **date** or **timestamp** field, **Where Clause** and Schedule Execution can be used together to extract data of a specified date.
- Example configurations:

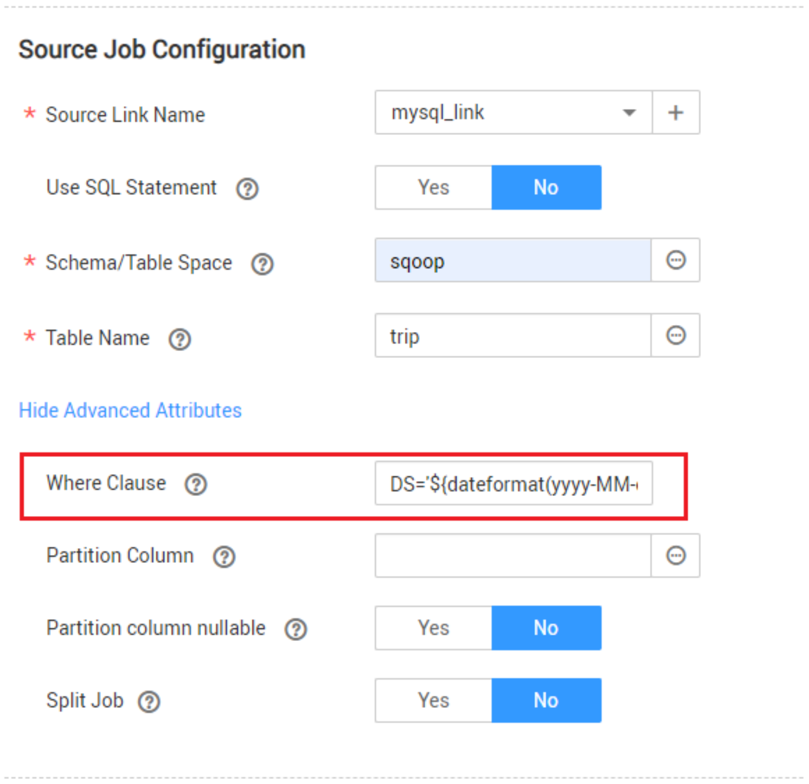
  Suppose that the database table contains column **DS** indicating the time, the value type of the column is **varchar(30)**, and the inserted time format is similar to *2017-xx-xx*. See **Figure 1-3**. Set the parameters as follows:


SELECT * FROM SQOOP.CDM_20171016

	FOO	BAR	DS
1	5	snap	2017-05-01
2	5	snap	2017-05-01
3	1	google	2017-05-02
4	4	oracle	2017-05-02
5	6	amd	2017-05-02
6	7	nvda	2017-05-02
7	1	google	2017-05-02
8	4	oracle	2017-05-02
9	6	amd	2017-05-02
10	7	nvda	2017-05-02
11	2	facebook	2017-10-15
12	3	tesla	2017-10-15
13	2	facebook	2017-10-15
14	3	tesla	2017-10-15

Figure 1-3 Table data

a. **WHERE Clause**: Set this parameter to **DS='${dateformat(yyyy-MM-dd,-1,DAY)}'**.


Source Job Configuration

* Source Link Name	mysql_link
Use SQL Statement	Yes [✓ No]
* Schema/Table Space	sqoop
* Table Name	trip

Hide Advanced Attributes

Where Clause	DS='${dateformat(yyyy-MM-
Partition Column	
Partition column nullable	Yes [✓ No]
Split Job	Yes [✓ No]

Figure 1-4 WHERE Clause

b. Scheduling job execution: Set **Cycle (days)** to **1** and **Start Time** to **00:00:00**.

In this way, all data generated on the previous day can be exported at 00:00:00 every day. **WHERE Clause** can be configured to various **macro variables of date and time**. You can use the macro variables of date and time and scheduled jobs with specified cycle of minutes, hours, days, weeks, or months together to automatically export data at a specific time.

1.1.3 HBase/CloudTable Incremental Migration

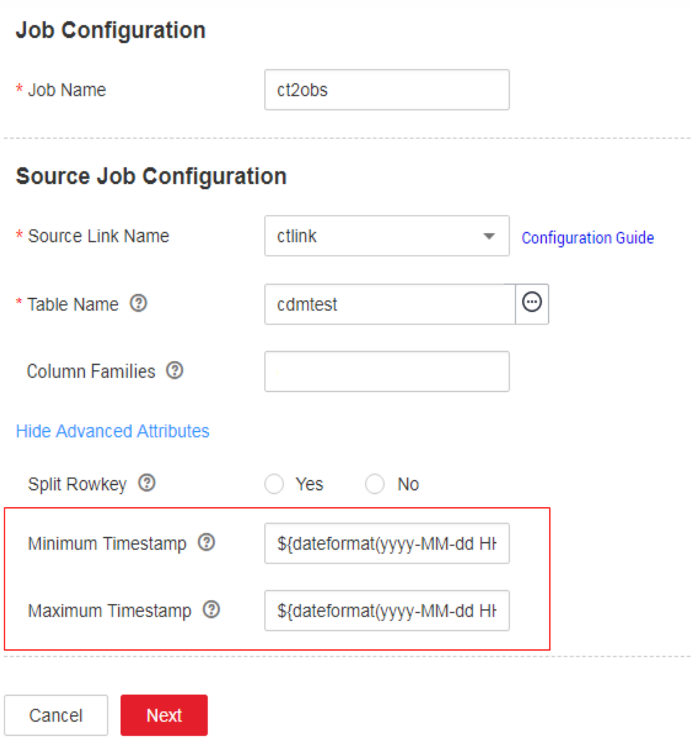
You can use CDM to export data in a specified period of time from HBase (including MRS HBase, FusionInsight HBase, and Apache HBase) and CloudTable. The CDM scheduled jobs can be used together to implement incremental migration of HBase and CloudTable.

NOTE

If you have configured a macro variable of date and time and schedule a CDM job through DataArts Studio DataArts Factory, the system replaces the macro variable of date and time with (*Planned start time of the data development job* – *Offset*) rather than (*Actual start time of the CDM job* – *Offset*).

When creating a table/file migration job and selecting the link to HBase or CloudTable as the source link, you can set the time range in advanced attributes.

Figure 1-5 Time range



Job Configuration

* Job Name: ct2obs

Source Job Configuration

* Source Link Name: ctlink
* Table Name: cdmtest
Column Families:

Split Rowkey: Yes No
Minimum Timestamp: ${dateformat(yyyy-MM-dd HH
Maximum Timestamp: ${dateformat(yyyy-MM-dd HH

- Start time (including the value) for extracting data. The format is *yyyy-MM-dd HH:mm:ss*. Only the data generated at the specified time and later is extracted.

- End time (excluding the value) for extracting data. The format is *yyyy-MM-dd HH:mm:ss*. Only the data generated before the time point is extracted.

The two parameters can be set to **macro variables of date and time**. Examples are as follows:

- If **Minimum Timestamp** is set to **${dateformat(yyyy-MM-dd HH:mm:ss, -1, DAY)}**, only the data generated after the day before is exported.
- If **Maximum Timestamp** is set to **${dateformat(yyyy-MM-dd HH:mm:ss)}**, only the data generated before the specified time point is exported.

If both parameters are configured, CDM exports only the data generated on the previous day. In addition, if the job is configured to execute at 00:00:00 every day, the data generated every day can be incrementally synchronized.

1.2 Using Macro Variables of Date and Time

During the creation of table/file migration jobs, CDM supports the macro variables of date and time in the following parameters of the source and destination links:

- Source directory or file
- Source table name
- Directory filter and file filter of the **wildcard** type
- Start time and end time of the **time filter** type
- Partition filter criteria and where clause
- Write directory
- Destination table name

You can use the **${}** macro variable definition identifier to define the macros of the time type. currently, dateformat and timestamp are supported.

By using the macro variables of date and time and scheduled job, you can implement incremental synchronization of databases and files.

NOTE

If you have configured a macro variable of date and time and schedule a CDM job through DataArts Studio DataArts Factory, the system replaces the macro variable of date and time with (*Planned start time of the data development job* – *Offset*) rather than (*Actual start time of the CDM job* – *Offset*).

dateformat

dateformat supports two types of parameters:

- **dateformat(format)**

  format indicates the date and time format. For details about the format definition, see the definition in **java.text.SimpleDateFormat.java**.

  For example, if the current date is **2017-10-16 09:00:00**, **yyyy-MM-dd HH:mm:ss** indicates **2017-10-16 09:00:00**.
- dateformat(format, dateOffset, dateType)
  - **format** indicates the format of the returned date.

- **dateOffset** indicates the date offset.
- **dateType** indicates the type of the date offset.

  Currently, **dateType** supports SECOND, MINUTE, HOUR, and DAY.

For example, if the current date is **2017-10-16 09:00:00**, then:

- **dateformat(yyyy-MM-dd HH:mm:ss, -1, DAY)** indicates the day before the current day, that is, **2017-10-15 09:00:00**.
- **dateformat(yyyy-MM-dd HH:mm:ss, -1, HOUR)** indicates one hour before the current time, that is, **2017-10-16 08:00:00**.
- **dateformat(yyyy-MM-dd HH:mm:ss, -1, MINUTE)** indicates one minute before the current time, that is, **2017-10-16 08:59:00**.
- **dateformat(yyyy-MM-dd HH:mm:ss, -1, SECOND)** indicates one second before the current time, that is, **2017-10-16 08:59:59**.

timestamp

timestamp supports two types of parameters:

- **timestamp()**

  Indicates the returned timestamp of the current time, that is, the number of milliseconds that have elapsed since 00:00:00 on January 1, 1970 (1970-01-01 00:00:00 GMT). For example, 1508078516286.
- **timestamp(dateOffset, dateType)**

  Indicates the timestamp returned after time offset. **dateOffset** and **dateType** indicate the date offset and the offset type, respectively.

  For example, if the current date is **2017-10-16 09:00:00**, **timestamp(-10, MINUTE)** indicates that the timestamp generated 10 minutes before the current time point is returned, that is, **1508115000000**.

Macro Variable Definition of Time and Date

Suppose that the current time is **2017-10-16 09:00:00**, then **Table 1-1** describes the macro variable definitions of time and date.

Table 1-1 Macro variable definition of time and date

Macro Variable	Description	Display Effect
${dateformat(yyyy-MM-dd)}	Returns the current date in **yyyy-MM-dd** format.	2017-10-16
${dateformat(yyyy/MM/dd)}	Returns the current date in **yyyy/MM/dd** format.	2017/10/16
${dateformat(yyyy_MM_dd HH:mm:ss)}	Returns the current time in **yyyy_MM_dd HH:mm:ss** format.	2017_10_16 09:00:00
${dateformat(yyyy-MM-dd HH:mm:ss, -1, DAY)}	Returns the current time in **yyyy-MM-dd HH:mm:ss** format. The date is one day before the current day.	2017-10-15 09:00:00

Macro Variable	Description	Display Effect
${timestamp()}	Returns the timestamp of the current time, that is, the number of milliseconds that have elapsed since 00:00:00 on January 1, 1970.	1508115600000
${timestamp(-10, MINUTE)}	Returns the timestamp generated 10 minutes before the current time point.	1508115000000
${timestamp(dateformat(yyyyMMdd))}	Returns the timestamp of 00:00:00 of the current day.	1508083200000
${timestamp(dateformat(yyyyMMdd,-1,DAY))}	Returns the timestamp of 00:00:00 of the previous day.	1507996800000
${timestamp(dateformat(yyyyMMddHH))}	Returns the timestamp of the current hour.	1508115600000

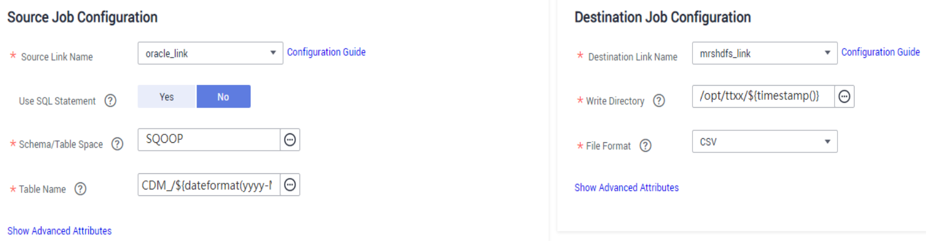
Time and Date Macro Variables of Paths and Table Names

Figure 1-6 shows an example. If:

- **Table Name** under **Source Link Configuration** is set to **CDM_/${dateformat(yyyy-MM-dd)}**.
- **Write Directory** under **Destination Link Configuration** is set to **/opt/ttxx/${timestamp()}**.

After the macro definition conversion, this job indicates that data in table **SQOOP.CDM_20171016** in the Oracle database is migrated to the **/opt/ttxx/1508115701746** directory of the HDFS server.

Figure 1-6 Setting **Table Name** and **Write Directory** to a time and date macro variable



Source Job Configuration

* Source Link Name	oracle_link	Configuration Guide
Use SQL Statement	Yes [✓ No]	
* Schema/Table Space	SQOOP	
* Table Name	CDM_/${dateformat(yyyy-I	

Show Advanced Attributes

Destination Job Configuration

* Destination Link Name	mrshdfs_link	Configuration Guide
* Write Directory	/opt/ttxx/${timestamp()}	
* File Format	CSV	

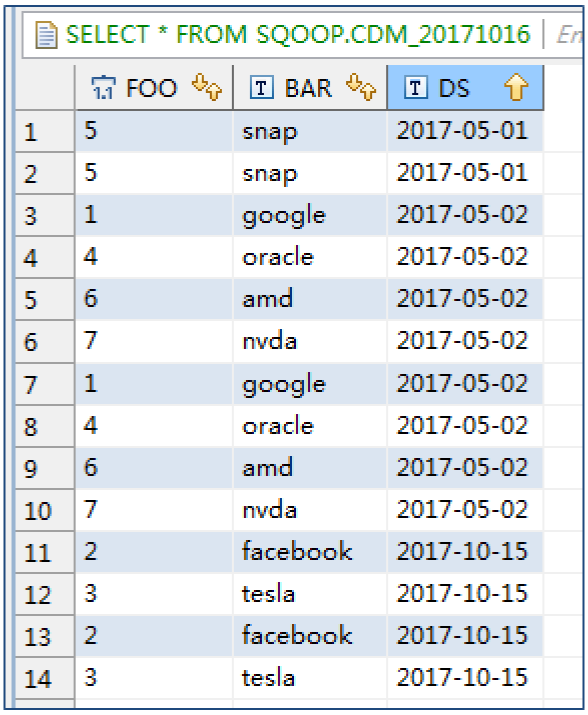
Show Advanced Attributes

Currently, a table name or path name can contain multiple macro variables. For example, **/opt/ttxx/${dateformat(yyyy-MM-dd)}/${timestamp()}** is converted to **/opt/ttxx/2017-10-16/1508115701746**.

Time and Date Macro Variables in the Where Clause

Figure 1-7 uses table **SQOOP.CDM_20171016** as an example. The table contains column **DS**, which indicates the time.

Figure 1-7 Table data



SELECT * FROM SQOOP.CDM_20171016

	FOO	BAR	DS
1	5	snap	2017-05-01
2	5	snap	2017-05-01
3	1	google	2017-05-02
4	4	oracle	2017-05-02
5	6	amd	2017-05-02
6	7	nvda	2017-05-02
7	1	google	2017-05-02
8	4	oracle	2017-05-02
9	6	amd	2017-05-02
10	7	nvda	2017-05-02
11	2	facebook	2017-10-15
12	3	tesla	2017-10-15
13	2	facebook	2017-10-15
14	3	tesla	2017-10-15

Suppose that the current date is **2017-10-16** and you want to export data generated the day before the current day (DS = 2017-10-15), then you can set the value of **Where Clause** to **DS='${dateformat(yyyy-MM-dd,-1,DAY)}'** when creating a job. In this way, you can export all data that complies with the DS = 2017-10-15 condition.

Implementing Incremental Synchronization by Configuring the Macro Variables of Date and Time and Scheduled Jobs

Two simple application scenarios are as follows:

- The database table contains column **DS** that indicates the time, the value type of the column is **varchar(30)**, and the inserted time format is similar to **2017-xx-xx**.

  In a scheduled job, the cycle is one day, and the scheduled job is executed at 00:00:00 every day. Set the value of **Where Clause** to **DS='${dateformat(yyyy-MM-dd,-1,DAY)}'**, and then data generated in the previous day will be exported at 00:00:00 every day.
- The database table contains column **time** that indicates the time, the type is **Number**, and the inserted time format is timestamp.

  In a scheduled job, the cycle is one day, and the scheduled job is executed at 00:00:00 every day. Set the value of **Where Clause** to **time between ${timestamp(-1,DAY)} and ${timestamp()}**, and then data generated on the previous day will be exported at 00:00:00 every day.

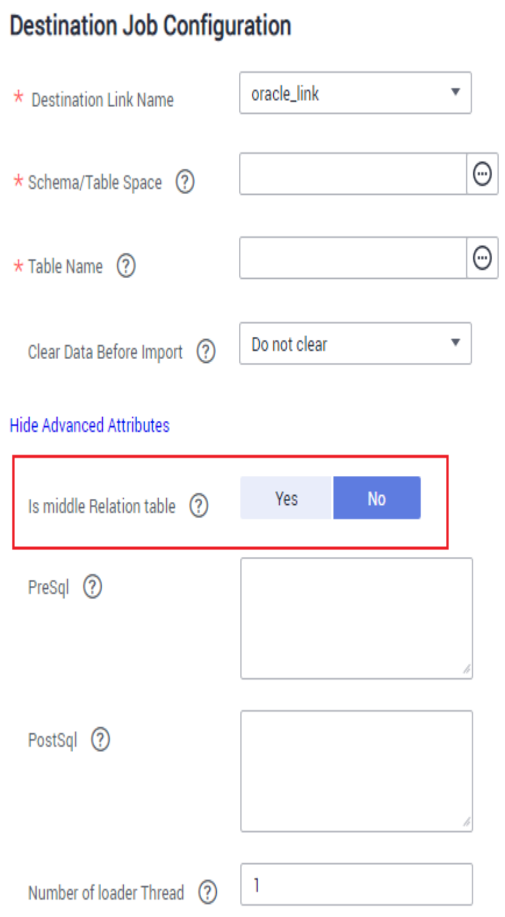
Configuration principles of other application scenarios are the same.

1.3 Migration in Transaction Mode

When a CDM job fails to be executed, CDM rolls back the data to the state before the job starts and automatically deletes data from the destination table.

- Parameter position: When creating a table/file migration job, if the migration source is a relational database, set **Import to Staging Table** in the advanced attributes of **Destination Job Configuration** to determine whether to enable the transaction mode.
- Parameter principle: If you set this parameter to **Yes**, CDM automatically creates a temporary table and imports the data to the temporary table. After the data is imported successfully, CDM migrates the data to the destination table in transaction mode of the database. If the import fails, the destination table is rolled back to the state before the job starts.

Figure 1-8 Migration in transaction mode


Destination Job Configuration

* Destination Link Name	oracle_link
* Schema/Table Space	
* Table Name	
Clear Data Before Import	Do not clear

Is middle Relation table	Yes [✓ No]
PreSql	
PostSql	
Number of loader Thread	1

NOTE

If you select **Clear part of data** or **Clear all data** for **Clear Data Before Import**, CDM does not roll back the deleted data in transaction mode.

1.4 Encryption and Decryption During File Migration

When you migrate files to a file system, CDM can encrypt and decrypt those files. Currently, CDM supports the following encryption modes:

- **AES-256-GCM**
- **KMS Encryption**

AES-256-GCM

Currently, only AES-256-GCM (NoPadding) is supported. This algorithm is used for encryption at the migration destination and decryption at the migration source. The supported source and destination data sources are as follows:

- Data sources supported by the migration source: OBS, FTP, SFTP, HDFS (supported in the binary format), and HTTP (applicable to scenarios where OBS shared files are downloaded)
- Data sources supported by the migration destination: OBS, FTP, SFTP, and HDFS (supported in the binary format)

The following part describes how to use AES-256-GCM to decrypt the encrypted files to be exported from OBS and encrypt the files to be imported to OBS. The methods for using the algorithm on other data sources are the same.

- **Configure decryption at the migration source.**

  When you use CDM to create a job for exporting files from OBS, set the migration source to OBS and set the following parameters in the advanced settings of **Source Job Configuration**:

  a. **Encryption**: Select **AES-256-GCM**.
  b. **DEK**: The key must be the same as that configured in **Encryption**. Otherwise, the decrypted data is incorrect and the system does not display an error message.
  c. **IV**: The initialization vector must be the same as that configured in **Encryption**. Otherwise, the decrypted data is incorrect and the system does not display an error message.

  In this way, after CDM exports encrypted files from OBS, the files written to the migration destination are decrypted plaintext files.

- **Configure encryption at the migration destination.**

  When you use CDM to create a job for importing files to OBS, set the migration destination to OBS and set the following parameters in the advanced settings of **Destination Job Configuration**:

  a. **Encryption**: Select **AES-256-GCM**.
  b. **DEK**: custom encryption key. The key consists of 64 hexadecimal numbers. It is case-insensitive but must contain 64 characters. For example, **DD0AE00DFECD78BF051BCFDA25BD4E320DB0A7AC75A1F3FC3D3C56A457DCDC1B**.

c. **IV**: custom initialization vector. The initialization vector consists of 32 hexadecimal numbers. It is case-insensitive but must contain 32 characters. For example, **5C91687BA886EDCD12ACBC3FF19A3C3F**.

In this way, after CDM imports files to OBS, the files on the migration destination are encrypted using the AES-256-GCM algorithm.

KMS Encryption

NOTE

The migration source does not support KMS encryption.

CDM supports KMS encryption if tables, files, or a whole database is migrated to OBS. In the **Advanced Attributes** area of the **Destination Job Configuration** page, set the parameters.

A key must be created in KMS of DEW in advance. For details, see the *Data Encryption Workshop User Guide*.

After KMS encryption is enabled, objects to be uploaded will be encrypted and stored on OBS. When you download the encrypted objects, the encrypted data will be decrypted on the server and displayed in plaintext to users.

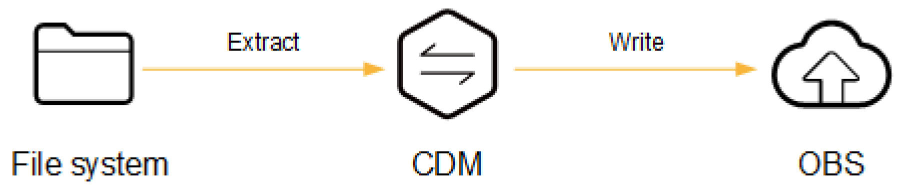
NOTE

- If KMS encryption is enabled, **MD5 verification** cannot be used.
- If the KMS ID of another project is used, change **Project ID** to the ID of the project to which KMS belongs. If KMS and CDM are in the same project, retain the default value of **Project ID**.
- After KMS encryption is performed, the encryption status of the objects on OBS cannot be changed.
- A key in use cannot be deleted. Otherwise, the object encrypted with this key cannot be downloaded.

1.5 MD5 Verification

CDM extracts data from the migration source and writes the data to the migration destination. **Figure 1-9** shows the migration mode when files are migrated to OBS.

Figure 1-9 Migrating files to OBS



During the process, CDM uses MD5 to verify file consistency.

- **Extract**
  - The migration source can be OBS, HDFS, FTP, SFTP, or HTTP. It can check whether the files extracted by CDM are consistent with source files.

  - This function is controlled by the **MD5 File Extension** parameter (available when **File Format** is set to **Binary**) in **Source Job Configuration**. Set this parameter to the file name extension of the MD5 file in the source file system.
  - If a source file **build.sh** and a file for saving MD5 value **build.sh.md5** are located in the same directory, and **MD5 File Extension** is configured, only the file **build.sh.md5** is migrated to the destination. Files without the MD5 value or whose MD5 values do not match fail to be migrated, and the MD5 file is not migrated.
  - If **MD5 File Extension** is not configured, all files are migrated.
- **Write**
  - Currently, this function can be used only when OBS serves as the migration destination. It can check whether the files written to OBS are consistent with those extracted from CDM.
  - This function is controlled by the **Validate MD5 Value** parameter in **Destination Job Configuration**. After the files are read and written to OBS, the MD5 value in the HTTP header is used to verify the files on OBS and the verification result is written to an OBS bucket (the bucket can be the one that does not store migration files). If the migration source does not have the MD5 file, the verification will not be performed.

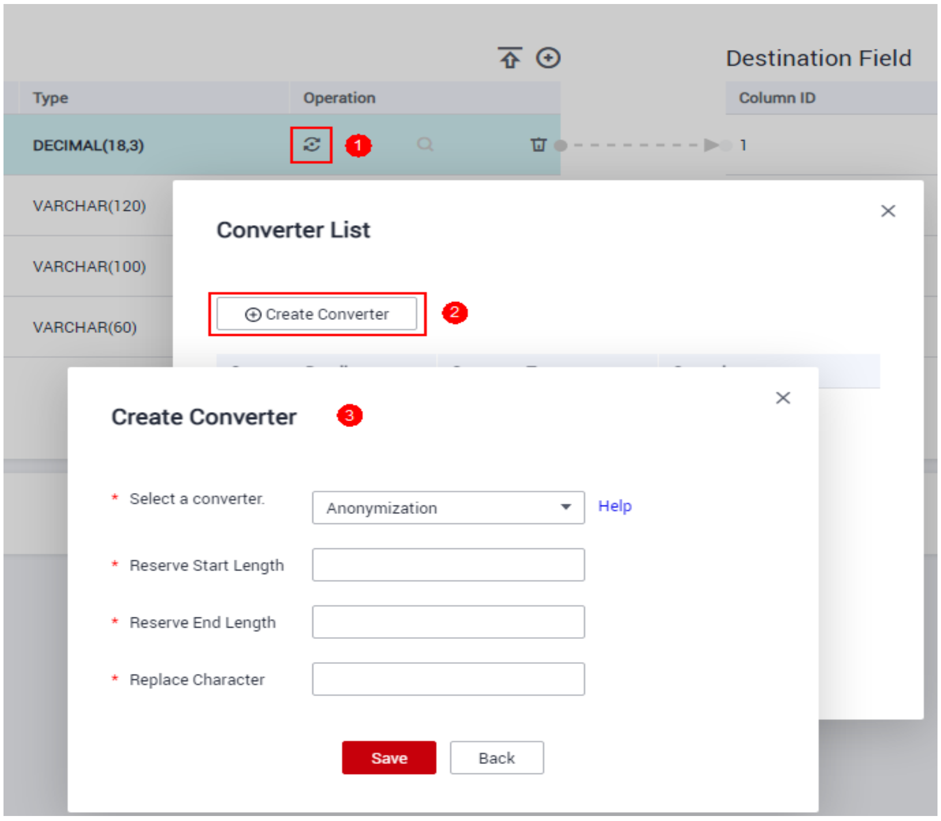
NOTE

- When files are migrated to a file system, only the extracted files are verified.
- When files are migrated to OBS, both the extracted files and files written to OBS are verified.
- If MD5 verification is used, **KMS encryption** cannot be used.

1.6 Field Conversion

You can create a field converter on the **Map Field** page when creating a table/file migration job.

Figure 1-10 Creating a field converter



NOTE

Field mapping is not involved when the binary format is used to migrate files to files.

CDM can convert fields during migration. Currently, the following field converters are supported:

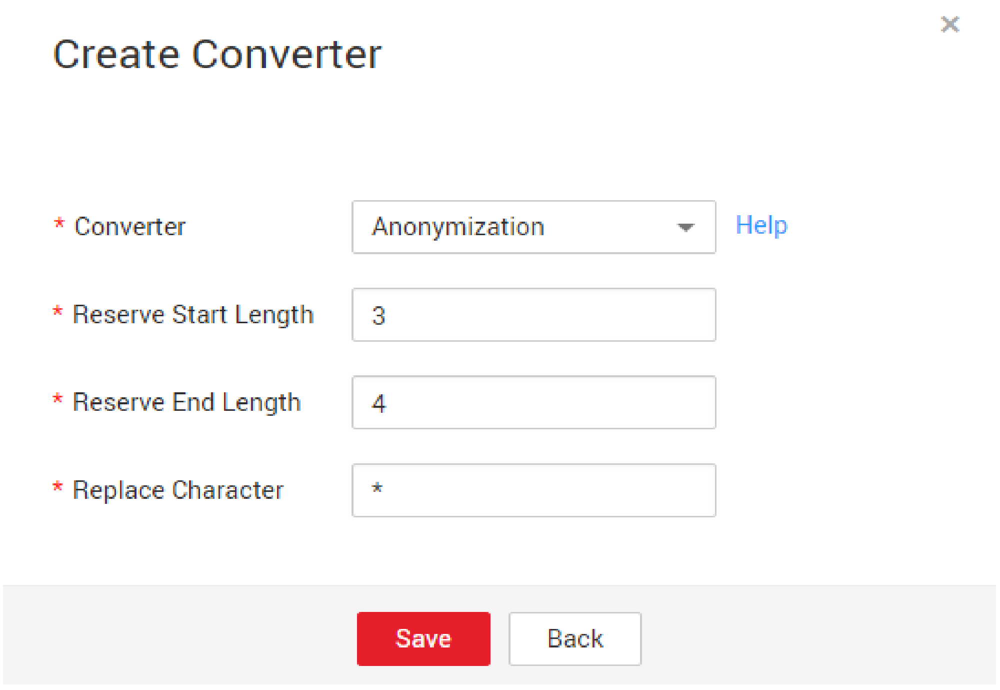
- **Anonymization**
- **Trim**
- **Reverse String**
- **Replace String**
- **Remove line break**
- **Expression Conversion**

Anonymization

This converter is used to hide key information about the character string. For example, if you want to convert **12345678910** to **123****8910**, configure the parameters as follows:

- Set **Reserve Start Length** to **3**.
- Set **Reserve End Length** to **4**.
- Set **Replace Character** to *.

Figure 1-11 Anonymization


Create Converter

* Converter	Anonymization
* Reserve Start Length	3
* Reserve End Length	4
* Replace Character	*

Trim

This converter is used to automatically delete the spaces before and after a string. No parameters need to be configured.

Reverse String

This converter is used to automatically reverse a string. For example, reverse **ABC** into **CBA**. No parameters need to be configured.

Replace String

This converter is used to replace a character string. You need to configure the object to be replaced and the new value.

Remove line break

This converter is used to delete the newline characters, such as \n, \r, and \r\n from the field.

Expression Conversion

This converter uses the JSP expression language (EL) to convert the current field or a row of data. The JSP EL is used to create arithmetic and logical expressions. Within a JSP EL expression, you can use integers, floating point numbers, strings, the built-in constants **true** and **false** for boolean values, and **null**.

- The expression supports the following environment variables:
  - **value**: indicates the current field value.

  - **row**: indicates the current row, which is an array type.

- The expression supports the following Utils:

  a. If the field is of the string type, convert all character strings into lowercase letters, for example, convert **aBC** to **abc**.

     Expression: StringUtils.lowerCase(value)

  b. Convert all character strings of the current field to uppercase letters.

     Expression: StringUtils.upperCase(value)

  c. Convert the format of the first date field from 2018-01-05 15:15:05 to 20180105.

     Expression: DateUtils.format(DateUtils.parseDate(row[0],"yyyy-MM-dd HH:mm:ss"),"yyyyMMdd")

  d. If the field value is a date string in *yyyy-MM-dd* format, extract the year from the field value, for example, extract **2017** from **2017-12-01**.

     Expression: StringUtils.substringBefore(value,"-")

  e. If the field value is of the numeric type, convert the value to a new value which is two times greater than the original value:

     Expression: value*2

  f. Convert the field value **true** to **Y** and other field values to **N**.

     Expression: value=="true"?"Y":"N"

  g. If the field value is of the string type and is left empty, convert it to **Default**. Otherwise, the field value will not be converted.

     Expression: empty value? "Default":value

  h. Convert date format **2018/01/05 15:15:05** to **2018-01-05 15:15:05**:

     Expression: DateUtils.format(DateUtils.parseDate(value,"yyyy/MM/dd HH:mm:ss"),"yyyy-MM-dd HH:mm:ss")

  i. Obtain a 36-bit universally unique identifier (UUID):

     Expression: CommonUtils.randomUUID()

  j. If the field is of the string type, capitalize the first letter, for example, convert **cat** to **Cat**.

     Expression: StringUtils.capitalize(value)

  k. If the field is of the string type, convert the first letter to a lowercase letter, for example, convert **Cat** to **cat**.

     Expression: StringUtils.uncapitalize(value)

  l. If the field is of the string type, use a space to fill in the character string to the specified length and center the character string. If the length of the character string is not shorter than the specified length, do not convert the character string. For example, convert **ab** to meet the specified length 4.

     Expression: StringUtils.center(value,*4*)

  m. Delete a newline (including **\n**, **\r**, and **\r\n**) at the end of a character string. For example, convert **abc\r\n\r\n** to **abc\r\n**.

     Expression: StringUtils.chomp(value)

  n. If the string contains the specified string, **true** is returned; otherwise, **false** is returned. For example, **abc** contains **a** so that **true** is returned.

Expression: StringUtils.contains(value,"*a*")

o. If the string contains any character of the specified string, **true** is returned; otherwise, **false** is returned. For example, **zzabyycdxx** contains either **z** or **a** so that **true** is returned.

   Expression: StringUtils.containsAny("value","*za*")

p. If the string does not contain any one of the specified characters, **true** is returned. If any specified character is contained, **false** is returned. For example, **abz** contains one character of **xyz** so that **false** is returned.

   Expression: StringUtils.containsNone(value,"*xyz*")

q. If the string contains only the specified characters, **true** is returned. If any other character is contained, **false** is returned. For example, **abab** contains only characters among **abc** so that **true** is returned.

   Expression: StringUtils.containsOnly(value,"*abc*")

r. If the character string is empty or null, convert it to the specified character string. Otherwise, do not convert the character string. For example, convert the empty character string to null.

   Expression: StringUtils.defaultIfEmpty(value,*null*)

s. If the string ends with the specified suffix (case sensitive), **true** is returned; otherwise, **false** is returned. For example, if the suffix of **abcdef** is not null, **false** is returned.

   Expression: StringUtils.endsWith(value,*null*)

t. If the string is the same as the specified string (case sensitive), **true** is returned; otherwise, **false** is returned. For example, after strings **abc** and **ABC** are compared, **false** is returned.

   Expression: StringUtils.equals(value,"*ABC*")

u. Obtain the first index of the specified character string in a character string. If no index is found, **-1** is returned. For example, the first index of **ab** in **aabaabaa** is 1.

   Expression: StringUtils.indexOf(value,"*ab*")

v. Obtain the last index of the specified character string in a character string. If no index is found, **-1** is returned. For example, the last index of **k** in **aFkyk** is 4.

   Expression: StringUtils.lastIndexOf(value,"*k*")

w. Obtain the first index of the specified character string from the position specified in the character string. If no index is found, **-1** is returned. For example, the first index of **b** obtained after the index 3 of **aabaabaa** is 5.

   Expression: StringUtils.indexOf(value,"*b*",*3*)

x. Obtain the first index of any specified character in a character string. If no index is found, **-1** is returned. For example, the first index of **z** or **a** in **zzabyycdxx.** is 0.

   Expression: StringUtils.indexOfAny(value,"*za*")

y. If the string contains any Unicode character, **true** is returned; otherwise, **false** is returned. For example, **ab2c** contains only non-Unicode characters so that **false** is returned.

   Expression: StringUtils.isAlpha(value)

z. If the string contains only Unicode characters and digits, **true** is returned; otherwise, **false** is returned. For example, **ab2c** contains only Unicode characters and digits, so that **true** is returned.

   Expression: StringUtils.isAlphanumeric(value)

aa. If the string contains only Unicode characters, digits, and spaces, **true** is returned; otherwise, **false** is returned. For example, **ab2c** contains only Unicode characters and digits, so that **true** is returned.

   Expression: StringUtils.isAlphanumericSpace(value)

ab. If the string contains only Unicode characters and spaces, **true** is returned; otherwise, **false** is returned. For example, **ab2c** contains Unicode characters and digits so that **false** is returned.

   Expression: StringUtils.isAlphaSpace(value)

ac. If the string contains only printable ASCII characters, **true** is returned; otherwise, **false** is returned. For example, for **!ab-c~**, **true** is returned.

   Expression: StringUtils.isAsciiPrintable(value)

ad. If the string is empty or null, **true** is returned; otherwise, **false** is returned.

   Expression: StringUtils.isEmpty(value)

ae. If the string contains only Unicode digits, **true** is returned; otherwise, **false** is returned.

   Expression: StringUtils.isNumeric(value)

af. Obtain the leftmost characters of the specified length. For example, obtain the leftmost two characters **ab** from **abc**.

   Expression: StringUtils.left(value,*2*)

ag. Obtain the rightmost characters of the specified length. For example, obtain the rightmost two characters **bc** from **abc**.

   Expression: StringUtils.right(value,*2*)

ah. Concatenate the specified character string to the left of the current character string and specify the length of the concatenated character string. If the length of the current character string is not shorter than the specified length, the character string will not be converted. For example, if **yz** is concatenated to the left of **bat** and the length must be 8 after concatenation, the character string is **yzyzybat** after conversion.

   Expression: StringUtils.leftPad(value,*8*,"*yz*")

ai. Concatenate the specified character string to the right of the current character string and specify the length of the concatenated character string. If the length of the current character string is not shorter than the specified length, the character string will not be converted. For example, if **yz** is concatenated to the right of **bat** and the length must be 8 after concatenation, the character string is **batyzyzy** after conversion.

   Expression: StringUtils.rightPad(value,*8*,"*yz*")

aj. If the field is of the string type, obtain the length of the current character string. If the character string is null, **0** is returned.

   Expression: StringUtils.length(value)

ak. If the field is of the string type, delete all the specified character strings from it. For example, delete **ue** from **queued** to obtain **qd**.

Expression: StringUtils.remove(value,"*ue*")

al. If the field is of the string type, remove the substring at the end of the field. If the specified substring is not at the end of the field, no conversion is performed. For example, remove **.com** at the end of **www.domain.com**.

Expression: StringUtils.removeEnd(value,"*.com*")

am. If the field is of the string type, delete the substring at the beginning of the field. If the specified substring is not at the beginning of the field, no conversion is performed. For example, delete **www.** at the beginning of **www.domain.com**.

Expression: StringUtils.removeStart(value,"*www.*")

an. If the field is of the string type, replace all the specified character strings in the field. For example, replace **a** in **aba** with **z** to obtain **zbz**.

Expression: StringUtils.replace(value,"*a*","*z*")

ao. If the field is of the string type, replace multiple characters in the character string at a time. For example, replace **h** in **hello** with **j** and **o** with **y** to obtain **jelly**.

Expression: StringUtils.replaceChars(value,"*ho*","*jy*")

ap. If the string starts with the specified prefix (case sensitive), **true** is returned; otherwise, **false** is returned. For example, **abcdef** starts with **abc**, so that **true** is returned.

Expression: StringUtils.startsWith(value,"*abc*")

aq. If the field is of the string type, delete all the specified characters from the field. For example, delete all **x**, **y**, and **z** from **abcyx** to obtain **abc**.

Expression: StringUtils.strip(value,"*xyz*")

ar. If the field is of the string type, delete all the specified characters at the end of the field, for example, delete all spaces at the end of the field.

Expression: StringUtils.stripEnd(value,*null*)

as. If the field is of the string type, delete all the specified characters at the beginning of the field, for example, delete all spaces at the beginning of the field.

Expression: StringUtils.stripStart(value,*null*)

at. If the field is of the string type, obtain the substring after the specified position (the index starts from 0, including the character at the specified position) of the character string. If the specified position is a negative number, calculate the position in the descending order. The first digit at the end is -1. For example, obtain the second character (c) of **abcde** and the string after it, that is, **cde**.

Expression: StringUtils.substring(value,*2*)

au. If the field is of the string type, obtain the substring in a specified range (the index starts from 0, including the character at the start and excluding the character at the end). If the range is a negative number, calculate the position in the descending order. The first digit at the end is -1. For example, obtain the string between the second character (c) and fourth character (e) of **abcde**, that is, **cd**.

Expression: StringUtils.substring(value,*2*,4) av. If the field is of the string type, obtain the substring after the first specified character. For example, obtain the substring after the first **b** in **abcba**, that is, **cba**.

Expression: StringUtils.substringAfter(value,"*b*")

aw. If the field is of the string type, obtain the substring after the last specified character. For example, obtain the substring after the last **b** in **abcba**, that is, **a**.

Expression: StringUtils.substringAfterLast(value,"*b*")

ax. If the field is of the string type, obtain the substring before the first specified character. For example, obtain the substring before the first **b** in **abcba**, that is, **a**.

Expression: StringUtils.substringBefore(value,"*b*")

ay. If the field is of the string type, obtain the substring before the last specified character. For example, obtain the substring before the last **b** in **abcba**, that is, **abc**.

Expression: StringUtils.substringBeforeLast(value,"*b*")

az. If the field is of the string type, obtain the substring nested within the specified string. If no substring is found, **null** is returned. For example, obtain the substring between **tag** in **tagabctag**, that is, **abc**.

Expression: StringUtils.substringBetween(value,"*tag*")

ba. If the field is of the string type, delete the control characters (char≤32) at both ends of the character string, for example, delete the spaces at both ends of the character string.

Expression: StringUtils.trim(value)

bb. Convert the character string to a value of the byte type. If the conversion fails, **0** is returned.

Expression: NumberUtils.toByte(value)

bc. Convert the character string to a value of the byte type. If the conversion fails, the specified value, for example, **1**, is returned.

Expression: NumberUtils.toByte(value,*1*)

bd. Convert the character string to a value of the double type. If the conversion fails, **0.0d** is returned.

Expression: NumberUtils.toDouble(value)

be. Convert the character string to a value of the double type. If the conversion fails, the specified value, for example, **1.1d**, is returned.

Expression: NumberUtils.toDouble(value,*1.1d*)

bf. Convert the character string to a value of the float type. If the conversion fails, **0.0f** is returned.

Expression: NumberUtils.toFloat(value)

bg. Convert the character string to a value of the float type. If the conversion fails, the specified value, for example, **1.1f**, is returned.

Expression: NumberUtils.toFloat(value,*1.1f*)

bh. Convert the character string to a value of the int type. If the conversion fails, **0** is returned.

Expression: NumberUtils.toInt(value)

bi. Convert the character string to a value of the int type. If the conversion fails, the specified value, for example, **1**, is returned.

    Expression: NumberUtils.toInt(value,*1*)

bj. Convert the character string to a value of the long type. If the conversion fails, **0** is returned.

    Expression: NumberUtils.toLong(value)

bk. Convert the character string to a value of the long type. If the conversion fails, the specified value, for example, **1L**, is returned.

    Expression: NumberUtils.toLong(value,*1L*)

bl. Convert the character string to a value of the short type. If the conversion fails, **0** is returned.

    Expression: NumberUtils.toShort(value)

bm. Convert the character string to a value of the short type. If the conversion fails, the specified value, for example, **1**, is returned.

    Expression: NumberUtils.toShort(value,*1*)

bn. Convert the IP string to a value of the long type, for example, convert **10.78.124.0** to **172915712**.

    Expression: CommonUtils.ipToLong(value)

bo. Read an IP address and physical address mapping file from the network, and download the mapping file to the map collection. *url* indicates the address for storing the IP mapping file, for example, **http://10.114.205.45:21203/sqoop/IpList.csv**.

    Expression: HttpsUtils.downloadMap("*url*")

bp. Cache the IP address and physical address mappings and specify a key for retrieval, for example, **ipList**.

    Expression: CommonUtils.setCache("*ipList*",HttpsUtils.downloadMap("*url*"))

bq. Obtain the cached IP address and physical address mappings.

    Expression: CommonUtils.getCache("*ipList*")

br. Check whether the IP address and physical address mappings are cached.

    Expression: CommonUtils.cacheExists("*ipList*")

bs. Based on the specified offset type (month/day/hour/minute/second) and offset (positive number indicates increase and negative number indicates decrease), convert the time in the specified format to a new time, for example, add 8 hours to **2019-05-21 12:00:00**.

    Expression: DateUtils.getCurrentTimeByZone("*yyyy-MM-dd HH:mm:ss*",value, "*hour*", *8*)

1.7 Migrating Files with Specified Names

You can migrate files (a maximum of 50) with specified names from FTP, SFTP, or OBS at a time. The exported files can only be written to the same directory on the migration destination.

When creating a table/file migration job, if the migration source is FTP, SFTP, or OBS, **Source Directory/File** can contain a maximum of 50 file names, which are separated by vertical bars (|). You can also customize a file separator.

NOTE

1. CDM supports incremental file migration (by skipping repeated files), but does not support resumable transfer.

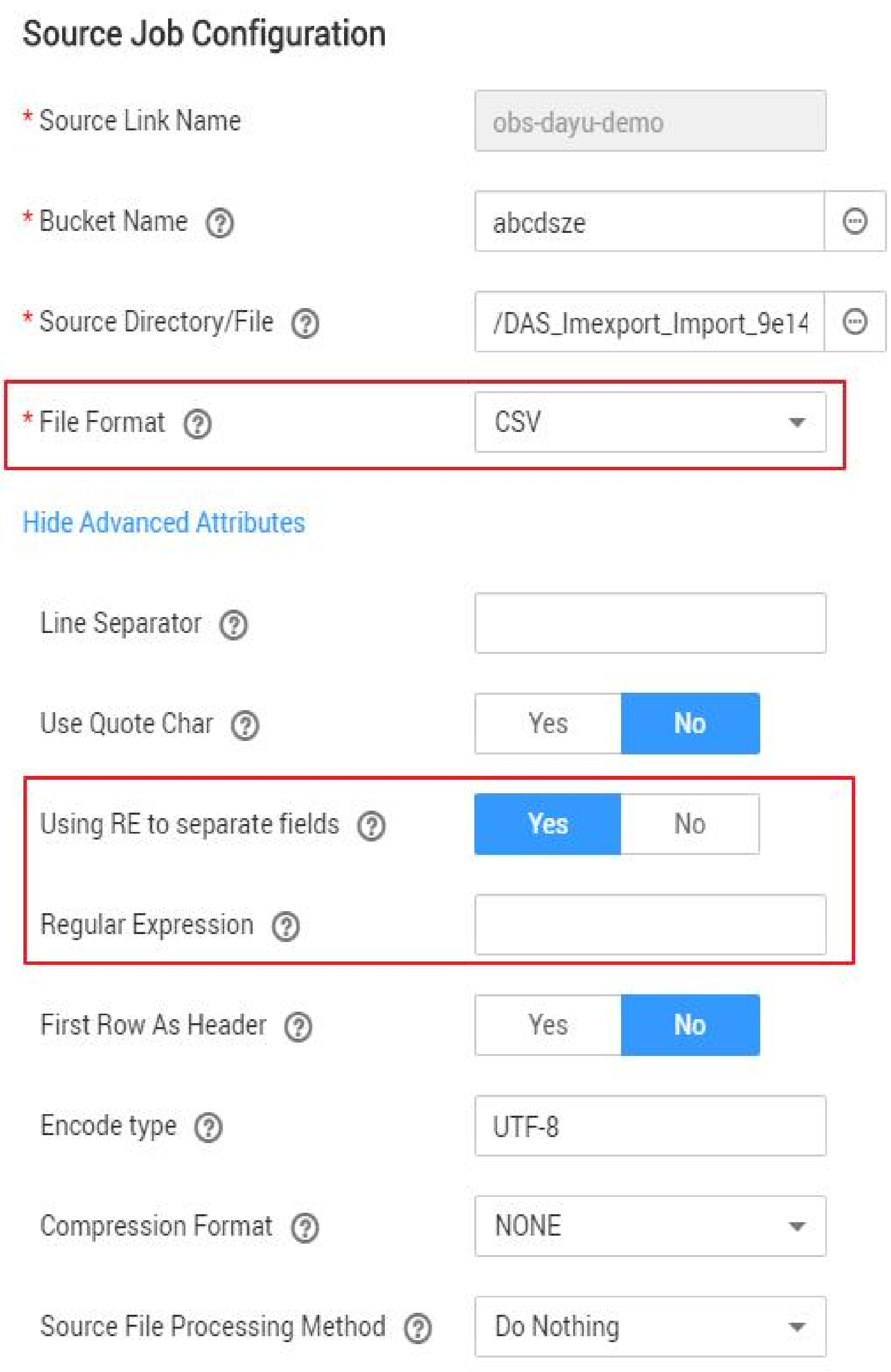
   For example, if three files are to be migrated and the second file fails to be migrated due to the network fault. When the migration task is started again, the first file is skipped. The second file, however, cannot be migrated from the point where the fault occurs, but can only be migrated again.
2. During file migration, a single task supports millions of files. If there are too many files in the directory to be migrated, you are advised to split the files into different directories and create multiple tasks.

1.8 Regular Expressions for Separating Semi-structured Text

During table/file migration, CDM uses delimiters to separate fields in CSV files. However, delimiters cannot be used in complex semi-structured data because the field values also contain delimiters. In this case, the regular expression can be used to separate the fields.

The regular expression is configured in **Source Job Configuration**. The migration source must be an object storage or file system, and **File Format** must be **CSV**.

Figure 1-12 Setting regular expression parameters



Source Job Configuration

* Source Link Name	obs-dayu-demo
* Bucket Name	abcdsze
* Source Directory/File	/DAS_Imexport_Import_9e14
* File Format	CSV
Hide Advanced Attributes	
Line Separator	
Use Quote Char	Yes [✓ No]
Using RE to separate fields	[✓ Yes] No
Regular Expression	
First Row As Header	Yes [✓ No]
Encode type	UTF-8
Compression Format	NONE
Source File Processing Method	Do Nothing

During the migration of CSV files, CDM can use regular expressions to separate fields and write parsed results to the migration destination. For details about the syntax of the regular expression, refer to the related documents. This section describes the regular expressions of the following log files:

- **Log4J Log**
- **Log4J Audit Log**
- **Tomcat Log**
- **Django Log**

- Apache Server Log

Log4J Log

- Log sample:
```
2018-01-11 08:50:59,001 INFO
[org.apache.sqoop.core.SqoopConfiguration.configureClassLoader(SqoopConfiguration.java:251)]
Adding jars to current classloader from property: org.apache.sqoop.classpath.extra
```
- Regular expression:
```
^(\d.*\d) (\w*)  \[(.*)\] (\w.*).*
```
- Parsing result:

Table 1-2 Log4J log parsing result

Column Number	Example Value
1	2018-01-11 08:50:59,001
2	INFO
3	org.apache.sqoop.core.SqoopConfiguration.configureClassLoader(SqoopConfiguration.java:251)
4	Adding jars to current classloader from property: org.apache.sqoop.classpath.extra

Log4J Audit Log

- Log sample:
```
2018-01-11 08:51:06,156 INFO
[org.apache.sqoop.audit.FileAuditLogger.logAuditEvent(FileAuditLogger.java:61)]
user=sqoop.anonymous.user    ip=189.xxx.xxx.75    op=show    obj=version    objId=x
```
- Regular expression:
```
^(\d.*\d) (\w*)  \[(.*)\] user=(\w.*)    ip=(\w.*)    op=(\w.*)    obj=(\w.*)    objId=(.*).*
```
- Parsing result:

Table 1-3 Log4J audit log parsing result

Column Number	Example Value
1	2018-01-11 08:51:06,156
2	INFO
3	org.apache.sqoop.audit.FileAuditLogger.logAuditEvent(FileAuditLogger.java:61)
4	sqoop.anonymous.user

Column Number	Example Value
5	189.xxx.xxx.75
6	show
7	version
8	x

Tomcat Log

- Log sample:
```
11-Jan-2018 09:00:06.907 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS
Name:             Linux
```
- Regular expression:
```
^(\d.*\d) (\w*) \[(.*)\] ([\w\.]*) (\w.*).*
```
- Parsing result:

Table 1-4 Tomcat log parsing result

Column Number	Example Value
1	11-Jan-2018 09:00:06.907
2	INFO
3	main
4	org.apache.catalina.startup.VersionLoggerListener.log
5	OS Name:Linux

Django Log

- Log sample:
```
[08/Jan/2018 20:59:07 ] settings     INFO     Welcome to Hue 3.9.0
```
- Regular expression:
```
^\[(.*)\] (\w*)     (\w*)     (.*).*
```
- Parsing result:

Table 1-5 Django log parsing result

Column Number	Example Value
1	08/Jan/2018 20:59:07
2	settings
3	INFO
4	Welcome to Hue 3.9.0

Apache Server Log

- Log sample:
```
[Mon Jan 08 20:43:51.854334 2018] [mpm_event:notice] [pid 36465:tid 140557517657856] AH00489: Apache/2.4.12 (Unix) OpenSSL/1.0.1t configured -- resuming normal operations
```
- Regular expression:
```
^\[(.*)\] \[(.*)\] \[(.*)\] (.*).*
```
- Parsing result:

Table 1-6 Apache server log parsing result

Column Number	Example Value
1	Mon Jan 08 20:43:51.854334 2018
2	mpm_event:notice
3	pid 36465:tid 140557517657856
4	AH00489: Apache/2.4.12 (Unix) OpenSSL/1.0.1t configured -- resuming normal operations

1.9 Recording the Time When Data Is Written to the Database

When you create a job on the CDM console to migrate tables or files of a relational database, you can add a field to record the time when they were written to the database.

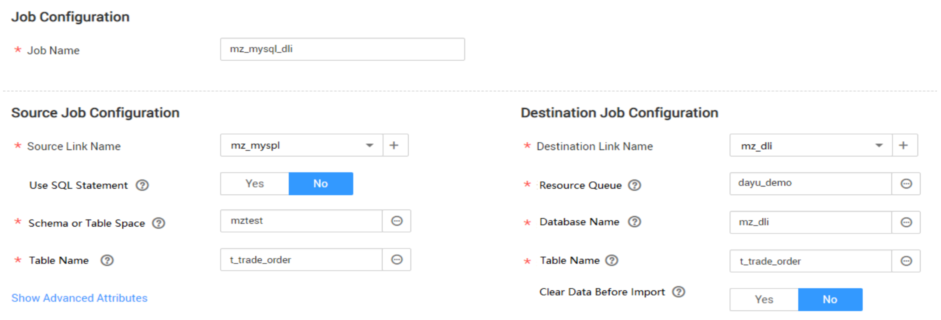
Prerequisites

A link has been created, and the source end of the connector is a relational database.

Creating a Table/File Migration Job

Step 1 Create a table/file migration job, and select the created source connector and destination connector.

Figure 1-13 Configuring the job



Job Configuration

* Job Name	mz_mysql_dli

Source Job Configuration

* Source Link Name	mz_myspl
Use SQL Statement	Yes [✓ No]
* Schema or Table Space	mztest
* Table Name	t_trade_order

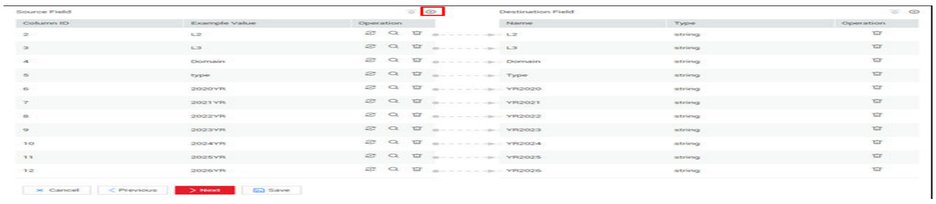
Show Advanced Attributes

Destination Job Configuration

* Destination Link Name	mz_dli
* Resource Queue	dayu_demo
* Database Name	mz_dli
* Table Name	t_trade_order
Clear Data Before Import	Yes [✓ No]

Step 2 Click **Next** to go to the **Map Field** page and click ⊕.

Figure 1-14 Configuring field mapping



Source Field

Column ID	Example Value	Operation
2	L2	
3	L3	
4	Domain	
5	type	
6	2020YR	
7	2021YR	
8	2022YR	
9	2023YR	
10	2024YR	
11	2025YR	
12	2026YR	

Destination Field

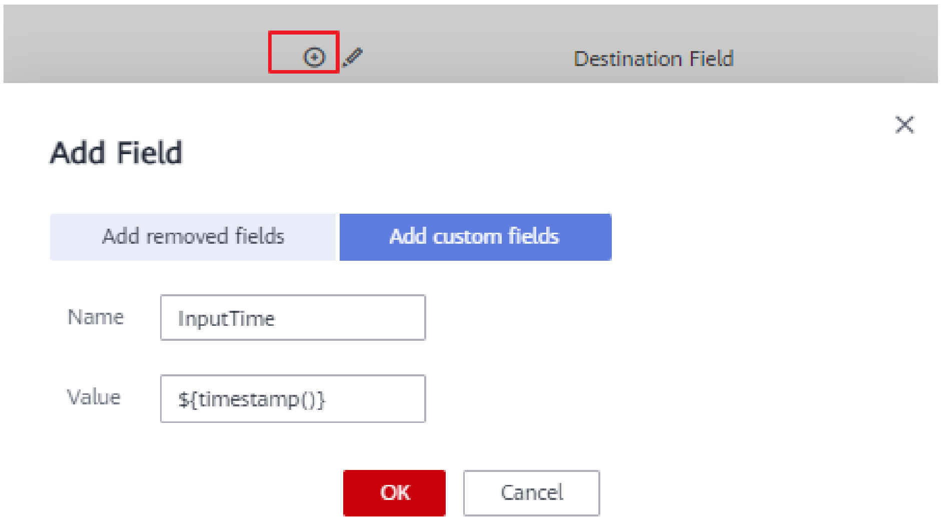
Name	Type	Operation
L2	string	
L3	string	
Domain	string	
Type	string	
YR2020	string	
YR2021	string	
YR2022	string	
YR2023	string	
YR2024	string	
YR2025	string	
YR2026	string	

Step 3 Click the **Custom Fields** tab, set the field name and value, and click **OK**.

Name: Enter **InputTime**.

Value: Enter **${timestamp()}**. For more time macro variables, see **Table 1-7**.

Figure 1-15 Add Field



Add Field

Name	InputTime
Value	${timestamp()}

Table 1-7 Macro variable definition of time and date

Macro Variable	Description	Display Effect
${dateformat(yyyy-MM-dd)}	Returns the current date in **yyyy-MM-dd** format.	2017-10-16
${dateformat(yyyy/MM/dd)}	Returns the current date in **yyyy/MM/dd** format.	2017/10/16
${dateformat(yyyy_MM_dd HH:mm:ss)}	Returns the current time in **yyyy_MM_dd HH:mm:ss** format.	2017_10_16 09:00:00
${dateformat(yyyy-MM-dd HH:mm:ss, -1, DAY)}	Returns the current time in **yyyy-MM-dd HH:mm:ss** format. The date is one day before the current day.	2017-10-15 09:00:00
${timestamp()}	Returns the timestamp of the current time, that is, the number of milliseconds that have elapsed since 00:00:00 on January 1, 1970.	1508115600000
${timestamp(-10, MINUTE)}	Returns the timestamp generated 10 minutes before the current time point.	1508115000000
${timestamp(dateformat(yyyyMMdd))}	Returns the timestamp of 00:00:00 of the current day.	1508083200000
${timestamp(dateformat(yyyyMMdd,-1,DAY))}	Returns the timestamp of 00:00:00 of the previous day.	1507996800000
${timestamp(dateformat(yyyyMMddHH))}	Returns the timestamp of the current hour.	1508115600000

NOTE

- After a field is added, its sample value is not displayed on the console. This does not affect the field value transmission. CDM directly writes the field value to the destination end.
- The **Custom Fields** tab is available only when the source connector is JDBC, HBase, MongoDB, Elasticsearch, or Kafka, or the destination connector is HBase.

Step 4 Click **Next** and set task parameters. Generally, retain the default values of all parameters.

In this step, you can configure the following optional functions:

- **Retry Upon Failure**: If the job fails to be executed, you can determine whether to automatically retry. Retain the default value **Never**.
- **Group**: Select the group to which the job belongs. The default group is **DEFAULT**. On the **Job Management** page, jobs can be displayed, started, or exported by group.
- **Schedule Execution**: If you want the job to be automatically executed at a scheduled time, retain the default value **No**.
- **Concurrent Extractors**: Enter the number of extractors to be concurrently executed. Retain the default value **1**.
- **Write Dirty Data**: Specify this parameter if data that fails to be processed or filtered out during job execution needs to be written to OBS for future viewing. Before writing dirty data, create an OBS link. Retain the default value **No** so that dirty data is not recorded.
- **Delete Job After Completion**: Retain the default value **Do not delete**.

Step 5 Click **Save and Run**. On the **Table/File Migration** page, you can view the job execution progress and result.

Step 6 After the job is successfully executed, in the **Operation** column of the job, click **Historical Record** to view the job's historical execution records and read/write statistics.

On the **Historical Record** page, click **Log** to view the job log.

----End

1.10 File Formats

When creating a CDM job, you need to specify **File Format** in the job parameters of the migration source and destination in some scenarios. This section describes the application scenarios, subparameters, common parameters, and usage examples of the supported file formats.

- **CSV**
- **JSON**
- **Binary**
- **Common parameters**
- **Solutions to File Format Problems**

CSV

To read or write a CSV file, set **File Format** to **CSV**. The CSV format can be used in the following scenarios:

- Import files to a database or NoSQL.
- Export data from a database or NoSQL to files.

After selecting the CSV format, you can also configure the following optional sub-parameters:

1. Line Separator

2. Field Delimiter

3. Encoding Type

4. Use Quote Character

5. Use RE to Separate Fields

6. Use First Row as Header

7. File Size

1. **Line Separator**

   Character used to separate lines in a CSV file. The value can be a single character, multiple characters, or special characters. Special characters can be entered using the URL encoded characters. The following table lists the URL encoded characters of commonly used special characters.

   Table 1-8 URL encoded characters of special characters

Special Character	URL Encoded Character
Space	%20
Tab	%09
%	%25
Enter	%0d
Newline character	%0a
Start of heading\u0001 (SOH)	%01

2. **Field Delimiter**

   Character used to separate columns in a CSV file. The value can be a single character, multiple characters, or special characters. For details, see **Table 1-8**.

3. **Encoding Type**

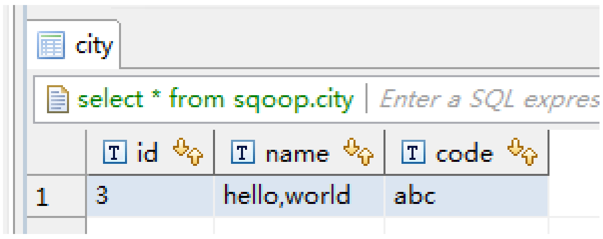
   Encoding type of a CSV file. The default value is **UTF-8**.

   If this parameter is specified at the migration source, the specified encoding type is used to parse the file. If this parameter is specified at the migration destination, the specified encoding type is used to write data to the file.

4. **Use Quote Character**

   – Exporting data from a database or NoSQL to CSV files (configuring **Use Quote Character** at the migration destination): If a field delimiter appears in the character string of a column of data at the migration source, set **Use Quote Character** to **Yes** at the migration destination to quote the character string as a whole and write it into the CSV file. Currently, CDM uses double quotation marks ("") as the quote character only. **Figure 1-16** shows that the value of the **name** field in the database contains a comma (,).

Figure 1-16 Field value containing the field delimiter



	id	name	code
1	3	hello,world	abc

If you do not use the quote character, the exported CSV file is displayed as follows:

```
3.hello,world,abc
```

If you use the quote character, the exported CSV file is displayed as follows:

```
3,"hello,world",abc
```

If the data in the database contains double quotation marks ("") and you set **Use Quote Character** to **Yes**, the quote character in the exported CSV file is displayed as three double quotation marks ("""). For example, if the value of a field is **a"hello,world"c**, the exported data is as follows:

```
"""a"hello,world"c"""
```

– Exporting CSV files to a database or NoSQL (configuring **Use Quote Character** at the migration source): If you want to import the CSV files with quoted values to a database correctly, set **Use Quote Character** to **Yes** at the migration source to write the quoted values as a whole.

5. **Use RE to Separate Fields**

   This function is used to parse complex semi-structured text, such as log files. For details, see **Using Regular Expressions to Separate Semi-structured Text**.

6. **Use First Row as Header**

   This parameter is used when CSV files are exported to other locations. If this parameter is specified at the migration source, CDM uses the first row as the header when extracting data. When the CSV files are transferred, the headers are skipped. The number of rows extracted from the migration source is more than the number of rows written to the migration destination. The log files will output the information that the header is skipped during the migration.

7. **File Size**

   This parameter is used when data is exported from the database to a CSV file. If a table contains a large amount of data, a large CSV file is generated after migration, which is inconvenient to download or view. In this case, you can specify this parameter at the migration destination so that multiple CSV files with the specified size can be generated. The value of this parameter is an integer. The unit is MB.

JSON

The following describes information about the JSON format:

- **JSON Types Supported by CDM**
- **JSON Reference Node**

- **Copying Data from a JSON File**

1. **JSON types supported by CDM: JSON object and JSON array**
   - JSON object: A JSON file contains a single object or multiple objects separated/merged by rows.
     i. The following is a single JSON object:
```
{
    "took" : 190,
    "timed_out" : false,
    "total" : 1000001,
    "max_score" : 1.0
 }
```
     ii. The following are JSON objects separated by rows:
```
{"took" : 188, "timed_out" : false, "total" : 1000003, "max_score" : 1.0 }
{"took" : 189, "timed_out" : false, "total" : 1000004, "max_score" : 1.0 }
```
     iii. The following are merged JSON objects:
```
{
   "took": 190,
   "timed_out": false,
   "total": 1000001,
   "max_score": 1.0
 }
 {
   "took": 191,
   "timed_out": false,
   "total": 1000002,
   "max_score": 1.0
 }
```
   - JSON array: A JSON file is a JSON array consisting of multiple JSON objects.
```
[{
     "took" : 190,
     "timed_out" : false,
     "total" : 1000001,
     "max_score" : 1.0
 },
 {
     "took" : 191,
     "timed_out" : false,
     "total" : 1000001,
     "max_score" : 1.0
 }]
```
2. **JSON Reference Node**

   Root node that records data. The data corresponding to the node is a JSON array. CDM extracts data from the array in the same mode. Use periods (.) to separate multi-layer nested JSON nodes.
3. **Copying Data from a JSON File**
   a. Example 1: Extract data from multiple objects that are separated or merged. A JSON file contains multiple JSON objects. The following gives an example:
```
{
   "took": 190,
   "timed_out": false,
   "total": 1000001,
   "max_score": 1.0
}
{
   "took": 191,
   "timed_out": false,
   "total": 1000002,
   "max_score": 1.0
}
{
   "took": 192,
   "timed_out": false,
   "total": 1000003,
   "max_score": 1.0
}
```

To extract data from the JSON object and write data to the database in the following formats, set **File Format** to **JSON** and **JSON Type** to **JSON object**, and then map fields.

took	timedOut	total	maxScore
190	false	1000001	1.0
191	false	1000002	1.0
192	false	1000003	1.0

b. Example 2: Extract data from the reference node. A JSON file contains a single JSON object, but the valid data is on a data node. The following gives an example:

```
{
   "took": 190,
   "timed_out": false,
   "hits": {
      "total": 1000001,
      "max_score": 1.0,
      "hits":
       [{
         "_id": "650612",
         "_source": {
            "name": "tom",
            "books": ["book1","book2","book3"]
         }
       },
       {
          "_id": "650616",
          "_source": {
             "name": "tom",
             "books": ["book1","book2","book3"]
          }
       },
     {
          "_id": "650618",
          "_source": {
             "name": "tom",
             "books": ["book1","book2","book3"]
          }
       }]
   }
}
```

To write data to the database in the following formats, set **File Format** to **JSON**, **JSON Type** to **JSON object**, and **JSON Reference Node** to **hits.hits**, and then map fields.

ID	SourceName	SourceBooks
650612	tom	["book1","book2","book3"]
650616	tom	["book1","book2","book3"]

ID	SourceName	SourceBooks
650618	tom	["book1","book2","book3"]

c. Example 3: Extract data from the JSON array. A JSON file is a JSON array consisting of multiple JSON objects. The following gives an example:

```
[{
    "took" : 190,
    "timed_out" : false,
    "total" : 1000001,
    "max_score" : 1.0
 },
 {
    "took" : 191,
    "timed_out" : false,
    "total" : 1000002,
    "max_score" : 1.0
 }]
```

To write data to the database in the following formats, set **File Format** to **JSON** and **JSON Type** to **JSON array**, and then map fields.

took	timedOut	total	maxScore
190	false	1000001	1.0
191	false	1000002	1.0

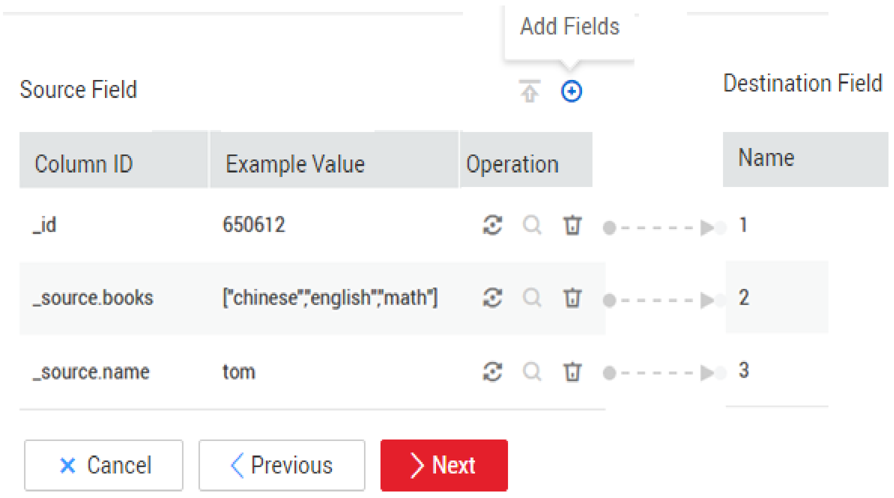
d. Example 4: Configure a converter when parsing the JSON file. On the premise of **example 2**, to add the **hits.max_score** field to all records, that is, to write the data to the database in the following formats, perform the following operations:

ID	SourceName	SourceBooks	MaxScore
650612	tom	["book1","book2","book3"]	1.0
650616	tom	["book1","book2","book3"]	1.0
650618	tom	["book1","book2","book3"]	1.0

Set **File Format** to **JSON**, **JSON Type** to **JSON object**, and **JSON Reference Node** to **hits.hits**, and then create a converter.

i. Click ⊕ to add a field.

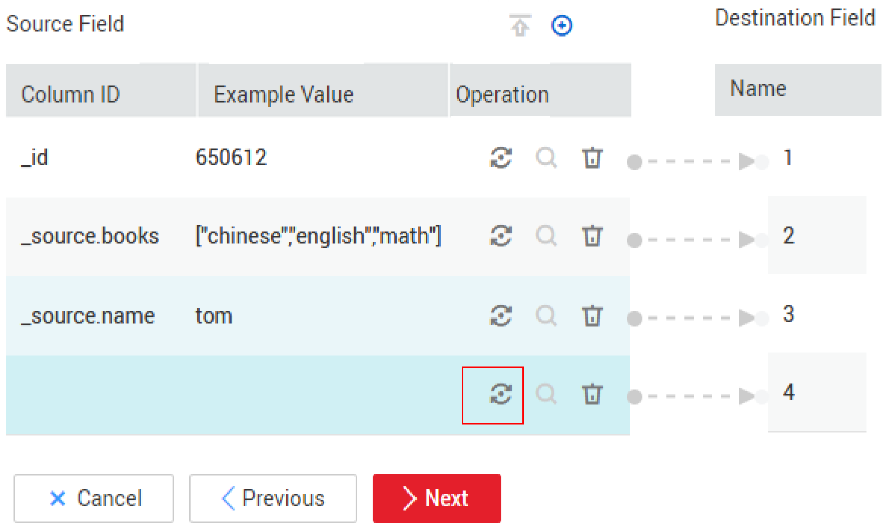
Figure 1-17 Adding a field



Column ID	Example Value	Operation	Name
_id	650612		1
_source.books	["chinese","english","math"]		2
_source.name	tom		3

ii. Click to create a converter for the new field.

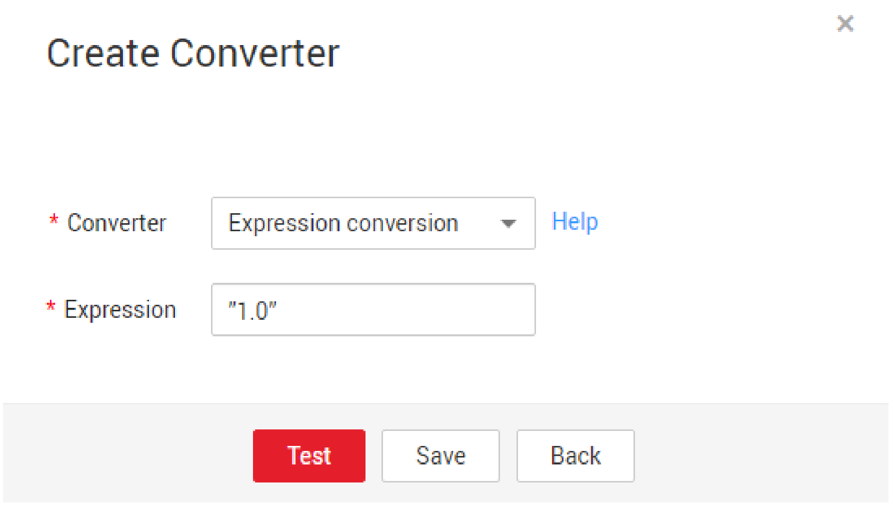
Figure 1-18 Creating a field converter



Column ID	Example Value	Operation	Name
_id	650612		1
_source.books	["chinese","english","math"]		2
_source.name	tom		3
			4

iii. Set **Converter** to **Expression conversion**, enter **"1.0"** in the **Expression** text box, and click **Save**.

Figure 1-19 Configuring a field converter



Create Converter

* Converter	Expression conversion
* Expression	"1.0"

Binary

If you want to copy files between file systems, you can select the binary format. The binary format delivers the optimal rate and performance in file transfer, and does not require field mapping.

- **Directory structure for file transfer**

  CDM can transfer a single file or all files in a directory at a time. After the files are transferred to the migration destination, the directory structure remains unchanged.

- **Migrating incremental files**

  When you use CDM to transfer files in binary format, configure **Duplicate File Processing Method** at the migration destination for incremental file migration. For details, see **Incremental File Migration**.

  During incremental file migration, set **Duplicate File Processing Method** to **Skip**. If new files exist at the migration source or a failure occurs during the migration, run the job again, so that the migrated files will not be migrated repeatedly.

- **Write to Temporary File**

  When migrating files in binary format, you can specify whether to write the files to a temporary file at the migration destination. If this parameter is specified, the file is written to a temporary file during file replication. After the file is successfully migrated, run the **rename** or **move** command to restore the file at the migration destination.

- **Generate MD5 Hash Value**

  An MD5 hash value is generated for each transferred file, and the value is recorded in a new **.md5** file. You can specify the directory where the MD5 value is generated.

Common parameters

- **Source File Processing Method**

After a file is copied successfully, CDM can perform operations on the source file, including renaming the file, deleting the file, and performing no operation on the file.

- **Start Job by Marker File**

  In automation scenarios, a scheduled task is configured on CDM to periodically read files from the migration source. However, files are being generated at the migration source. As a result, CDM reads data repeatedly or fails to read data from the migration source. You can specify the marker file for starting a job as **ok.txt** in the job parameters of the migration source. After the file is successfully generated at the migration source, the **ok.txt** file is generated in the file directory. In this way, CDM can read the complete file.

  In addition, you can set the suspension period. Within the suspension period, CDM periodically queries whether the marker file exists. If the file does not exist after the suspension period expires, the job fails.

  The marker file will not be migrated.

- **Job Success Marker File**

  After data is successfully migrated to a file system, an empty file is generated in the destination directory. You can specify the file name. Generally, this parameter is used together with **Start Job by Marker File**.

  Note that the file cannot be confused with the file to be transferred. For example, if the file to be transferred is **finish.txt** and the job success marker file is set to **finish.txt**, the two files will overwrite each other.

- **Filter**

  When using CDM to migrate files, you can specify a filter to filter files. Files can be filtered by wildcard character or time filter.

  – If you select **Wildcard**, CDM migrates only the paths or files that meet the filter condition.
  – If you select **Time Filter**, CDM migrates only the files modified after the specified time point.

  For example, the **/table/** directory stores a large number of data table directories divided by day. **DRIVING_BEHAVIOR_20180101** to **DRIVING_BEHAVIOR_20180630** store all data of **DRIVING_BEHAVIOR** from January to June. To migrate only the table data of **DRIVING_BEHAVIOR** in March, set **Source Directory/File** to **/table**, **Filter Type** to **Wildcard**, and **Path Filter** to **DRIVING_BEHAVIOR_201803***.

Solutions to File Format Problems

1. When data in a database is exported to a CSV file, if the data contains commas (,), the data in the exported CSV file is disordered.

   The following solutions are available:

   a. Specify a field delimiter.

      Use a character that does not exist in the database or a rare non-printable character as the field delimiter. For example, set **Field Delimiter** at the migration destination to **%01**. In this way, the exported field delimiter is **\u0001**. For details, see Table 1-8.

   b. Use the quote character.

      Set **Use Quote Character** to **Yes** at the migration destination. In this way, if the field in the database contains the field delimiter, CDM quotes the field using the quote character and write the field as a whole to the CSV file.

2. The data in the database contains line separators.

   Scenario: When you use CDM to export a table in the MySQL database (a field value contains the line separator **\n**) to a CSV file, and then use CDM to import the exported CSV file to MRS HBase, data in the exported CSV file is truncated.

   Solution: Specify a line separator.

   When you use CDM to export MySQL table data to a CSV file, set **Line Separator** at the migration destination to **%01** (ensure that the value does not appear in the field value). In this way, the line separator in the exported CSV file is **%01**. Then use CDM to import the CSV file to MRS HBase. Set **Line Separator** at the migration source to **%01**. This avoids data truncation.

2 Advanced Data Development Guidance

2.1 Job Dependency

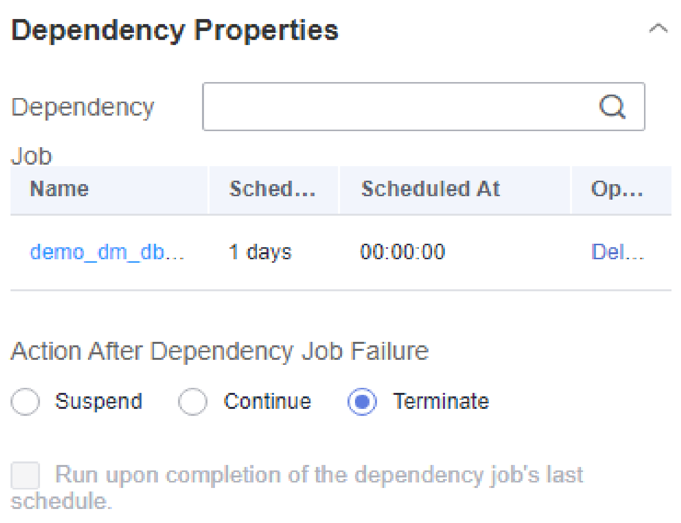
You can set a job that meets the scheduling period conditions as the dependency jobs for a job that is scheduled periodically. For details about how to set a dependency job, see **Setting Up Scheduling for a Job Using the Batch Processing Mode**.

For example, you can set a dependency job (job B) for job A which is scheduled periodically. In this case, job A will be executed only when all the instances of job B are executed successfully within a specified period.

NOTE

- The specified period is calculated as follows (see **How a Job Runs After a Dependency Job Is Set for It** for details):
  - Same-cycle dependency: If the scheduling periods of the two jobs are accurate to the same level (for example, minute, hour, or day), the specified period is **(Execution time of job A – Recurrence of job A, Execution time of job A]**.
  - Cross-cycle dependency: If the scheduling periods of the two jobs are accurate to different levels, the specified period is **[Natural start time of the previous recurrence of job A, Natural start time of the current recurrence of job A)**.
- Parameter **Policy for Current job If Dependency job Fails** determines whether job A will check the status of job B's instances.
  - If this parameter is set to **Suspend** or **Terminate**, job A will be suspended or terminated if instances of job B fail during a specified time period.
  - If this parameter is set to **Continue**, job A will be executed only if all the instances of job B are executed (regardless of whether the execution is successful or not).

Figure 2-1 Job dependency attributes



Dependency Properties

Dependency Job

Name	Sched...	Scheduled At	Op...
demo_dm_db...	1 days	00:00:00	Del...

Action After Dependency Job Failure

Suspend Continue [✓ Terminate]

Run upon completion of the dependency job's last schedule.

This section describes **how to set the conditions of a dependency job** and **how a job runs after a dependency job is set for it**.

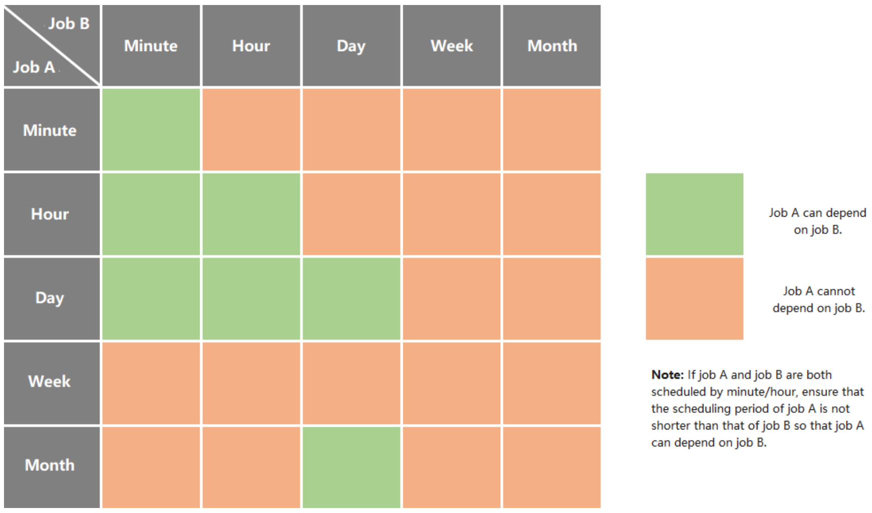
Setting Conditions of a Dependency Job

The recurrence of a periodically scheduled job can be minute, hour, day, week, or month. If job A and job B are both periodically scheduled jobs, and you want to set job B as the dependency job of job A, their recurrences must meet the following requirements:

- The recurrence of job A cannot be shorter than that of job B. For example, if both job A and job B are scheduled by minute or hour and the interval of job A is shorter than that of job B, then job B cannot be set as the dependency job of job A. If job A is scheduled by minute and job B is scheduled by hour, job B cannot be set as the dependency job of job A.
- The recurrence of neither job A nor job B can be week. For example, if the recurrence of job A or job B is week, job B cannot be set as the dependency job of job A.
- A job whose recurrence is month can depend only on a job whose recurrence is day. For example, if the recurrence of job A is month, job B can be set as the dependency job of job A only if job B's recurrence is day.

Figure 2-2 shows the requirements of the recurrences of the jobs that can function as the dependency jobs of other jobs

Figure 2-2 Job dependency



Job A \ Job B	Minute	Hour	Day	Week	Month
Minute	🟩	🟧	🟧	🟧	🟧
Hour	🟩	🟩	🟧	🟧	🟧
Day	🟩	🟩	🟩	🟧	🟧
Week	🟧	🟧	🟧	🟧	🟧
Month	🟧	🟧	🟩	🟧	🟧

🟩	Job A can depend on job B.
🟧	Job A cannot depend on job B.

Note: If job A and job B are both scheduled by minute/hour, ensure that the scheduling period of job A is not shorter than that of job B so that job A can depend on job B.

How a Job Runs After a Dependency Job Is Set for It

It varies depending on whether a job and its dependency job has the same recurrence. In this example, assume that the **Policy for Current job If Dependency job Fails** parameter is set to **Continue**, and job A does not check the running statuses of job B's instances. If this parameter is set to **Suspend** or **Terminate**, job A will also check whether there are failed instances in job B.

- **Same-cycle dependency**: Job A and its dependency job B have the same recurrence, for example, minute, hour, or day.

  After job B is set as the dependency job of job A, job A checks whether instances of job B are running within a specified time range **(Execution time of job A – Recurrence of job A, Execution time of job A)**. Job A will be executed only if all the instances of job B are executed.

  Example 1: Job A depends on job B and they are both scheduled by minute. Job A starts at 10:00 and the interval is 20 minutes. Job B starts at 10:00 and the interval is 10 minutes. The following table lists how the two jobs run.

Table 2-1 Example 1: dependency between jobs with the same recurrence

Time Point	Job B (Starting at 10:00 and Scheduled Every 10 Minutes)	Job A (Starting at 10:00 and Scheduled Every 20 Minutes)
10:00	Executed	Executed after job B's instances are executed in the **(09:40, 10:00]** time period
10:10	Executed	-
10:20	Executed	Executed after job B's instances are executed in the **(10:00, 10:20]** time period
10:30	Executed	-

Time Point	Job B (Starting at 10:00 and Scheduled Every 10 Minutes)	Job A (Starting at 10:00 and Scheduled Every 20 Minutes)
...	...	...

Example 2: Job A depends on job B and they are both scheduled by day. Job A starts at 09:00 on August 1, and job B starts at 10:00 on August 1. The following table lists how the two jobs run.

Table 2-2 Example 2: dependency between jobs with the same recurrence

Time Point	Job B (Starting at 10:00 on August 1 and Scheduled by Day)	Job A (Starting at 09:00 on August 1 and Scheduled by Day)
09:00 on August 1	-	Not executed if no instance of job B is running in the **(09:00 on July 31, 09:00 on August 1]** time period
10:00 on August 1	Executed	-
09:00 on August 2	-	Executed after job B's instances are executed in the **(09:00 on August 1, 09:00 on August 2]** time period
10:00 on August 2	Executed	-
...	...	...

- **Cross-cycle dependency**: Job A and its dependency job B have different recurrences.

  After job B is set as the dependent job of job A, job A checks whether any instance of job B is running in the time range **(Natural start time of the previous recurrence of job A, Natural start time of the current recurrence of job A)**. Job A will be executed only after all the instances of job B are executed.

NOTE

The natural start time of a recurrence is defined as follows:

- If the recurrence is hour, the **natural start time of the previous recurrence** is 00:00 of the previous hour, and the **natural start time of the current recurrence** is 00:00 of the current hour.
- If the recurrence is day, the **natural start time of the previous recurrence** is 00:00:00 of the previous day, and the **natural start time of the current recurrence** is 00:00:00 of the current day.
- If the recurrence is month, the **natural start time of the previous recurrence** is 00:00:00 on 1st of the previous month, and the **natural start time of the current recurrence** is 00:00:00 on 1st of the current month.

Example 3: Job A depends on job B. Job A is scheduled by day, and job B is scheduled by hour. Job A is executed at 02:00 every day. Job B starts at 00:00 and is executed at an interval of 10 hours. The following table lists how the two jobs run.

Table 2-3 Example 3: dependency between jobs with different recurrences

Time Point	Job B (Starting at 00:00 at an Interval of 10 hours and Scheduled by Hour)	Job A (Scheduled at 02:00 Every Day)
00:00 on the first day	Executed	-
02:00 on the first day	-	Not executed if no instance of job B is running in the **[00:00:00 on day 0, 00:00:00 on day 1)** time period
10:00 on the first day	Executed	-
20:00 on the first day	Executed	-
00:00 on the second day	Executed	-
02:00 on the second day	-	Executed if instances of job B are executed in the **[00:00:00 on day 1, 00:00:00 on day 2)** time period

Time Point	Job B (Starting at 00:00 at an Interval of 10 hours and Scheduled by Hour)	Job A (Scheduled at 02:00 Every Day)
10:00 on the second day	Executed	-
20:00 on the second day	Executed	-
...	...	...

Example 4: Job A depends on job B. Job A is scheduled by month, and job B is scheduled by day. Job A is executed at 02:00 on the first and second days of each month. Job B is executed at 00:00 on August 1. The following table lists how the two jobs run.

Table 2-4 Example 4: dependency between jobs with different recurrences

Time Point	Job B (Scheduled by Day and Executed at 00:00 on August 1)	Job A (Scheduled by Month and Executed at 02:00 on the First and Second Days of Each Month)
00:00 on August 1	Executed	-
02:00 on August 1	-	Not executed if no instance of job B is running in the **[00:00:00 on July 1, 00:00:00 on August 1)** time period
00:00 on August 2	Executed	-
02:00 on August 2	-	Not executed if no instance of job B is running in the **[00:00:00 on July 1, 00:00:00 on August 1)** time period
...	-	...

Time Point	Job B (Scheduled by Day and Executed at 00:00 on August 1)	Job A (Scheduled by Month and Executed at 02:00 on the First and Second Days of Each Month)
00:00 on September 1	Executed	-
02:00 on September 1	-	Executed if instances of job B are executed in the **[00:00:00 on August 1, 00:00:00 on September 1)** time period
00:00 on September 2	Executed	-
02:00 on September 2	-	Executed if instances of job B are executed in the **[00:00:00 on August 1, 00:00:00 on September 1)** time period
...	...	...

2.2 IF Statements

When developing and orchestrating jobs in DataArts Factory, you can use IF statements to determine the branch to execute.

This section describes how to use IF statements in the following scenarios:

- **Determining the IF Statement Branch to Be Executed Based on the Execution Status of the Previous Node**
- **Determining the IF Statement Branch to Be Executed Based on the Execution Result of the Previous Node**
- **Configuring the Policy for Executing a Node with Multiple IF Statements**

IF statements use EL expressions. You can select EL expressions and follow the instruction in this section to develop jobs.

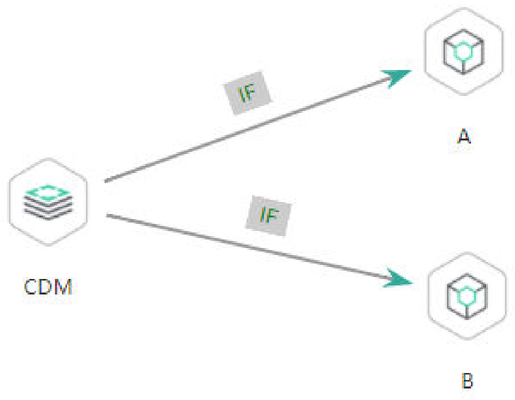
For details about how to use EL expressions, see **EL Expressions**.

Determining the IF Statement Branch to Be Executed Based on the Execution Status of the Previous Node

Scenario

Generally, you can determine the IF statement branch to be executed based on whether the previous CDM node is successfully executed. For details on how to set IF statements, see **Figure 2-3**.

Figure 2-3 Example job



Configuration Method

Step 1 Log in to the DataArts Studio console, locate the target DataArts Studio instance, and click **Access** on the instance card.

Step 2 Click the **Workspaces** tab. In the workspace list, locate the target workspace and click **DataArts Factory**. The DataArts Factory console is displayed.

Step 3 On the **Develop Job** page, create a job, drag a CDM node and two Dummy nodes and drop them on the canvas in the right pane. Click and hold ⊕ to connect the CDM node to the Dummy nodes, as shown in **Figure 2-3**. Set the **Failure Policy** for the CDM node to **Go to the next node**.

Step 4 Right-click the connection line and select **Set Condition**. In the **Edit EL Expression** dialog box, enter the IF statement in the text box.

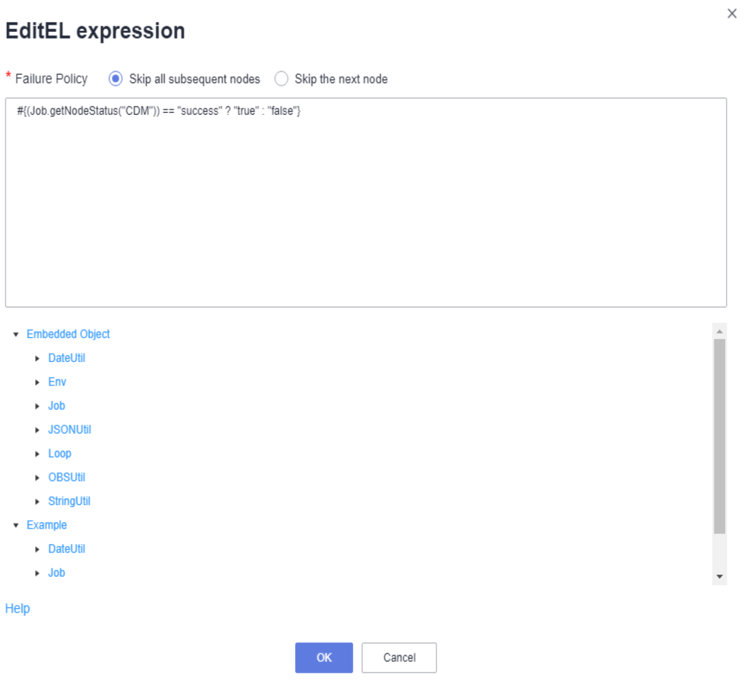
Each statement branch requires an IF statement. The IF statement is a ternary expression based on the EL expression syntax. If the result of the ternary expression is **true**, subsequent nodes will be connected. Otherwise, subsequent nodes will be skipped.

In this demo, the **#{Job.getNodeStatus("node_name")}** EL expression is used to obtain the execution status of a specified node. If the execution is successful, **success** is returned; otherwise, **fail** is returned. In this example, the IF statement expressions are as follows:

- The IF statement expression for branch A is **#{(Job.getNodeStatus("CDM")) == "success" ?**. "true" : "false"}
- The IF statement expression for branch B is **#{(Job.getNodeStatus("CDM")) == "fail" ?**. "true" : "false"}

After entering the IF statement expression, you can select either **Skip all subsequent nodes** or **Skip the next node** for **Failure Policy**. After the configuration is complete, click **OK** to save the job.

Figure 2-4 Configuring a failure policy



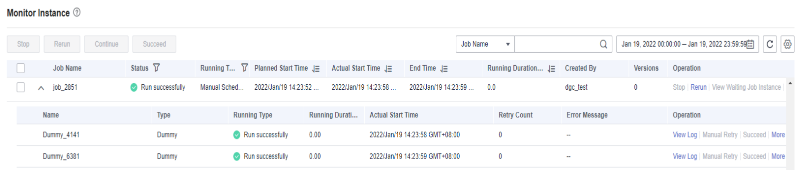
* Failure Policy [✓ Skip all subsequent nodes] Skip the next node

```
#{(Job.getNodeStatus("CDM")) == "success" ? "true" : "false"}
```

Step 5 Click **Test** to test the job and view the execution result on the **Monitor Instance** page.

Step 6 After the job is executed, view the job instance running result on the **Monitor Instance** page. The execution result meets the expectation. If the execution result is **fail**, branch A is skipped and branch B is executed.

Figure 2-5 Job execution result



Job Name	Status	Running T...	Planned Start Time	Actual Start Time	End Time	Running Duration...	Created By	Versions	Operation
job_2851	Run successfully	Manual Sched...	2022/Jan/19 14:23:52 ...	2022/Jan/19 14:23:58 ...	2022/Jan/19 14:23:59 ...	0.0	dgc_test	0	Stop | Rerun | View Waiting Job Instance

Name	Type	Running Type	Running Durati...	Actual Start Time	Retry Count	Error Message	Operation
Dummy_4141	Dummy	Run successfully	0.00	2022/Jan/19 14:23:58 GMT+08:00	0	--	View Log | Manual Retry | Succeed | More
Dummy_6381	Dummy	Run successfully	0.00	2022/Jan/19 14:23:59 GMT+08:00	0	--	View Log | Manual Retry | Succeed | More

----End

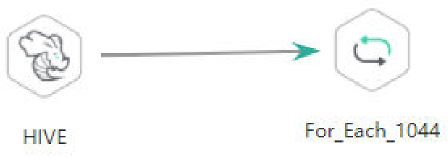
Determining the IF Statement Branch to Be Executed Based on the Execution Result of the Previous Node

Scenario Description

Scenario: Use the execution result of the select statement on the HIVE SQL node as a parameter to determine the IF statement branch to be executed.

The execution result of the select statement on the HIVE SQL node is a two-dimensional array. To obtain the values in the array, use the EL expression **#{Loop.dataArray[][]}**. Currently, only the For Each node supports this expression. Therefore, you need to connect the HIVE SQL node to a For Each node. **Figure 2-6** shows the job orchestration.

Figure 2-6 Example job

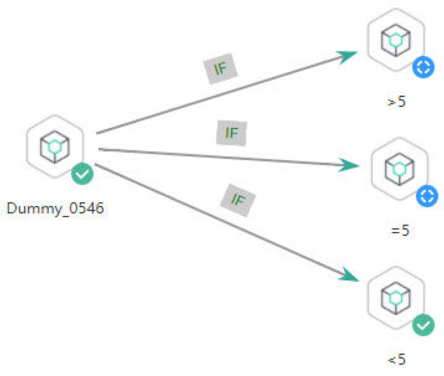


Key configurations of the For Each node are as follows:

- **Dataset**: Enter the execution result of the select statement on the HIVE SQL node. Use the **#{Job.getNodeOutput('HIVE')}** expression, where **HIVE** is the name of the previous node.
- **Subjob Parameter Name**: Enter the parameter defined in the subjob. Transfer the output of the previous node of the main job to the sub-job for use. The variable name is **result**, and its value is a column in the dataset. The EL expression **#{Loop.dataArray[0][0]}** is used.

The sub-job selected on the For Each node determines the IF statement branch to be executed based on the subjob parameter transferred from the For Each node. Figure 2-7 shows the job orchestration.

Figure 2-7 Example sub-job



The IF statement is the key configuration of the subjob. This example uses the expression **${result}** to obtain the value of the job parameter.

NOTE

Do not use the **#{Job.getParam("job_param_name")}** EL expression because this expression can only obtain the values of the parameters configured in the current job, but cannot obtain the parameter values transferred from the parent job or the global variables configured in the workspace. The expression only works for the current job.

To obtain the parameter values passed from the parent job and the global variables configured for the workspace, you are advised to use the **${job_param_name}** expression.

Configuration Method

Developing a Subjob

Step 1 Log in to the DataArts Studio console, locate the target DataArts Studio instance, and click **Access** on the instance card.

Step 2 Click the **Workspaces** tab. In the workspace list, locate the target workspace and click **DataArts Factory**. The DataArts Factory console is displayed.

Step 3 On the **Develop Job** page, create a data development subjob named **foreach**.

Drag four Dummy nodes and drop them on the canvas, click and hold ⊕ to connect them, as shown in **Figure 2-7**.

Step 4 Right-click the connection line and select **Set Condition**. In the **Edit EL Expression** dialog box, enter the IF statement in the text box.

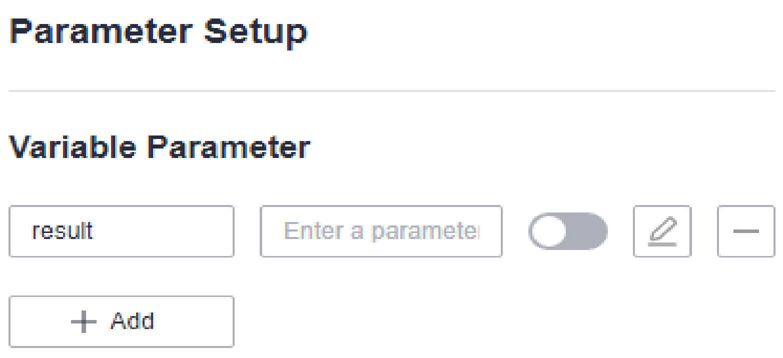
Each statement branch requires an IF statement. The IF statement is a ternary expression based on the EL expression syntax. If the result of the ternary expression is **true**, subsequent nodes will be connected. Otherwise, subsequent nodes will be skipped.

- For the **>5** branch, the IF statement expression is **#{${result} > 5 ? "true" : "false"}**.
- For the **=5** branch, the IF statement expression is **#{${result} == 5 ? "true" : "false"}**.
- For the **<5** branch, the IF statement expression is **#{${result} < 5 ? "true" : "false"}**.

After entering the IF statement expression, you can select either **Skip all subsequent nodes** or **Skip the next node** for **Failure Policy**.

Step 5 Configure job parameters. Set the parameter name to **result**. This parameter is only used by the For Each node in the main job **testif** to identify subjob parameters. You do not need to set the parameter value.

Figure 2-8 Configuring job parameters


Parameter Setup

Variable Parameter

result

Step 6 Save the job.

----End

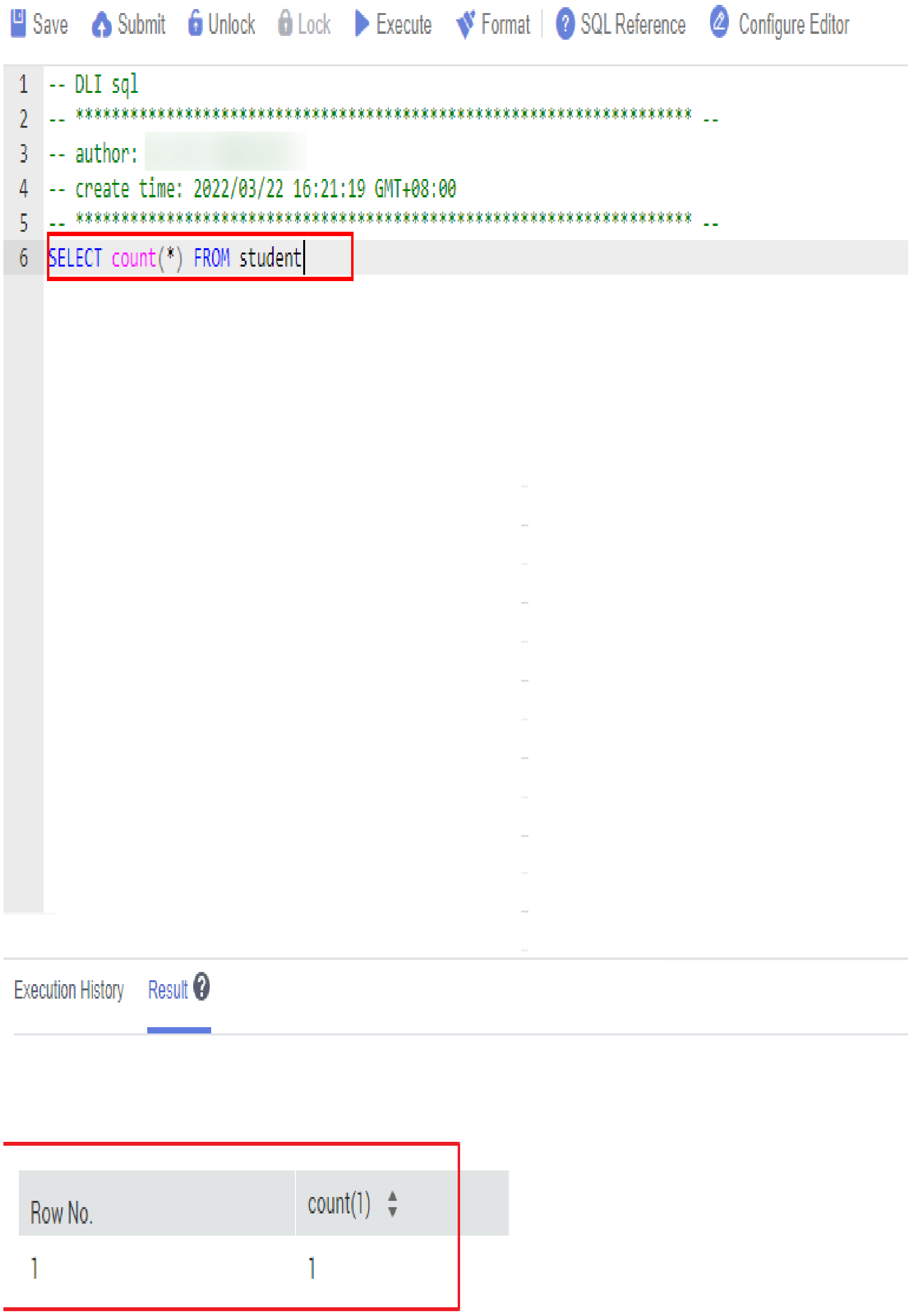
Developing a Job

Step 1 On the **Develop Job** page, create a data development job named **testif**. Drag a HIVE SQL node and a For Each node and drop them on the canvas. Click and hold ⊕ to connect the nodes, as shown in **Figure 2-6**.

Step 2 Configure properties for the HIVE SQL node. Reference the following SQL script (there is no special requirement for other properties):

```
SELECT count(*) FROM student // Count from the student table. The script execution result is a two-dimensional array.
```

Figure 2-9 HIVE SQL script execution result



```
-- DLI sql
-- ******************************************************************** --
-- author:
-- create time: 2022/03/22 16:21:19 GMT+08:00
-- ******************************************************************** --
SELECT count(*) FROM student
```

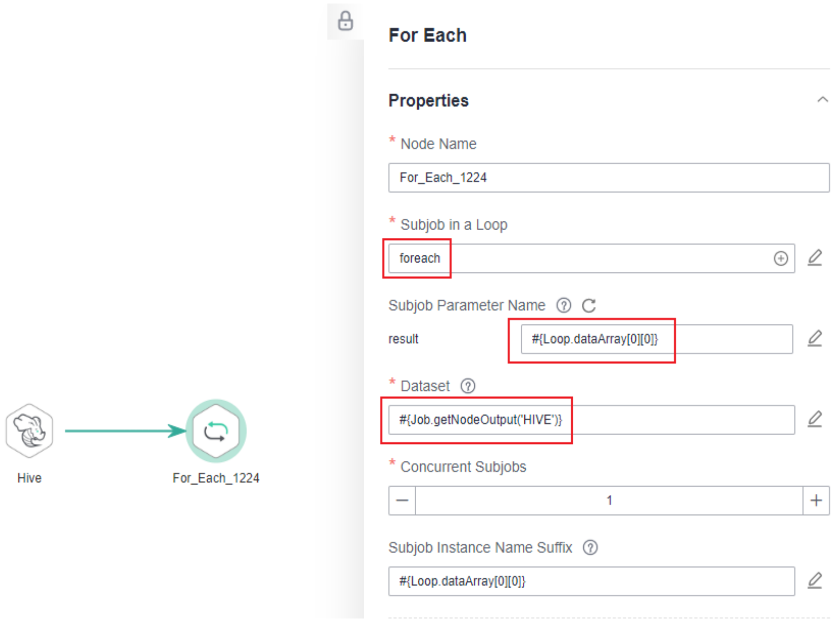
Row No.	count(1)
1	1

Step 3 Configure properties for the For Each node.

- **Subjob in a Loop**: Select **foreach**, the subjob that has been developed.
- **Dataset**: Enter the execution result of the select statement on the HIVE SQL node. Use the **#{Job.getNodeOutput('HIVE')}** expression, where **HIVE** is the name of the previous node.

- **Subjob Parameter Name**: Enter the parameter defined in the subjob. Transfer the output of the previous node of the main job to the sub-job for use. The variable name is **result** (parameter name of the subjob), and its value is a column in the dataset. The EL expression **#{Loop.dataArray[0][0]}** is used.

Figure 2-10 Properties of the For Each node



For Each

Properties

* Node Name: For_Each_1224

* Subjob in a Loop: foreach

Subjob Parameter Name

result: #{Loop.dataArray[0][0]}

* Dataset: #{Job.getNodeOutput('HIVE')}

* Concurrent Subjobs: 1

Subjob Instance Name Suffix: #{Loop.dataArray[0][0]}

Step 4 Save the job.

----End

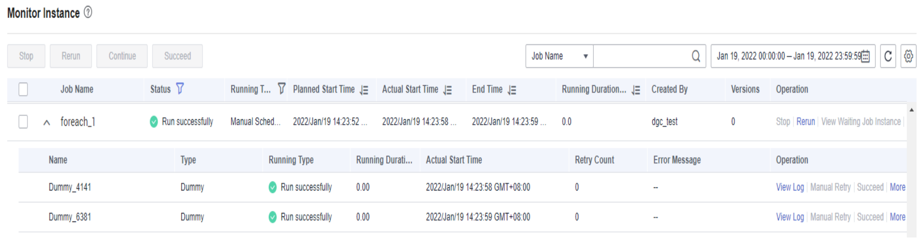
Testing the Main Job

Step 1 Click **Test** above the main job canvas to test the job. After the main job is executed, the subjob is automatically invoked through the For Each node and executed.

Step 2 In the navigation pane on the left, choose **Monitor Instance** to view the job execution result.

Step 3 After the job is executed, view the execution result of the subjob **foreach** on the **Monitor Instance** page. The execution result meets the expectation. Currently, the execution result of the Hive SQL statement is **1**. Therefore, the **>5** and **=5** branches are skipped, and the **<5** branch is successfully executed.

Figure 2-11 Execution result of the subjob



Monitor Instance

Jan 19, 2022 00:00:00 – Jan 19, 2022 23:59:59

Job Name	Status	Running T...	Planned Start Time	Actual Start Time	End Time	Running Duration...	Created By	Versions	Operation
foreach_1	Run successfully	Manual Sched...	2022/Jan/19 14:23:52 ...	2022/Jan/19 14:23:58 ...	2022/Jan/19 14:23:59 ...	0.0	dgc_test	0	Stop | Rerun | View Waiting Job Instance

Name	Type	Running Type	Running Durati...	Actual Start Time	Retry Count	Error Message	Operation
Dummy_4141	Dummy	Run successfully	0.00	2022/Jan/19 14:23:58 GMT+08:00	0	--	View Log | Manual Retry | Succeed | More
Dummy_6381	Dummy	Run successfully	0.00	2022/Jan/19 14:23:59 GMT+08:00	0	--	View Log | Manual Retry | Succeed | More

----End

Configuring the Policy for Executing a Node with Multiple IF Statements

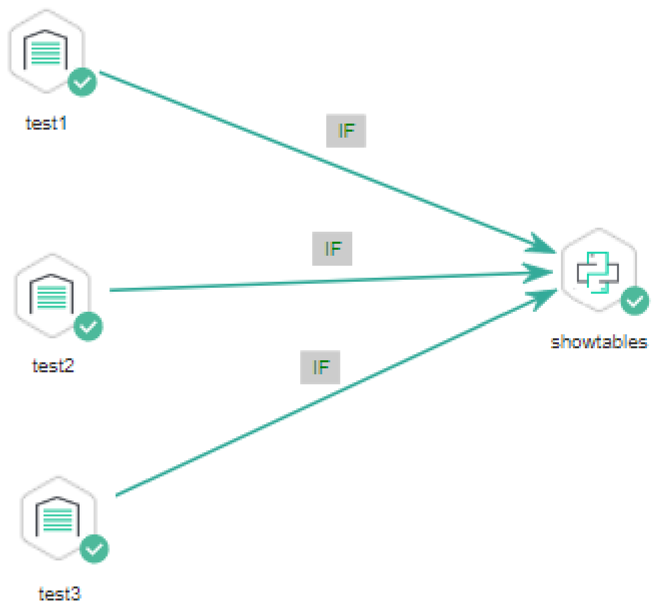
If the execution of a node depends on multiple IF statements, the policy for executing the node can be **AND** or **OR**.

If you choose the **OR** policy, the node will be executed if any one of the IF statements is met.

If you choose the **AND** policy, the node will be executed only if all of the IF statements are met.

If you choose neither, the **OR** policy will be used.

Figure 2-12 A job with multiple IF statements



Configuration Method

Configure the execution policy.

Step 1 Log in to the DataArts Studio console, locate the target DataArts Studio instance, and click **Access** on the instance card.

Step 2 Click the **Workspaces** tab. In the workspace list, locate the target workspace and click **DataArts Factory**. The DataArts Factory console is displayed.

Step 3 On the DataArts Factory console, choose **Configuration** > **Configure** > **Default Configuration**.

Step 4 Select **AND** or **OR** for **Multi-IF Policy**.

Step 5 Click **Save**.

----End

Develop a job.

Step 1 On the **Develop Job** page, create a data development job.

Step 2 Drag three DWS SQL operators as parent nodes and one Python operator as a child node to the canvas. Click and hold ⊕ to connect the nodes to orchestrate the job shown in **Figure 2-12**.

Step 3 Right-click the connection line and select **Set Condition**. In the **Edit EL Expression** dialog box, enter the IF statement in the text box.

Each statement branch requires an IF statement. The IF statement is a ternary expression based on the EL expression syntax.

- The IF statement expression for the test1 node is **#{(Job.getNodeStatus("test1")) == "success" ? "true" : "false"},**
- The IF statement expression for the test2 node is **#{(Job.getNodeStatus("test2")) == "success" ? "true" : "false"},**
- The IF statement expression for the test3 node is **#{(Job.getNodeStatus("test3")) == "success" ? "true" : "false"},**

The expression of each node is determined using the IF statement based on the execution status of the previous node.

After entering the IF statement expression, you can select either **Skip all subsequent nodes** or **Skip the next node** for **Failure Policy**.

----End

Test the job.

Step 1 Click **Save** above the canvas to save the job.

Step 2 Click **Test** above the canvas to test the job.

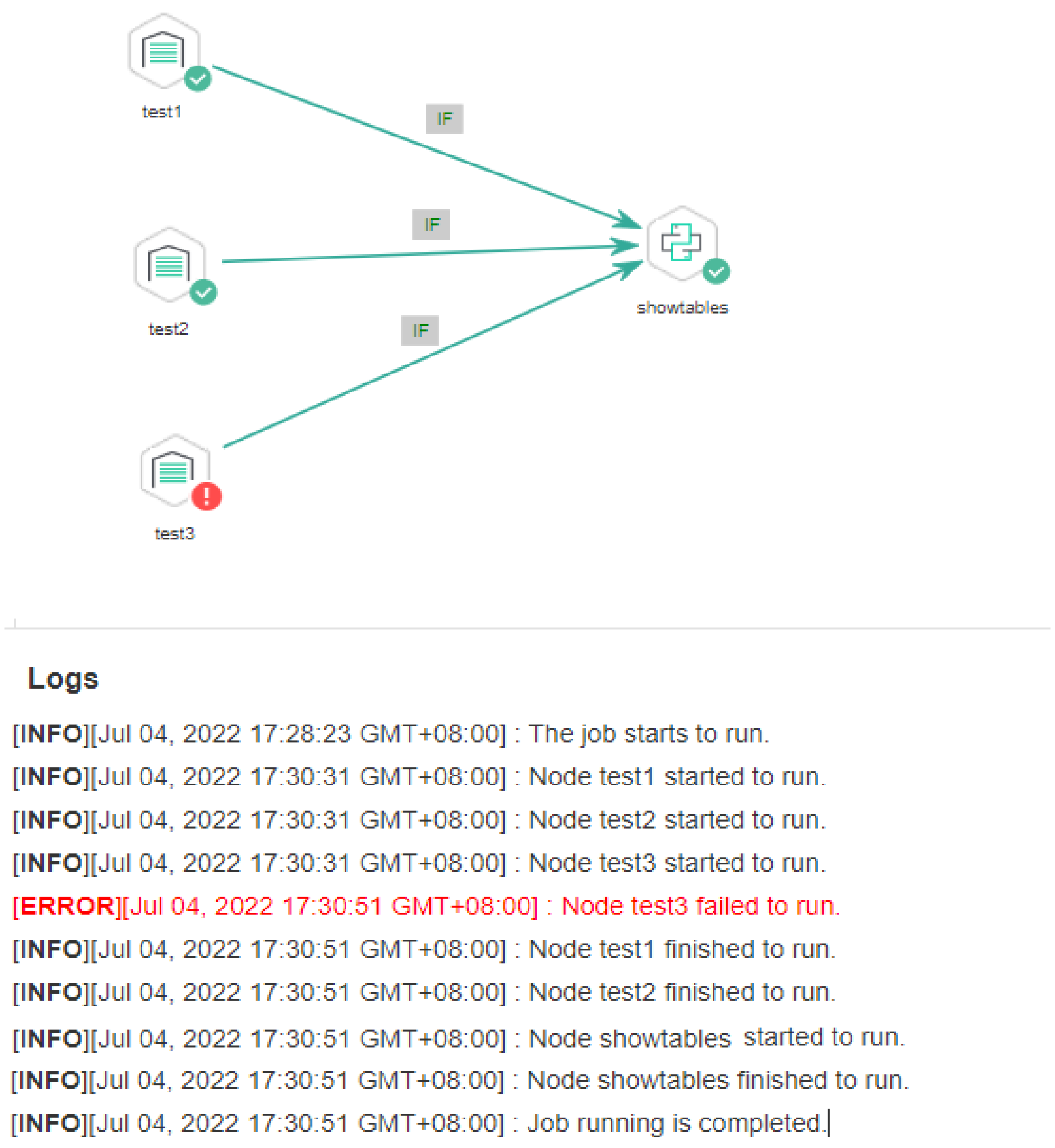
If **test1** is executed successfully, the corresponding IF statement is true.

If **test2** is executed successfully, the corresponding IF statement is true.

If **test3** fails to be executed, the corresponding IF statement is false.

If **Multi-IF Policy** is set to **OR**, the **showtables** node is executed and the job execution is complete.

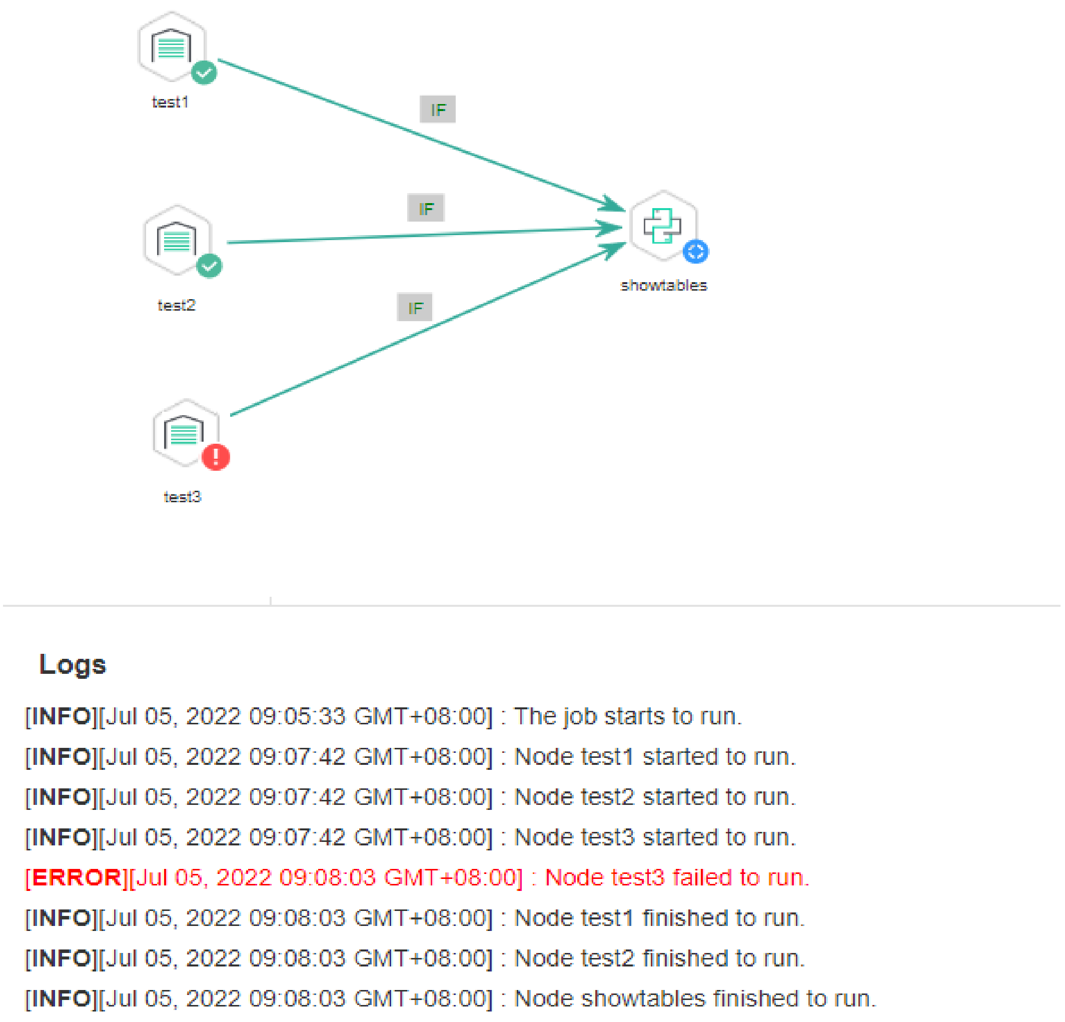
Figure 2-13 How the job runs if Multi-IF Policy is OR



```
[INFO][Jul 04, 2022 17:28:23 GMT+08:00] : The job starts to run.
[INFO][Jul 04, 2022 17:30:31 GMT+08:00] : Node test1 started to run.
[INFO][Jul 04, 2022 17:30:31 GMT+08:00] : Node test2 started to run.
[INFO][Jul 04, 2022 17:30:31 GMT+08:00] : Node test3 started to run.
[ERROR][Jul 04, 2022 17:30:51 GMT+08:00] : Node test3 failed to run.
[INFO][Jul 04, 2022 17:30:51 GMT+08:00] : Node test1 finished to run.
[INFO][Jul 04, 2022 17:30:51 GMT+08:00] : Node test2 finished to run.
[INFO][Jul 04, 2022 17:30:51 GMT+08:00] : Node showtables  started to run.
[INFO][Jul 04, 2022 17:30:51 GMT+08:00] : Node showtables finished to run.
[INFO][Jul 04, 2022 17:30:51 GMT+08:00] : Job running is completed.
```

If **Multi-IF Policy** is set to **AND**, the **showtables** node is skipped and the job execution is complete.

Figure 2-14 How the job runs if Multi-IF Policy is AND



```
[INFO][Jul 05, 2022 09:05:33 GMT+08:00] : The job starts to run.
[INFO][Jul 05, 2022 09:07:42 GMT+08:00] : Node test1 started to run.
[INFO][Jul 05, 2022 09:07:42 GMT+08:00] : Node test2 started to run.
[INFO][Jul 05, 2022 09:07:42 GMT+08:00] : Node test3 started to run.
[ERROR][Jul 05, 2022 09:08:03 GMT+08:00] : Node test3 failed to run.
[INFO][Jul 05, 2022 09:08:03 GMT+08:00] : Node test1 finished to run.
[INFO][Jul 05, 2022 09:08:03 GMT+08:00] : Node test2 finished to run.
[INFO][Jul 05, 2022 09:08:03 GMT+08:00] : Node showtables finished to run.
```

----End

2.3 Obtaining the Return Value of a Rest Client Node

The Rest Client node can execute RESTful requests on HUAWEI CLOUD.

This tutorial describes how to obtain the return value of the Rest Client node, covering the following two application scenarios:

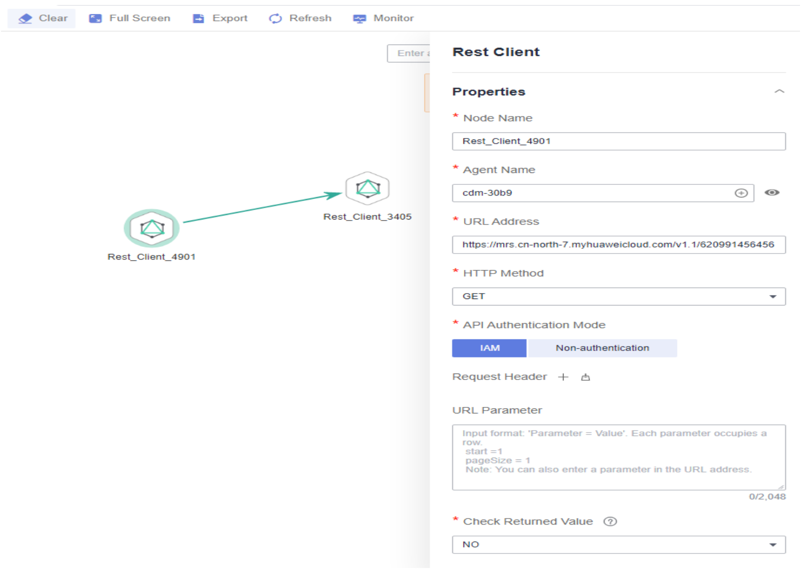
- **Obtaining the Return Value Through Parameter "The response message body parses the transfer parameter"**
- **Obtaining the Return Value Using an EL Expression**

Obtaining the Return Value Through Parameter "The response message body parses the transfer parameter"

As shown in **Figure 2-15**, the first Rest Client node invokes the API of MRS to query the cluster list. **Figure 2-16** shows the JSON message body returned by the API.

- Scenario: The ID of the first cluster in the cluster list needs to be obtained and transferred to other nodes as a parameter.
- Key configurations: Set **The response message body parses the transfer parameter** of the first Rest Client to **clusterId=clusters[0].clusterId**. Other Rest Client nodes can reference the ID of the first cluster in ${clusterId} mode.

Figure 2-15 Rest Client job example 1



Rest Client

Properties

* Node Name: Rest_Client_4901

* Agent Name: cdm-30b9

* URL Address: https://mrs.cn-north-7.myhuaweicloud.com/v1.1/620991456456

* HTTP Method: GET

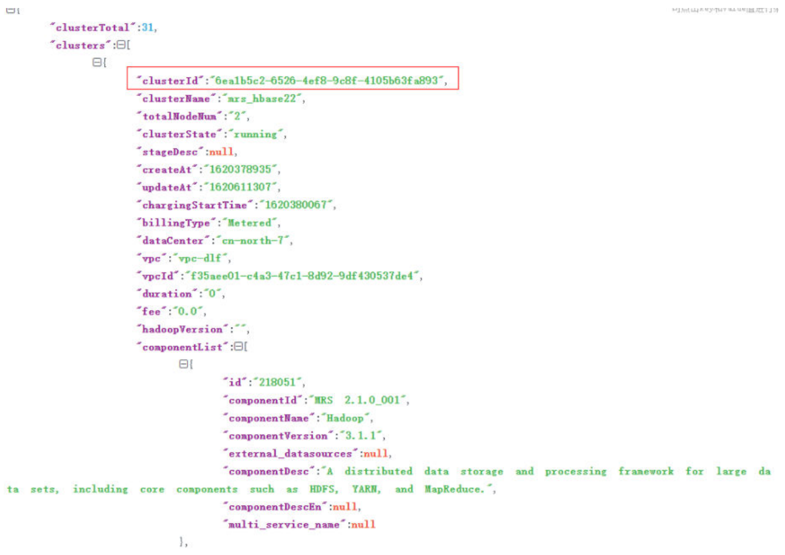
* API Authentication Mode: [✓ IAM] Non-authentication

Request Header

URL Parameter: Input format: 'Parameter = Value'. Each parameter occupies a row.
start =1
pageSize = 1
Note: You can also enter a parameter in the URL address.
0/2,048

* Check Returned Value: NO

Figure 2-16 JSON message body



```
{
    "clusterTotal":31,
    "clusters":[
        {
            "clusterId":"6ea1b5c2-6526-4ef8-9c8f-4105b63fa893",
            "clusterName":"mrs_hbase22",
            "totalNodeNum":"2",
            "clusterState":"running",
            "stageDesc":null,
            "createAt":"1620378935",
            "updateAt":"1620611307",
            "chargingStartTime":"1620380067",
            "billingType":"Metered",
            "dataCenter":"cn-north-7",
            "vpc":"vpc-dlf",
            "vpcId":"f35aee01-c4a3-47c1-8d92-9df430537de4",
            "duration":"0",
            "fee":"0.0",
            "hadoopVersion":"",
            "componentList":[
                {
                    "id":"218051",
                    "componentId":"MRS  2.1.0_001",
                    "componentName":"Hadoop",
                    "componentVersion":"3.1.1",
                    "external_datasources":null,
                    "componentDesc":"A distributed data storage and processing framework for large da
ta sets, including core components such as HDFS, YARN, and MapReduce.",
                    "componentDescEn":null,
                    "multi_service_name":null
                },
```

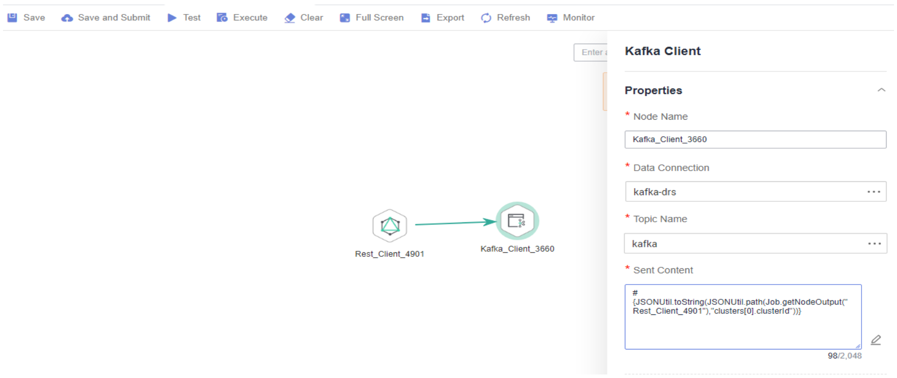
Obtaining the Return Value Using an EL Expression

The Rest Client node can be used together with EL expressions. You can select different EL expressions based on scenarios. This section describes how to develop your own jobs based on your service requirements. For details about how to use EL expressions, see **EL Expressions**.

As shown in **Figure 2-17**, the Rest Client invokes the API of MRS to query the cluster list and then invokes the Kafka Client to send a message.

- Scenario: The Kafka Client sends a character string message. The message content is the ID of the first cluster in the cluster list.
- Key configurations: When you configure the Kafka Client, use the following EL expression to obtain a specific field in the message body returned by the REST API:
  #{JSONUtil.toString(JSONUtil.path(Job.getNodeOutput("Rest_Client_4901"),"clusters[0].clusterId"))}

Figure 2-17 Rest Client job example 2



Kafka Client

Properties

* Node Name

Kafka_Client_3660

* Data Connection

kafka-drs

* Topic Name

kafka

* Sent Content

```
#
{JSONUtil.toString(JSONUtil.path(Job.getNodeOutput("
Rest_Client_4901"),"clusters[0].clusterId"))}
```

98/2,048

2.4 Using For Each Nodes

Scenario

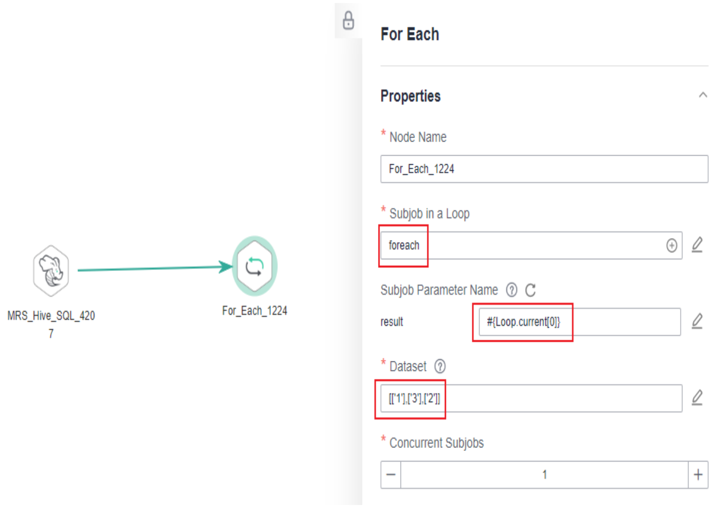
During job development, if some jobs have different parameters but the same processing logic, you can use For Each nodes to avoid repeated job development.

You can use a For Each node to execute a subjob in a loop and use a dataset to replace the parameters in the subjob. The key parameters are as follows:

- **Subjob in a Loop**: Select the subjob to be executed in a loop.
- **Dataset**: Enter a set of parameter values of the subjobs. The value can be a specified dataset such as **[['1'],['3'],['2']]** or an EL expression such as **#{Job.getNodeOutput('preNodeName')}**, which is the output value of the previous node.
- **Subjob Parameter Name**: The parameter name is the variable defined in the subjob. The parameter value is usually set to a group of data in the dataset. Each time the job is run, the parameter value is transferred to the subjob for use. For example, parameter value **#{Loop.current[0]}** indicates that the first value of each group of data in the dataset is traversed and transferred to the subjob.

Figure 2-18 shows an example For Each node. As shown in the figure, the parameter name of the **foreach** subjob is **result**, and the parameter value is the traversal of the one-dimensional array dataset **[['1'],['3'],['2']]** (that is, the value is **1**, **3**, and **2** in the first, second, and third loop, respectively).

Figure 2-18 For Each node



For Each

Properties

* Node Name: For_Each_1224

* Subjob in a Loop: foreach

Subjob Parameter Name: result #{Loop.current[0]}

* Dataset: [['1'],['3'],['2']]

* Concurrent Subjobs: 1

For Each Nodes and EL Expressions

To use For Each nodes properly, you must be familiar with EL expressions. For details about how to use EL expressions, see **EL Expressions**.

For Each nodes use the following EL expressions most:

- #{Loop.dataArray}: dataset input by the For Each node. It is a two-dimensional array.
- #{Loop.current}: The For Loop node processes a dataset line by line. *Loop.current* indicates a line of data that is being processed. *Loop.current* is a one-dimensional array, and its format is #{Loop.current[0]}, #{Loop.current[1]}, or others. The value 0 indicates that the first value in the current line is traversed.
- #{Loop.offset}: current offset when the For Each node processes the dataset. The value starts from 0.
- #{Job.getNodeOutput('preNodeName')}: obtains the output of the previous node.

Examples

Scenario

To meet data normalization requirements, you need to periodically import data from multiple source DLI tables to the corresponding destination DLI tables, as listed in **Table 1**.

Table 2-5 Tables to be imported

Source Table	Destination Table
a_new	a
b_2	b
c_3	c
d_1	d
c_5	e
b_1	f

If you use SQL nodes to execute import scripts, a large number of scripts and nodes need to be developed, resulting in repeated work. In this case, you can use the For Each operator to perform cyclic jobs to reduce the development workload.

Configuration Method

Step 1 Prepare the source and destination tables. To facilitate subsequent job execution and verification, you need to create a source DLI table and a destination DLI table and insert data into the tables.

1. Create a DLI table. You can create a DLI SQL script on the **DataArts Factory** page and run the following commands to create a DLI table. You can also run the following SQL commands in the SQL editor on the DLI console.

```
/* Create a data table. */
CREATE TABLE a_new (name STRING, score INT) STORED AS PARQUET;
CREATE TABLE b_2 (name STRING, score INT) STORED AS PARQUET;
CREATE TABLE c_3 (name STRING, score INT) STORED AS PARQUET;
CREATE TABLE d_1 (name STRING, score INT) STORED AS PARQUET;
CREATE TABLE c_5 (name STRING, score INT) STORED AS PARQUET;
CREATE TABLE b_1 (name STRING, score INT) STORED AS PARQUET;
CREATE TABLE a (name STRING, score INT) STORED AS PARQUET;
CREATE TABLE b (name STRING, score INT) STORED AS PARQUET;
CREATE TABLE c (name STRING, score INT) STORED AS PARQUET;
CREATE TABLE d (name STRING, score INT) STORED AS PARQUET;
CREATE TABLE e (name STRING, score INT) STORED AS PARQUET;
CREATE TABLE f (name STRING, score INT) STORED AS PARQUET;
```

2. Insert data into the source data table. You can create a DLI SQL script on the **DataArts Factory** page and run the following commands to create a DLI table. You can also run the following SQL commands in the SQL editor on the DLI console.

```
/* Insert data into the source data table. */
INSERT INTO a_new VALUES ('ZHAO','90'),('QIAN','88'),('SUN','93');
INSERT INTO b_2 VALUES ('LI','94'),('ZHOU','85');
INSERT INTO c_3 VALUES ('WU','79');
INSERT INTO d_1 VALUES ('ZHENG','87'),('WANG','97');
INSERT INTO c_5 VALUES ('FENG','83');
INSERT INTO b_1 VALUES ('CEHN','99');
```

Step 2 Prepare dataset data. You can obtain a dataset in any of the following ways:

1. Import the data in **Table 1** into the DLI table and use the result read by the SQL script as the dataset.
2. You can save the data in **Table 1** to a CSV file in the OBS bucket. Then use a DLI SQL or DWS SQL statement to create an OBS foreign table, associate it

with the CSV file, and use the query result of the OBS foreign table as the dataset. For details about how to create a foreign table on DLI, see **OBS Source Stream**. For details about how to create a foreign table on DWS, see **Creating a Foreign Table**.

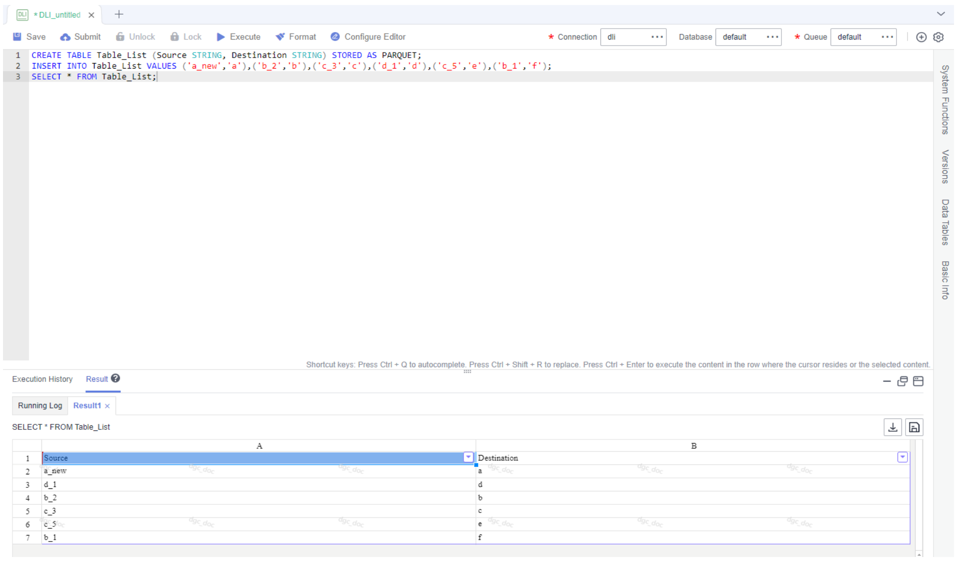
3. You can save the data in **Table 1** to a CSV file in the HDFS. Then use a Hive SQL statement to create a Hive foreign table, associate it with the CSV file, and use the query result of the Hive foreign table as the dataset. For details about how to create a DLI foreign table, see **Creating a Table**.

This section uses method 1 as an example to describe how to import data from **Table 1** to the DLI table (**Table_List**). You can create a DLI SQL script on the **DataArts Factory** page and run the following commands to import data into the table. You can also run the following SQL commands in the SQL editor on the DLI console.

```
/* Create the Table_List data table, insert data in Table 1 into the table, and check the generated data. */
CREATE TABLE Table_List (Source STRING, Destination STRING) STORED AS PARQUET;
INSERT INTO Table_List VALUES ('a_new','a'),('b_2','b'),('c_3','c'),('d_1','d'),('c_5','e'),('b_1','f');
SELECT * FROM Table_List;
```

The generated data in the **Table_List** table is as follows:

Figure 2-19 Data in the Table_List table



```
CREATE TABLE Table_List (Source STRING, Destination STRING) STORED AS PARQUET;
INSERT INTO Table_List VALUES ('a_new','a'),('b_2','b'),('c_3','c'),('d_1','d'),('c_5','e'),('b_1','f');
SELECT * FROM Table_List;
```

Shortcut keys: Press Ctrl + Q to autocomplete. Press Ctrl + Shift + R to replace. Press Ctrl + Enter to execute the content in the row where the cursor resides or the selected content.

SELECT * FROM Table_List

	A	B
1	Source	Destination
2	a_new	a
3	d_1	d
4	b_2	b
5	c_3	c
6	c_5	e
7	b_1	f

Step 3 Create a subjob named **ForeachDemo** to be executed cyclically. In this operation, a task containing the DLI SQL node is defined to be executed cyclically.

1. Access the DataArts Studio **DataArts Factory** page, choose **Develop Job**. Create a job named **ForeachDemo**, select the DLI SQL node, and configure the job as shown in **Figure 2-20**.

   In the DLI SQL statement, set the variable to be replaced to **${}**. The following SQL statement is used to import all data in the **${Source}** table to the **${Destination}** table. **${fromTable}** and **${toTable}** are the variables. The SQL statement is as follows:

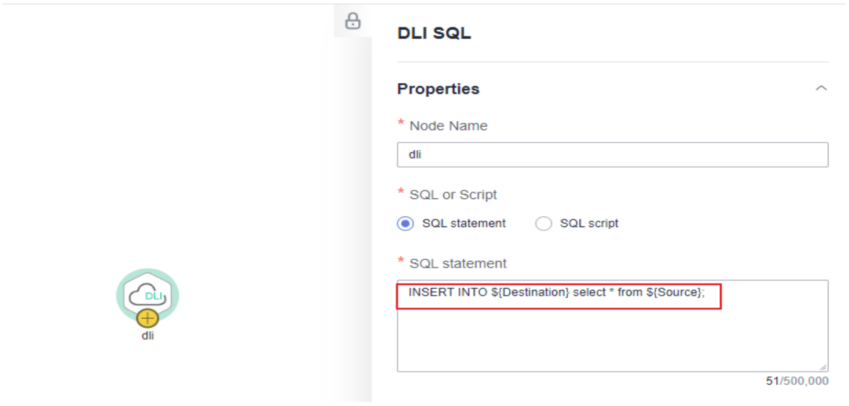
   ```
   INSERT INTO ${Destination} select * from ${Source};
   ```

NOTE

Do not use the **#{Job.getParam("job_param_name")}** EL expression because this expression can only obtain the values of the parameters configured in the current job, but cannot obtain the parameter values transferred from the parent job or the global variables configured in the workspace. The expression only works for the current job.

To obtain the parameter values passed from the parent job and the global variables configured for the workspace, you are advised to use the **${job_param_name}** expression.

Figure 2-20 Cyclically executing a subjob



* Node Name: dli

* SQL or Script: [✓ SQL statement] SQL script

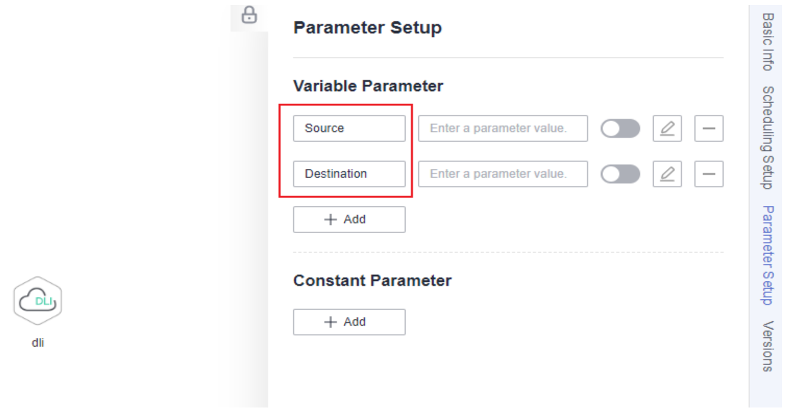
* SQL statement:

```
INSERT INTO ${Destination} select * from ${Source};
```

51/500,000

2. After configuring the SQL statement, configure parameters for the subjob. You only need to set the parameter names, which are used by the For Each operator of the **ForeachDemo_master** job to identify subjob parameters.

Figure 2-21 Configuring subjob parameters



Variable Parameter

Source	Enter a parameter value.
Destination	Enter a parameter value.

+ Add

Constant Parameter

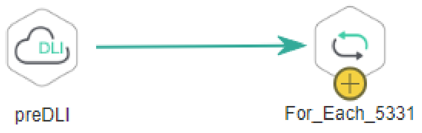
+ Add

3. Save the job.

Step 4 Create a master job named **ForeachDemo_master** where the For Each operator is located.

1. Access the DataArts Studio **DataArts Studio** page and choose **Develop Job**. Create a data development master job named **ForeachDemo_master**. Select the DLI SQL and For Each nodes and click and drag ⊕ to compile the job shown in **Figure 2-22**.

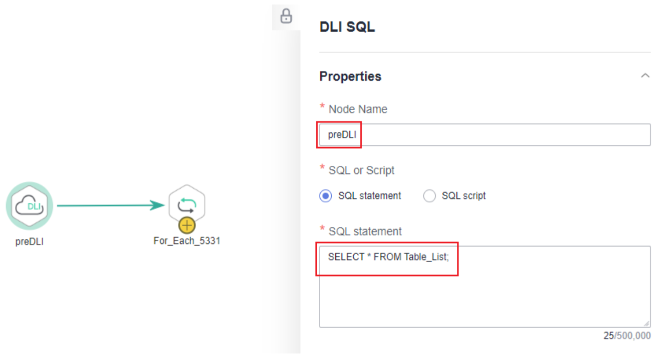
Figure 2-22 Compiling a job



2. Configure the properties of the DLI SQL node. Select **SQL statement** and and enter the following statement. The DLI SQL node reads data from the DLI table **Table_List** and uses it as the dataset.

```
SELECT * FROM Table_List;
```

Figure 2-23 DLI SQL node configuration



DLI SQL

Properties

* Node Name: preDLI

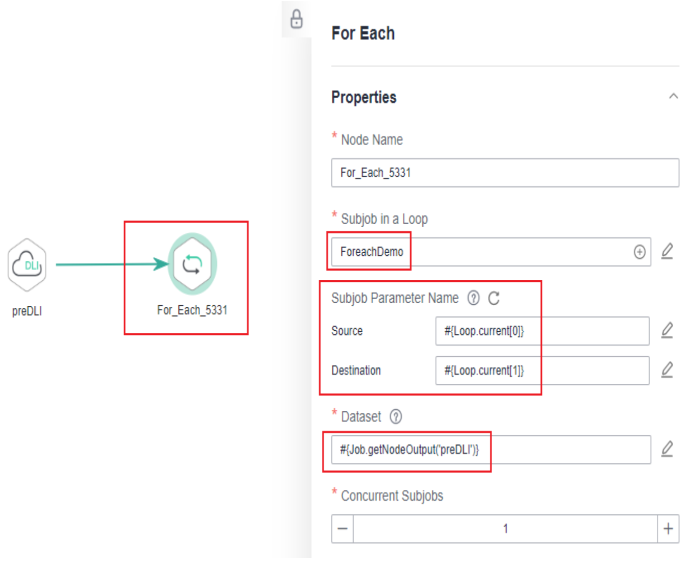
* SQL or Script: [✓ SQL statement] SQL script

* SQL statement: SELECT * FROM Table_List;

25/500,000

3. Configure properties for the For Each node.
   - **Subjob in a Loop**: Select **ForeachDemo**, which is the subjob that has been developed in **step 2**.
   - **Dataset**: Enter the execution result of the select statement on the DLI SQL node. Use the **#{Job.getNodeOutput('preDLI')}** expression, where **preDLI** is the name of the previous node.
   - **Subjob Parameter Name**: used to transfer data in the dataset to the subjob **Source** corresponds to the first column in the **Table_List** table of the dataset, and **Destination** corresponds to the second column. Therefore, enter EL expression **#{Loop.current[0]}** for **Source** and **#{Loop.current[1]}** for **Destination**.

Figure 2-24 Configuring the For Each node



For Each

Properties

* Node Name: For_Each_5331

* Subjob in a Loop: ForeachDemo

Subjob Parameter Name

Source	#{Loop.current[0]}
Destination	#{Loop.current[1]}

* Dataset: #{Job.getNodeOutput('preDLI')}

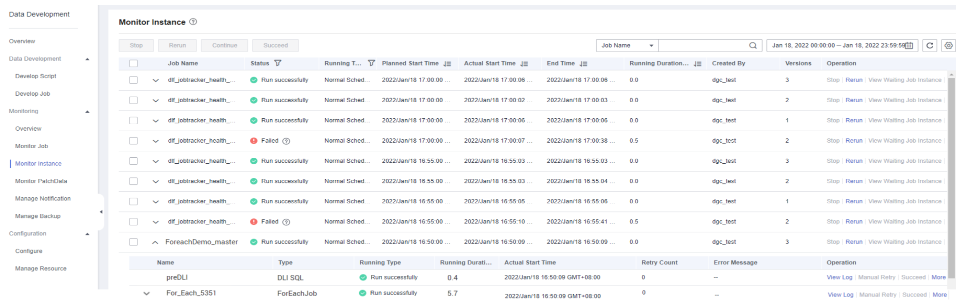
* Concurrent Subjobs: 1

4. Save the job.

Step 5 Test the main job.

1. Click **Test** above the main job canvas to test the job. After the main job is executed, the subjob is automatically invoked through the For Each node and executed.
2. In the navigation pane on the left, choose **Monitor Instance** to view the job execution status. After the job is successfully executed, you can view the subjob instances generated on the For Each node. Because the dataset contains six rows of data, six subjob instances are generated.

Figure 2-25 Viewing job instances



Monitor Instance

Job Name	Status	Running T...	Planned Start Time	Actual Start Time	End Time	Running Duration...	Created By	Versions	Operation
dlf_jobtracker_health_...	Run successfully	Normal Sched...	2022/Jan/18 17:00:00 ...	2022/Jan/18 17:00:06 ...	2022/Jan/18 17:00:06 ...	0.0	dgc_test	3	Stop | Rerun | View Waiting Job Instance
dlf_jobtracker_health_...	Run successfully	Normal Sched...	2022/Jan/18 17:00:00 ...	2022/Jan/18 17:00:02 ...	2022/Jan/18 17:00:03 ...	0.0	dgc_test	2	Stop | Rerun | View Waiting Job Instance
dlf_jobtracker_health_...	Run successfully	Normal Sched...	2022/Jan/18 17:00:00 ...	2022/Jan/18 17:00:06 ...	2022/Jan/18 17:00:06 ...	0.0	dgc_test	1	Stop | Rerun | View Waiting Job Instance
dlf_jobtracker_health_...	Failed	Normal Sched...	2022/Jan/18 17:00:00 ...	2022/Jan/18 17:00:07 ...	2022/Jan/18 17:00:38 ...	0.5	dgc_test	2	Stop | Rerun | View Waiting Job Instance
dlf_jobtracker_health_...	Run successfully	Normal Sched...	2022/Jan/18 16:55:00 ...	2022/Jan/18 16:55:03 ...	2022/Jan/18 16:55:03 ...	0.0	dgc_test	3	Stop | Rerun | View Waiting Job Instance
dlf_jobtracker_health_...	Run successfully	Normal Sched...	2022/Jan/18 16:55:00 ...	2022/Jan/18 16:55:03 ...	2022/Jan/18 16:55:04 ...	0.0	dgc_test	2	Stop | Rerun | View Waiting Job Instance
dlf_jobtracker_health_...	Run successfully	Normal Sched...	2022/Jan/18 16:55:00 ...	2022/Jan/18 16:55:05 ...	2022/Jan/18 16:55:06 ...	0.0	dgc_test	1	Stop | Rerun | View Waiting Job Instance
dlf_jobtracker_health_...	Failed	Normal Sched...	2022/Jan/18 16:55:00 ...	2022/Jan/18 16:55:10 ...	2022/Jan/18 16:55:41 ...	0.5	dgc_test	2	Stop | Rerun | View Waiting Job Instance
ForeachDemo_master	Run successfully	Normal Sched...	2022/Jan/18 16:50:00 ...	2022/Jan/18 16:50:09 ...	2022/Jan/18 16:50:09 ...	0.0	dgc_test	3	Stop | Rerun | View Waiting Job Instance

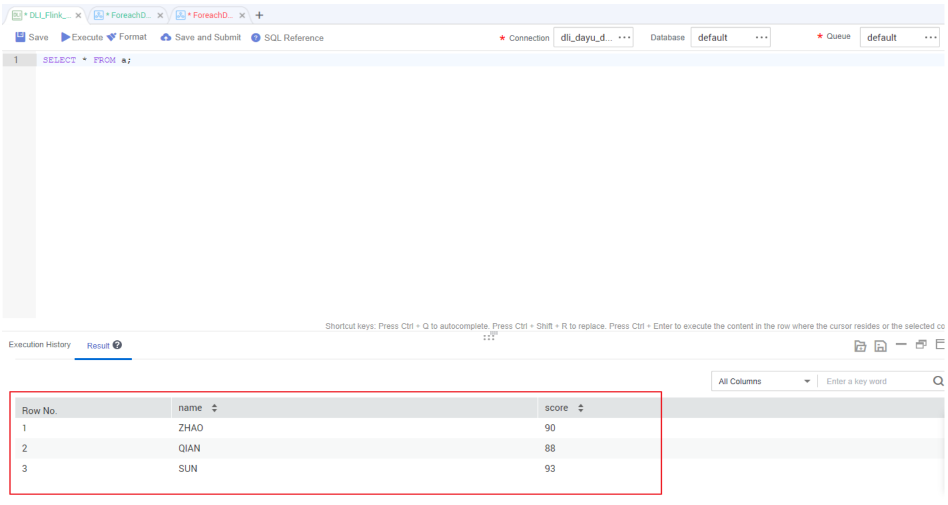
Name	Type	Running Type	Running Durati...	Actual Start Time	Retry Count	Error Message	Operation
preDLI	DLI SQL	Run successfully	0.4	2022/Jan/18 16:50:09 GMT+08:00	0	--	View Log | Manual Retry | Succeed | More
For_Each_5351	ForEachJob	Run successfully	5.7	2022/Jan/18 16:50:09 GMT+08:00	0	--	View Log | Manual Retry | Succeed | More

3. Check whether the data has been inserted into the six DLI destination tables. You can create a DLI SQL script on the **DataArts Factory** page and run the following commands to import data into the table. You can also run the following SQL commands in the SQL editor on the DLI console.

```
/* Run the following command to query the data in a table (table a is used as an example): */
SELECT * FROM a;
```

Compare the obtained data with the data in **Insert data into the source data table**. The inserted data meets the expectation.

Figure 2-26 Destination table data



```
SELECT * FROM a;
```

Shortcut keys: Press Ctrl + Q to autocomplete. Press Ctrl + Shift + R to replace. Press Ctrl + Enter to execute the content in the row where the cursor resides or the selected co

Row No.	name	score
1	ZHAO	90
2	QIAN	88
3	SUN	93

----End

More Cases for Reference

For Each nodes can work with other nodes to implement more functions. You can refer to the following cases to learn more about how to use For Each nodes.

- **Creating Table Migration Jobs in Batches Using CDM Nodes**
- **Determining the IF Statement Branch to Be Executed Based on the Execution Result of the Previous Node**

2.5 Scheduling Jobs Across Workspaces

Scenario

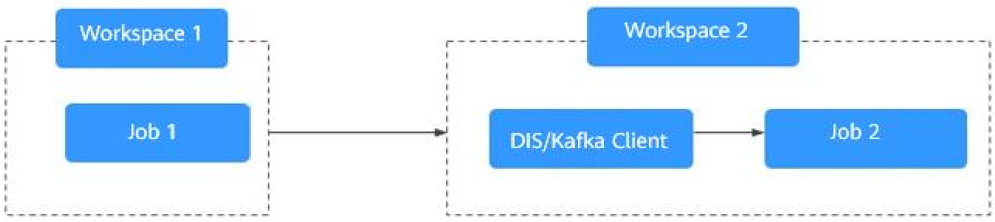
If you have assigned permissions based on workspaces, users in different workspaces can only perform operations on jobs in their own workspaces. However, jobs in different workspaces depend on each other. This tutorial describes how to schedule jobs across workspaces.

Solution

The DataArts Studio DataArts Factory module supports job running triggered by events. Therefore, MRS Kafka can be used as the job dependency to implement cross-workspace job scheduling.

As shown in the following figure, after Job 1 in workspace 1 is executed, you can use the Kafka Client to send a message to trigger Job 2; Configure event-triggered scheduling for Job 2, that is, trigger job 2 based on the messages sent by the Kafka Client.

Figure 2-27 Scheduling solution



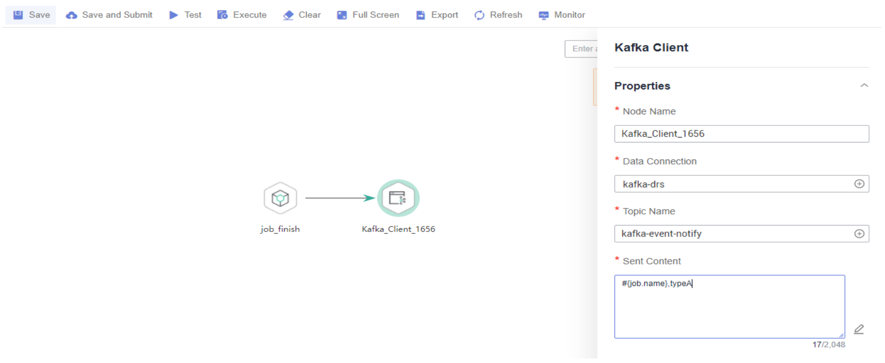
Configuration Method

Step 1 Log in to the DataArts Studio console, locate the target DataArts Studio instance, and click **Access** on the instance card.

Step 2 Locate the row that contains a workspace and click **DataArts Factory** in the **Quick Entry** column. On the displayed page, create a job named **job_test1**. Drag a Dummy and a Kafka Client node and drop them on the canvas, and click and hold ⊕ to connect the nodes, as shown in **Figure 2-28**.

For the Kafka Client node, you need to select a topic and set **Text** to the EL expression **#{job.name},typeA**. After the job in workspace A is executed, Kafka Client will be used to send a single character string message.

Figure 2-28 Creating a job named job_test1 in workspace A



Kafka Client

Properties

* Node Name: Kafka_Client_1656

* Data Connection: kafka-drs

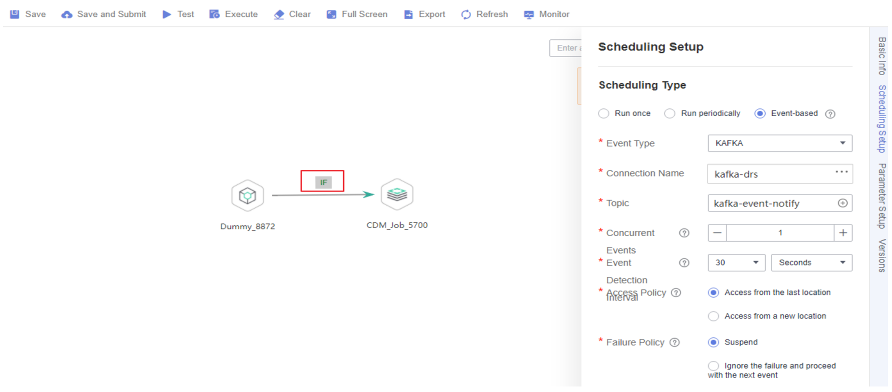
* Topic Name: kafka-event-notify

* Sent Content: #{job.name},typeA

17/2,048

Step 3 In workspace B, create a job named **job_test2**. Drag a Dummy and a CDM Job node and drop them on the canvas, and click and hold ⊕ to connect the nodes, as shown in **Figure 2-29**.

Figure 2-29 Creating a job named job_test2 in workspace B



Scheduling Setup

Scheduling Type

Run once Run periodically [✓ Event-based]

* Event Type: KAFKA

* Connection Name: kafka-drs

* Topic: kafka-event-notify

* Concurrent Events: 1

* Event Detection Interval: 30 Seconds

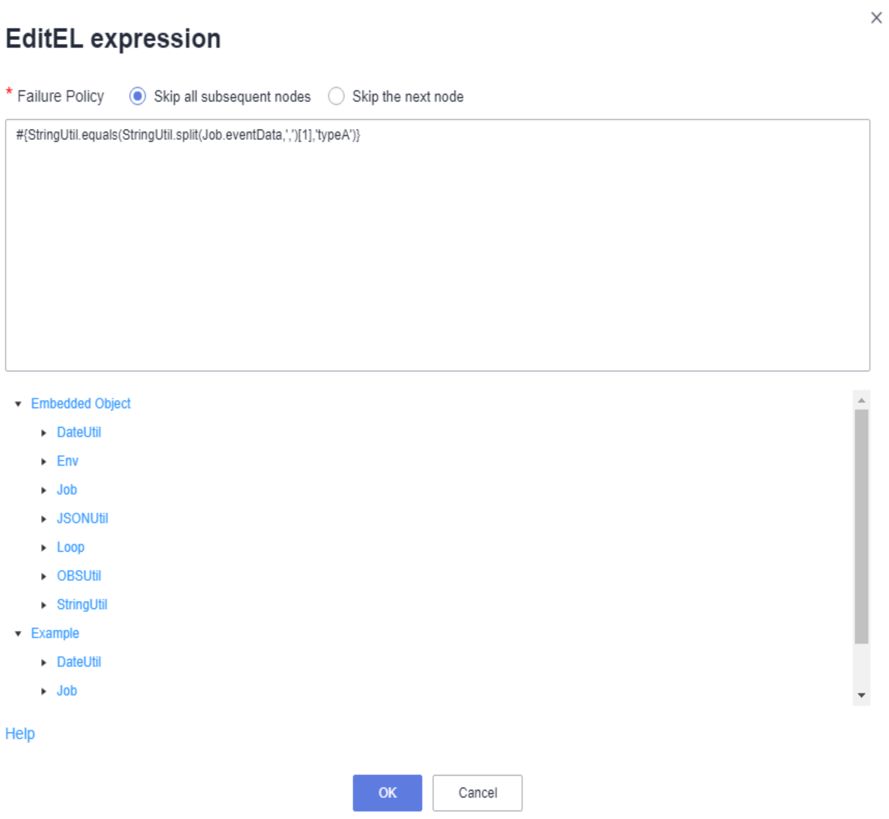
* Access Policy: [✓ Access from the last location] Access from a new location

* Failure Policy: [✓ Suspend] Ignore the failure and proceed with the next event

- Select **Event-based** for **Scheduling Type**, **KAFKA** for **Event Type**, and the topic configured for the Kafka Client node in **job_test1** in workspace A.
- Set the IF judgment condition to ensure that the CDM job node is executed only when a message triggering **job_test1** is received.
  a. Right-click the connection line and select **Set Condition**. In the **Edit EL Expression** dialog box, enter the IF judgment condition in the text box. The IF judgment condition is a ternary expression based on the EL expression syntax. The node following the connection line will be executed only if the result of the ternary expression is **true**. Otherwise, subsequent nodes will be skipped.

     ```
     #{StringUtil.equals(StringUtil.split(Job.eventData,',')[1],'typeA')}
     ```

     This IF judgment condition indicates that subsequent job nodes are executed only if **typeA** follows the comma in the message obtained from Kafka.
  b. Select either **Skip all subsequent nodes** or **Skip the next node** for **Failure Policy**. After the configuration is complete, click **OK** to save the job.



* Failure Policy [✓ Skip all subsequent nodes] Skip the next node

```
#{StringUtil.equals(StringUtil.split(Job.eventData,',')[1],'typeA')}
```

Step 4 Click **Test** to test the job and view the execution result on the **Monitor Instance** page. After **job_test1** in workspace A is tested, **job_test2** in workspace B is triggered.

----End

3 Cross-Workspace DataArts Studio Data Migration

3.1 Overview

Each workspace in a instance contains all the functions. Workspaces are allocated by branch or subsidiary (such as the group, subsidiary, and department), business domain (such as the procurement, production, and sales), or implementation environment (such as the development, test, and production environment). There are no fixed rules.

As your business grows, you may allocate workspaces in a more detailed manner. In this case, you can migrate data from a workspace to another. The data includes data connections in the Management Center, CDM links and jobs, DataArts Architecture tables, DataArts Factory scripts, DataArts Factory jobs, and DataArts Quality jobs.

Preparations

- Create a workspace. You must have the Administrator or Tenant Administrator permission.
- To perform the migration, you must have the developer permission of both workspaces.
- CDM clusters and DataArts DataService Exclusive clusters are isolated from each other. You are advised to prepare corresponding clusters in the target workspace in advance.
- The migration depends on the OBS service. Plan OBS buckets and folders in advance.
- DataArts Studio data migration depends on the backup, import, and export capabilities of each module. You can choose to migrate the data of the module you want.
  - **Management Center Data Migration**
  - **DataArts Migration Data Migration**
  - **DataArts Architecture Data Migration**

– **DataArts Factory Data Migration**
– **DataArts Quality Data Migration**
– **DataArts Catalog Data Migration**
– **DataArts Security Data Migration**
– **DataArts DataService Data Migration**

3.2 Management Center Data Migration

This function depends on the resource migration function of Management Center.

The resources that can be migrated include data services, metadata categories, metadata tags, metadata collection tasks, and the data connections created in Management Center.

Constraints

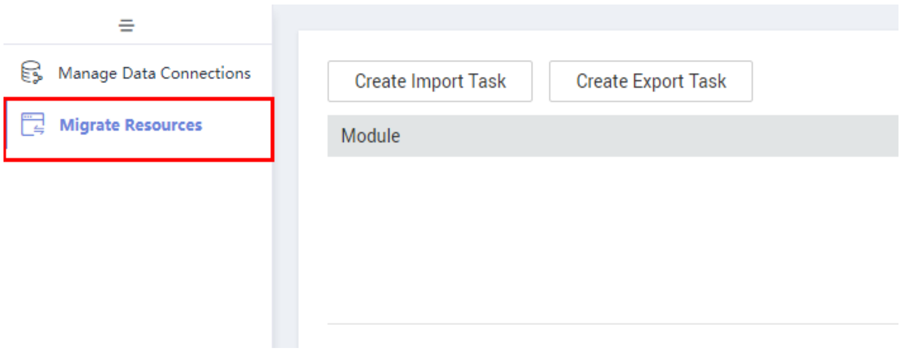
- Resource import and export depend on the OBS service.
- Collection tasks with the same name cannot be migrated repeatedly.
- Categories and tags with the same name cannot be migrated repeatedly.
- Only an exported .zip file can be imported. During the import, the system verifies the resources in the file.
- For security concerns, passwords of connections are not exported when the connections are exported. You need to enter the passwords when importing the connections.

Exporting Resources from the Old Workspace

Log in to the console, access the **Management Center** module of the old workspace, and perform the following operations to export resources:

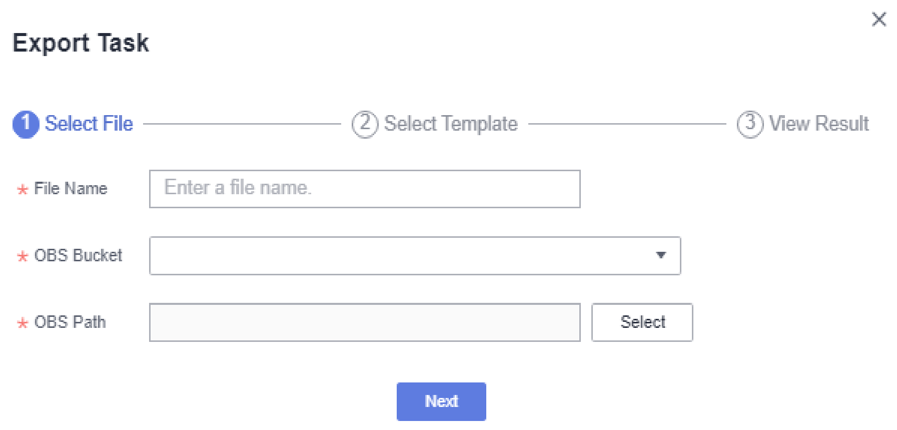
1. In the navigation pane, choose **Migrate Resources**.

Figure 3-1 Migrating Resources



2. Click **Create Export Task** to configure the file name and the OBS path for saving the file.

Figure 3-2 Export Task


Export Task

1 Select File — 2 Select Template — 3 View Result

* File Name: Enter a file name.

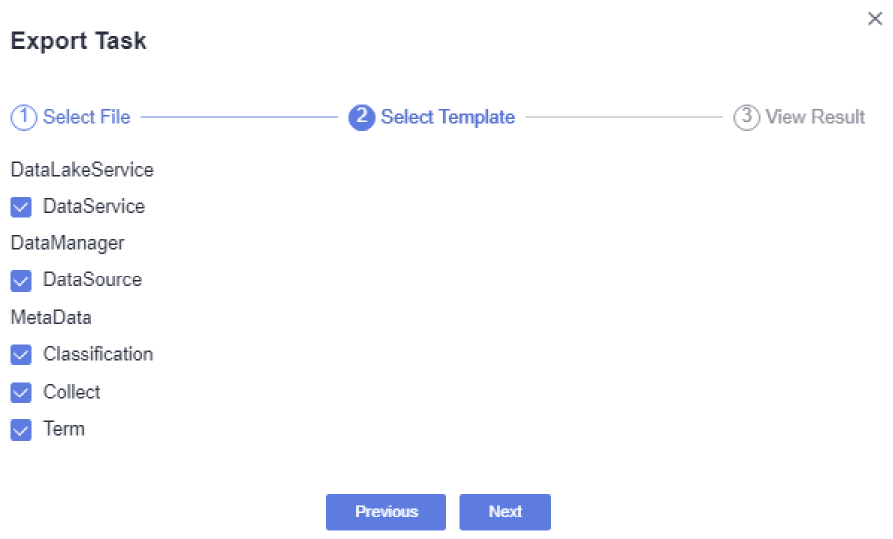
* OBS Bucket:

* OBS Path: Select

Next

3. Click **Next** and select the resource to export.

Figure 3-3 Selecting the resource to export


Export Task

1 Select File — 2 Select Template — 3 View Result

DataLakeService
- [✓] DataService

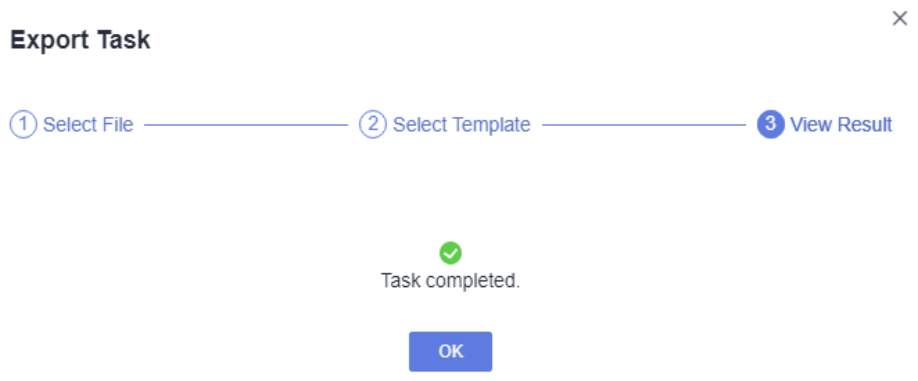
DataManager
- [✓] DataSource

MetaData
- [✓] Classification
- [✓] Collect
- [✓] Term

Previous Next

4. Click **Next** and wait until the export is complete. The resource package is exported to the OBS path set in 2.

Figure 3-4 Export completed


Export Task

1 Select File — 2 Select Template — 3 View Result

Task completed.

OK

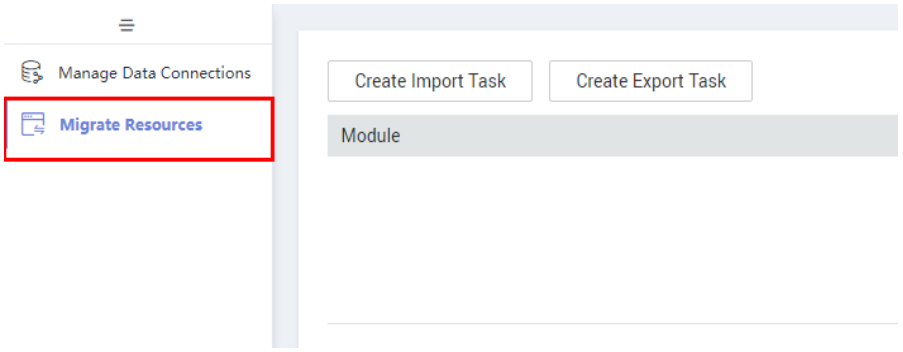
If no result is displayed in 1 minute, the export fails. Try again. If the fault persists, contact customer service or technical support.

Importing Resources to Another Workspace

Log in to the console, access the **Management Center** module of the new workspace, and perform the following operations to import resources:

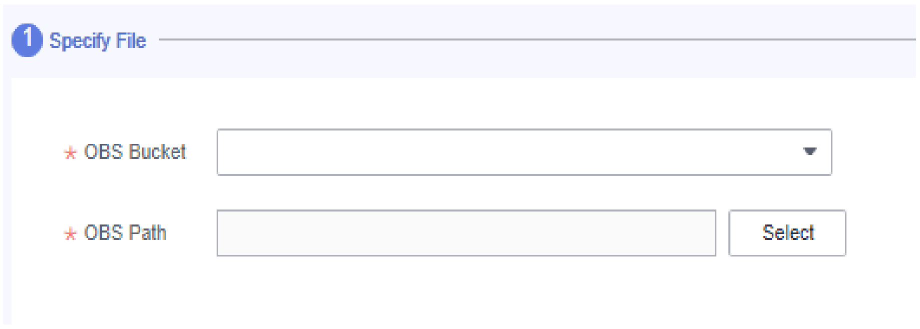
1. In the navigation pane, choose **Migrate Resources**.

Figure 3-5 Migrating Resources



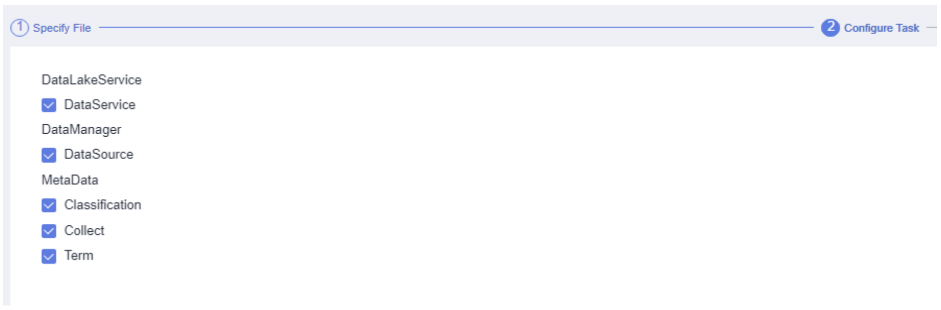
2. Click **Create Import Task** and configure the path for saving the resources to import. Only an exported .zip file can be imported.

Figure 3-6 Configuring the path for saving the resources to import



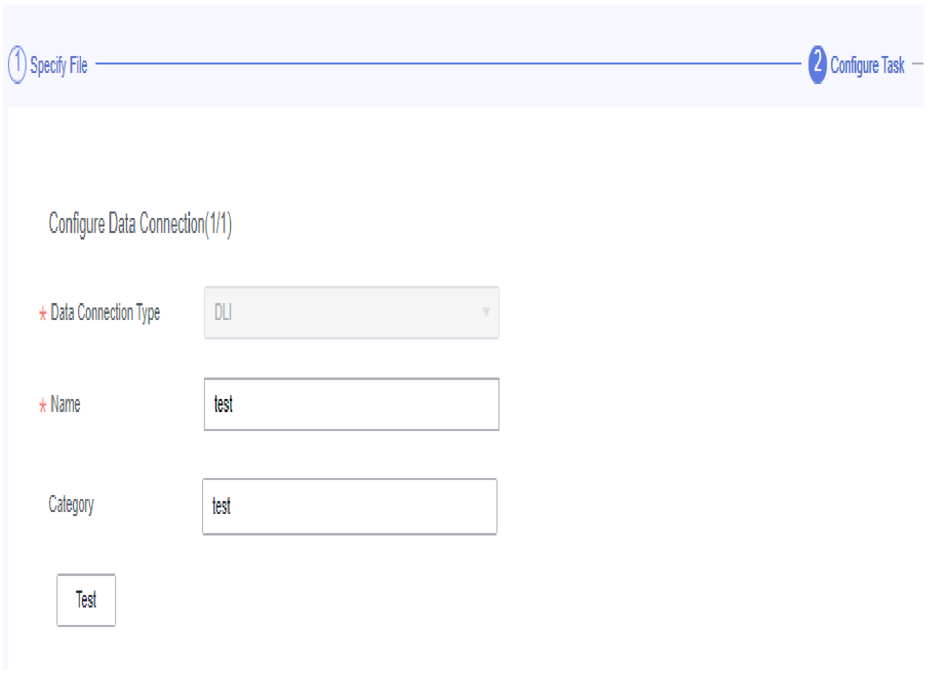
3. Click **Next** and select the resource to import.

Figure 3-7 Selecting the resource to import



4. If you select **DataSource**, click **Next** to configure a data connection. The number of data connections required is determined by the number of data sources. Each data connection requires a password.

Figure 3-8 Configuring a data connection

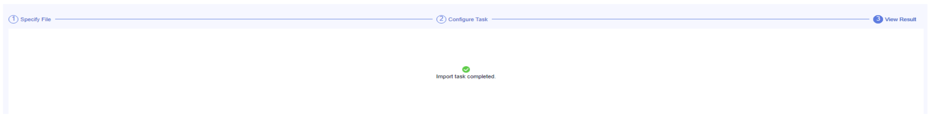


Configure Data Connection(1/1)

* Data Connection Type: DLI
* Name: test
Category: test

5. Click **Next** and wait until the import is complete.

Figure 3-9 Import completed



Import task completed.

If no result is displayed in 1 minute, the import fails. Try again. If the failure persists, contact the customer service or technical support.

Verifying the Migration

After resources are imported to the new workspace, you can check whether the following imported resources are the same as those in the old workspace:

- Data connections in Management Center
- Metadata collection tasks, metadata classifications, and metadata tags in DataArts Catalog
- APIs published in DataArts DataService

3.3 DataArts Migration Data Migration

This function depends on the batch job import and export functions of CDM.

CDM links and jobs can be exported to a local PC.

Constraints

- Data such as cluster configurations and environment variables cannot be imported or exported. If such data needs to be migrated, you need to manually synchronize the data.
- For security concerns, CDM does not export the passwords of the links to data sources. Therefore, you need to add the passwords to the exported JSON file before importing the file or configure the passwords in the import dialog box.
- If you import a JSON file from a local PC, the file cannot be larger than 1 MB.

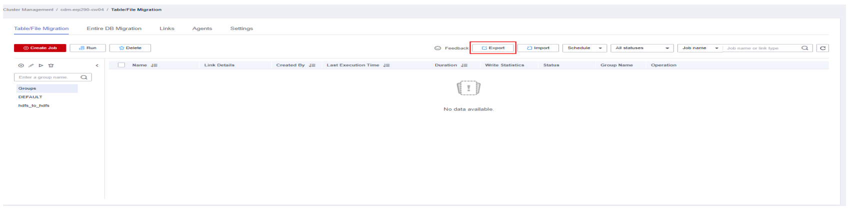
Exporting Jobs and Links from the Old Workspace

Log in to the console, access the DataArts Migration module of the old workspace, and perform the following operations to export jobs and links:

Step 1 In the left navigation pane, choose **Cluster Management**. In the right pane, locate the target cluster and click **Job Management** in the **Operation** column to go to the **Table/File Migration** page.

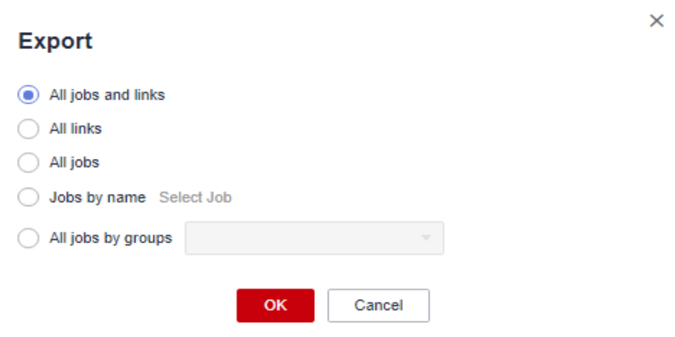
Step 2 Click **Export** above the job list.

Figure 3-10 Exporting jobs and links



Step 3 In the displayed dialog box, select **All jobs and links** and click **OK** to export all jobs and links.

Figure 3-11 Exporting all jobs and links



Step 4 After the export is successful, you can obtain the exported JSON file.

----End

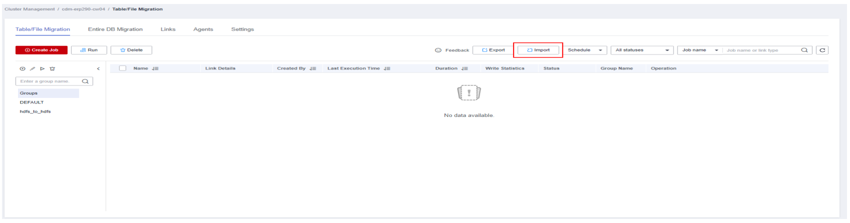
Importing Jobs and Links to the New Workspace

Log in to the console, access the DataArts Migration module of the new workspace, and perform the following operations to import jobs and links:

Step 1 In the left navigation pane, choose **Cluster Management**. In the right pane, locate the target cluster and click **Job Management** in the **Operation** column to go to the **Table/File Migration** page.

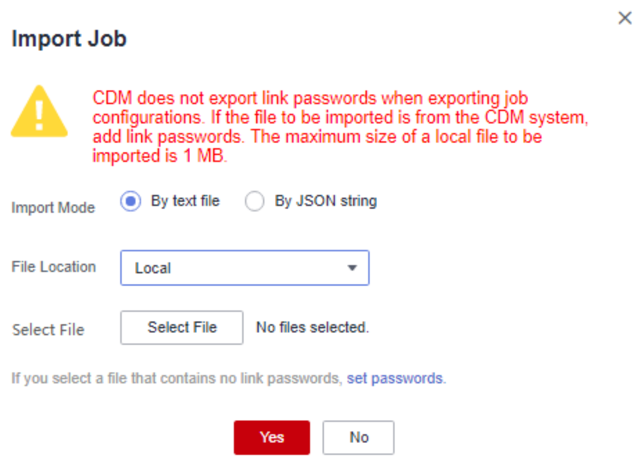
Step 2 Click **Import** above the job list to import the JSON file.

Figure 3-12 Importing jobs and links



Step 3 In the displayed dialog box, select the JSON file exported from the old workspace and upload it.

Figure 3-13 Selecting a JSON file



Import Job

CDM does not export link passwords when exporting job configurations. If the file to be imported is from the CDM system, add link passwords. The maximum size of a local file to be imported is 1 MB.

Import Mode [✓ By text file] By JSON string

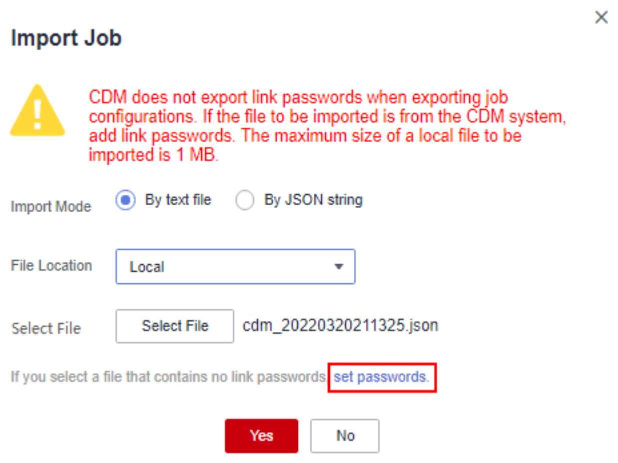
File Location: Local

Select File: Select File No files selected.

If you select a file that contains no link passwords, set passwords.

Step 4 After the JSON file is uploaded, click **set passwords** to set passwords or SKs for data links.

Figure 3-14 Setting passwords



Import Job

CDM does not export link passwords when exporting job configurations. If the file to be imported is from the CDM system, add link passwords. The maximum size of a local file to be imported is 1 MB.

Import Mode [✓ By text file] By JSON string

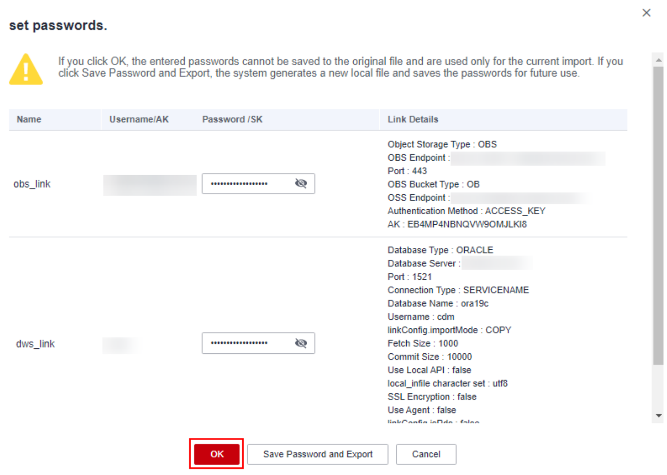
File Location: Local

Select File: Select File cdm_20220320211325.json

If you select a file that contains no link passwords, set passwords.

Step 5 In the displayed dialog box, enter the password or SK of each data link and click **OK** to return to the **Import Job** dialog box.

Figure 3-15 Setting passwords



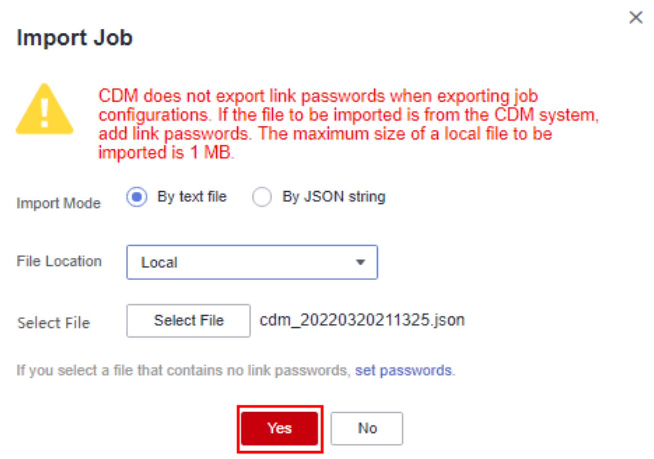
set passwords.

If you click OK, the entered passwords cannot be saved to the original file and are used only for the current import. If you click Save Password and Export, the system generates a new local file and saves the passwords for future use.

Name	Username/AK	Password /SK	Link Details
obs_link	[redacted]	••••••••••••••••••	Object Storage Type : OBS OBS Endpoint : [redacted] Port : 443 OBS Bucket Type : OB OSS Endpoint : [redacted] Authentication Method : ACCESS_KEY AK : EB4MP4NBNQVW9OMJLKI8
dws_link	[redacted]	••••••••••••••••••	Database Type : ORACLE Database Server : [redacted] Port : 1521 Connection Type : SERVICENAME Database Name : ora19c Username : cdm linkConfig.importMode : COPY Fetch Size : 1000 Commit Size : 10000 Use Local API : false local_infile character set : utf8 SSL Encryption : false Use Agent : false

Step 6 Click **OK** to start importing jobs and links.

Figure 3-16 Starting the import



Import Job

CDM does not export link passwords when exporting job configurations. If the file to be imported is from the CDM system, add link passwords. The maximum size of a local file to be imported is 1 MB.

Import Mode: (●) By text file ( ) By JSON string

File Location: Local

Select File: cdm_20220320211325.json

If you select a file that contains no link passwords, set passwords.

Step 7 After the import is complete, the import result is displayed. If the import fails, make changes based on the error message and import the file again.

----End

Verifying the Migration

Check whether the jobs and links imported to the new workspace are consistent with those in the old workspace. If they are consistent, the CDM data migration is successful.

3.4 DataArts Architecture Data Migration

This function depends on the import and export functions of DataArts Architecture.

Constraints

- Before importing tables/entities in ER modeling, and dimensions, fact tables, and summary tables in dimensional modeling, ensure that a data connection has been created in Management Center and is available.
- Time filters, and data in the Review Center and Configuration Center cannot be imported or exported. You must synchronize them manually before migrating other data.
- You can import a file as large as 4 MB and export a maximum of 500 tables at a time.

Exporting Table Data from the Old Workspace

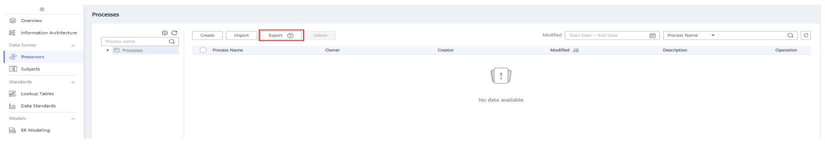
Log in to the console, access the DataArts Architecture module of the old workspace, and perform the following operations to export **processes**, **subjects**, **lookup tables**, **data standards**, **ER modeling tables/entities**, **dimensions/fact tables**, **business metrics**, **technical metrics**, and **summary tables**.

Exporting processes

Step 1 On the **DataArts Architecture** page, choose **Processes** in the left navigation pane.

Step 2 Click **Export** above the list to export all processes. After the export is successful, you can obtain the exported .xlsx file.

Figure 3-17 Exporting processes



Process Name	Owner	Creator	Modified	Description	Operation

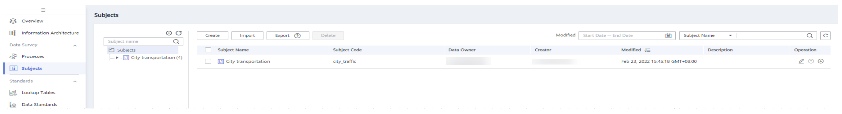
----End

Exporting subjects

Step 1 On the **DataArts Architecture** page, choose **Subjects** in the left navigation pane.

Step 2 Click **Export** above the list to export all subjects. After the export is successful, you can obtain the exported .xlsx file.

Figure 3-18 Exporting subjects



Subject Name	Subject Code	Data Owner	Creator	Modified	Description	Operation
City transportation	city_traffic	[redacted]	[redacted]	Feb 23, 2022 15:45:18 GMT+08:00		

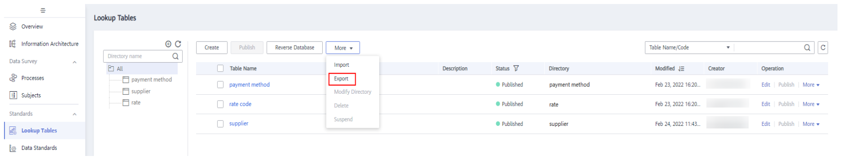
----End

Exporting Lookup Tables

Step 1 On the **DataArts Architecture** page, choose **Lookup Tables** in the left navigation pane.

Step 2 Select the lookup tables to export, click **More** above the list, and select **Export** from the drop-down list box to export the selected lookup tables. After the export is successful, you can obtain the exported .xlsx file.

Figure 3-19 Exporting lookup tables



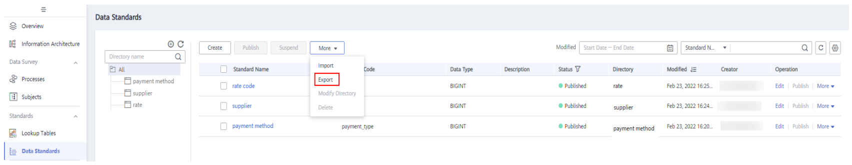
----End

Exporting Data Standards

Step 1 On the **DataArts Architecture** page, choose **Data Standards** in the left navigation pane.

Step 2 Select the data standards to export, click **More** above the list, and select **Export** from the drop-down list box to export the selected data standards. After the export is successful, you can obtain the exported .xlsx file.

Figure 3-20 Exporting data standards



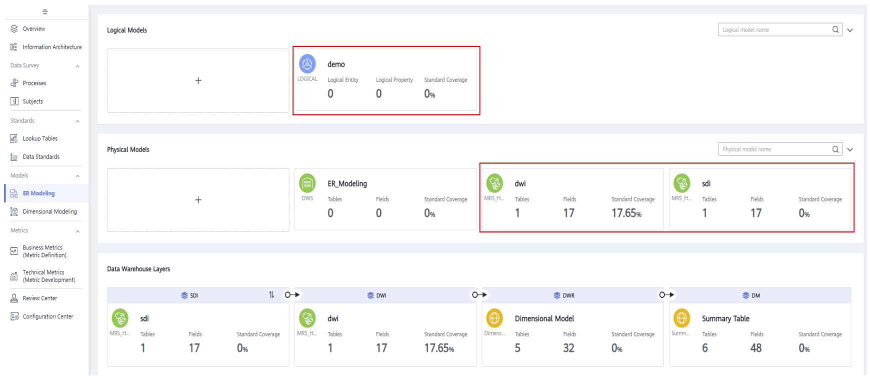
----End

Exporting ER modeling tables/entities

Step 1 On the **DataArts Architecture** page, choose **ER Modeling** in the left navigation pane.

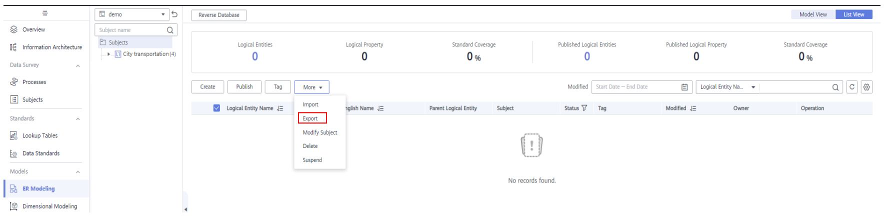
Step 2 Access a logical or physical model and export tables/entities in the model. This section uses logical model **demo** as an example.

Figure 3-21 Accessing the model



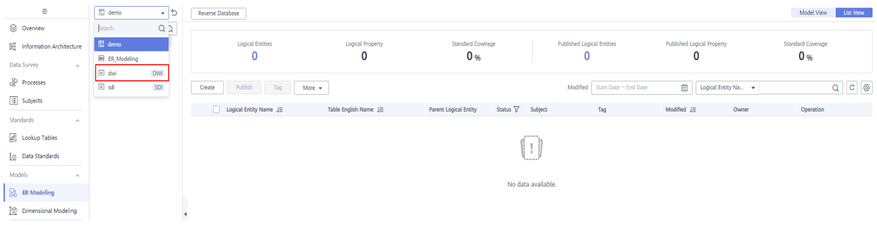
Step 3 Select the tables/entities to export, click **More** above the list, and select **Export** from the drop-down list box (you are advised to select **Table** for **Export**). After the export is successful, you can obtain the exported .xlsx file.

Figure 3-22 Exporting ER modeling tables/entities



Step 4 In the subject list, select other models and repeat **Step 3** to download tables/entities of other models.

Figure 3-23 Exporting tables/entities of other models



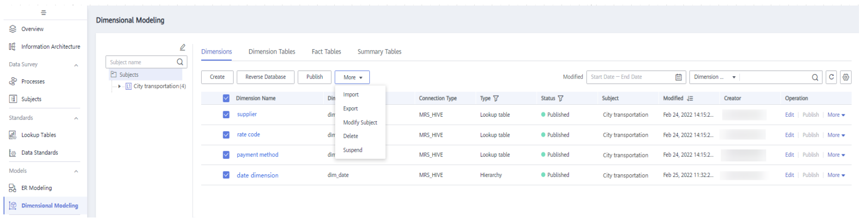
----End

Exporting dimensions/fact tables

Step 1 On the **DataArts Architecture** page, choose **Dimensional Modeling** in the left navigation pane.

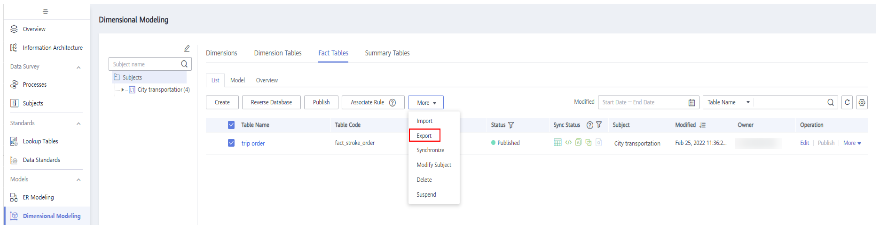
Step 2 On the displayed **Dimensions** page, select the dimensions to export, click **More** above the list, and select **Export** from the drop-down list box. After the export is successful, you can obtain the exported .xlsx file.

Figure 3-24 Exporting dimensions



Step 3 Click the **Fact Tables** tab, select the fact tables to export, click **More** above the list, and select **Export** from the drop-down list box to export the selected fact tables. After the export is successful, you can obtain the exported .xlsx file.

Figure 3-25 Exporting fact tables



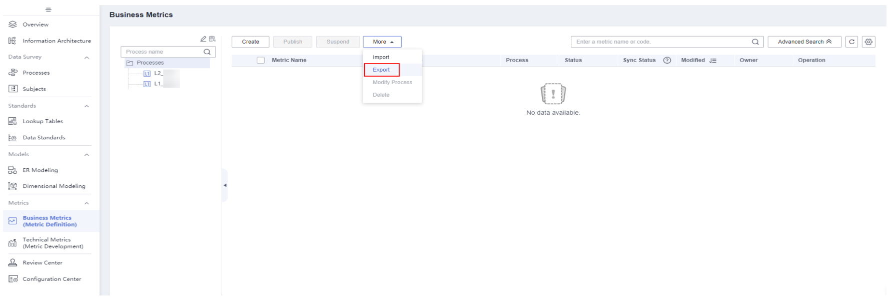
----End

Exporting business metrics

Step 1 On the **DataArts Architecture** page, choose **Business Metrics** in the left navigation pane.

Step 2 Select the business metrics to export, click **More** above the list, and select **Export** from the drop-down list box to export the selected metrics. After the export is successful, you can obtain the exported .xlsx file.

Figure 3-26 Exporting business metrics



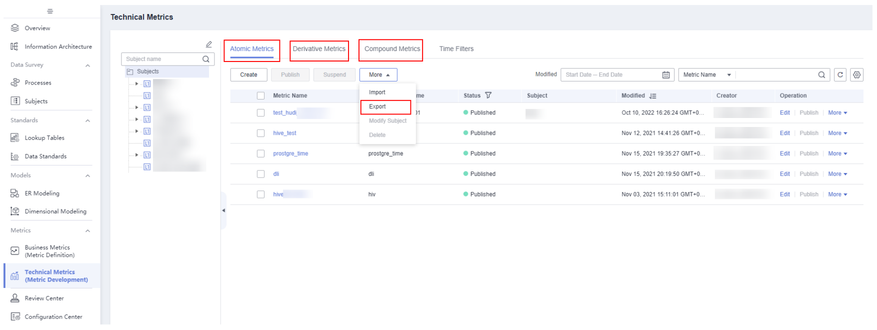
----End

Exporting technical metrics

Step 1 On the **DataArts Architecture** page, choose **Technical Metrics** in the left navigation pane.

Step 2 Click the **Atomic Metrics**, **Derivative Metrics**, and **Composite Metrics** tab, respectively, select the metric to be exported, click **More** in the **Operation** column, and select **Export**. After the export is successful, you can obtain the exported .xlsx file.

Figure 3-27 Exporting technical metrics



Metric Name	Name	Status	Subject	Modified	Creator	Operation
test_hudi		Published		Oct 10, 2022 16:26:24 GMT+0...		Edit Publish More
hive_test		Published		Nov 12, 2021 14:41:26 GMT+0...		Edit Publish More
prostgre_time	prostgre_time	Published		Nov 15, 2021 19:35:27 GMT+0...		Edit Publish More
dli	dli	Published		Nov 15, 2021 20:19:50 GMT+0...		Edit Publish More
hive	hiv	Published		Nov 03, 2021 15:11:01 GMT+0...		Edit Publish More

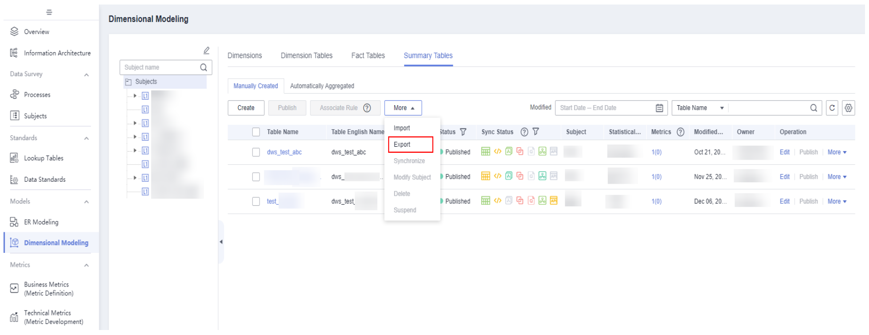
----End

Exporting summary tables

Step 1 On the **DataArts Architecture** page, choose **Dimensional Modeling** in the left navigation pane.

Step 2 Click the **Summary Tables** tab, select the summary tables to export, click **More** above the list, and select **Export** from the drop-down list box to export the selected summary tables. After the export is successful, you can obtain the exported .xlsx file.

Figure 3-28 Exporting summary tables



----End

Importing Table Data to the New Workspace

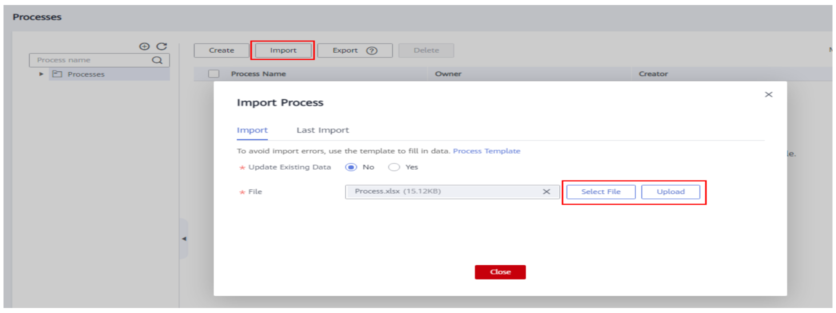
Log in to the console, access the DataArts Architecture module of the new workspace, and perform the following operations to import **processes**, **subjects**, **lookup tables**, **data standards**, **ER modeling tables/entities**, **dimensions/fact tables**, **business metrics**, **technical metrics**, and **summary tables**.

Importing processes

Step 1 On the **DataArts Architecture** page, choose **Processes** in the left navigation pane.

Step 2 Click **Import** above the list. In the displayed dialog box, select and upload the process file to import.

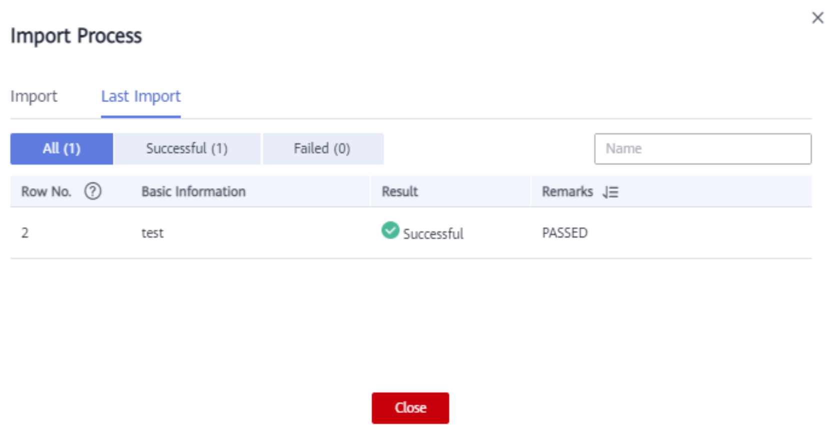
Figure 3-29 Import processes



To avoid import errors, use the template to fill in data. Process Template

Step 3 After the file is uploaded, the system automatically starts importing it. After the file is imported, the import result is displayed.

Figure 3-30 Successful process import


Import Process

Import | Last Import

All (1) | Successful (1) | Failed (0)

Row No.	Basic Information	Result	Remarks
2	test	Successful	PASSED

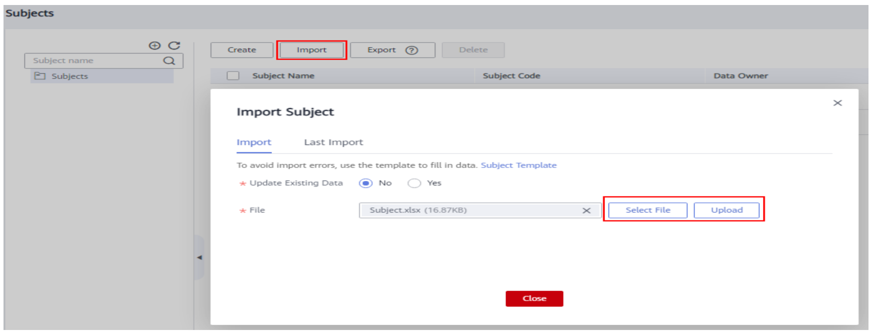
----End

Importing subjects

Step 1 On the **DataArts Architecture** page, choose **Subjects** in the left navigation pane.

Step 2 Click **Import** above the list. In the displayed dialog box, select and upload the subject file to import.

Figure 3-31 Importing subjects



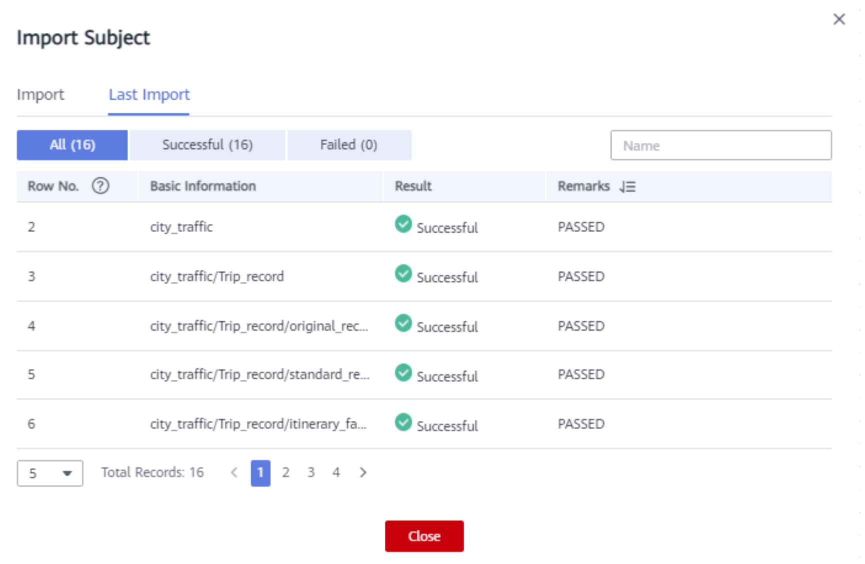
To avoid import errors, use the template to fill in data. Subject Template

* Update Existing Data [✓ No] Yes

* File Subject.xlsx (16.87KB)

Step 3 After the file is uploaded, the system automatically starts importing it. After the file is imported, the import result is displayed.

Figure 3-32 Successful subject import



Import Subject

Import | Last Import

All (16) | Successful (16) | Failed (0)

Row No.	Basic Information	Result	Remarks
2	city_traffic	Successful	PASSED
3	city_traffic/Trip_record	Successful	PASSED
4	city_traffic/Trip_record/original_rec...	Successful	PASSED
5	city_traffic/Trip_record/standard_re...	Successful	PASSED
6	city_traffic/Trip_record/itinerary_fa...	Successful	PASSED

5 Total Records: 16 1 2 3 4

Step 4 After the import is successful, click **Publish**. The subject status will change to **Published**.

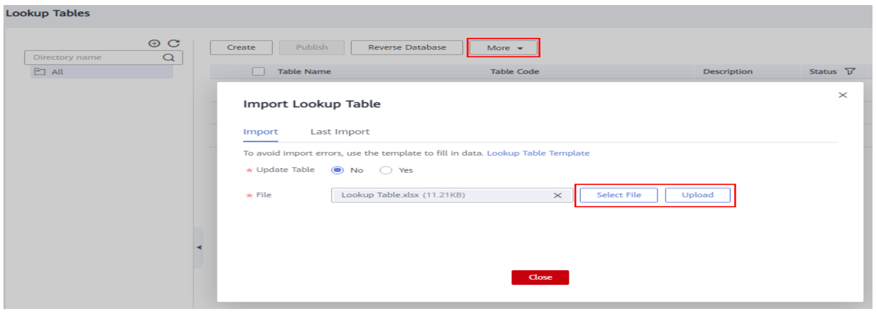
----End

Importing lookup tables

Step 1 On the **DataArts Architecture** page, choose **Lookup Tables** in the left navigation pane.

Step 2 Click **More** above the list and select **Import** from the drop-down list box. In the displayed dialog box, select and upload the lookup table file to import.

Figure 3-33 Importing lookup tables



Import Lookup Table

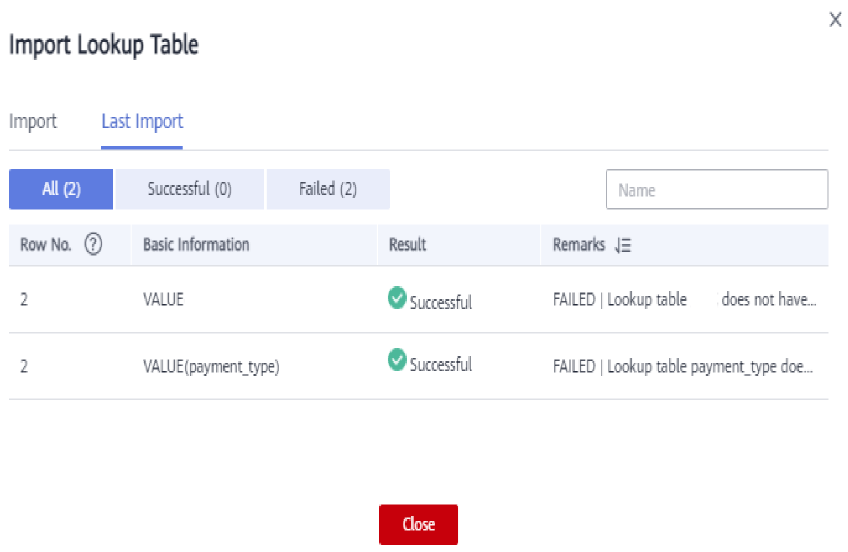
To avoid import errors, use the template to fill in data. Lookup Table Template

Update Table [✓ No] Yes

File: Lookup Table.xlsx (11.21KB)

Step 3 After the file is uploaded, the system automatically starts importing it. After the file is imported, the import result is displayed.

Figure 3-34 Successful import of lookup tables



Row No.	Basic Information	Result	Remarks
2	VALUE	✓ Successful	FAILED | Lookup table does not have...
2	VALUE(payment_type)	✓ Successful	FAILED | Lookup table payment_type doe...

Step 4 After the import is successful, click **Publish**. The lookup table status will change to **Published**.

----End

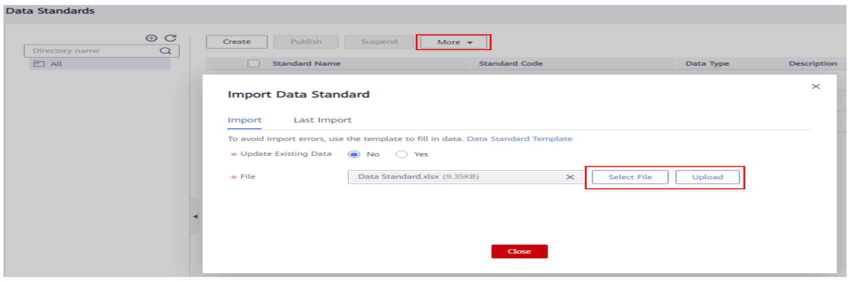
Importing data standards

Step 1 On the **DataArts Architecture** page, choose **Data Standards** in the left navigation pane.

Step 2 If you access the **Data Standards** page for the first time, you will be prompted to customize the data standard template. Edit the data standard template for the new workspace by referring to the **Standard Templates** page of **Configuration Center** in the old workspace, and then click **OK**.

Step 3 Click **More** above the list and select **Import** from the drop-down list box. In the displayed dialog box, select and upload the data standard file to import.

Figure 3-35 Importing data standards



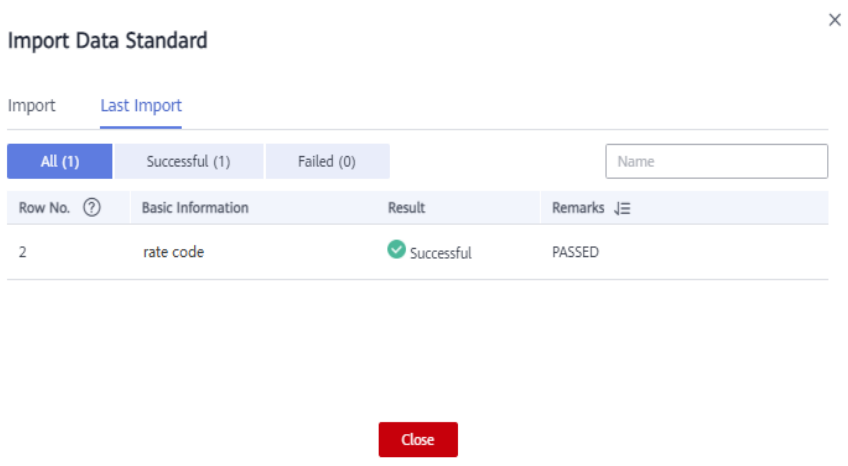
To avoid import errors, use the template to fill in data. Data Standard Template

* Update Existing Data [✓ No] Yes

* File Data Standard.xlsx (9.35KB)

Step 4 After the file is uploaded, the system automatically starts importing it. After the file is imported, the import result is displayed.

Figure 3-36 Successful import of data standards



Import Data Standard

Import | Last Import

All (1) | Successful (1) | Failed (0) | Name

Row No.	Basic Information	Result	Remarks
2	rate code	✓ Successful	PASSED

Close

Step 5 After the import is successful, click **Publish**. The lookup table status will change to **Published**.

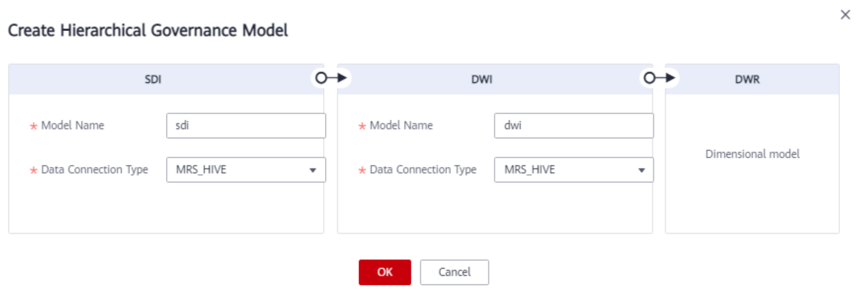
----End

Importing ER modeling tables/entities

Step 1 On the **DataArts Architecture** page, choose **ER Modeling** in the left navigation pane.

Step 2 If no ER model has been created, a dialog box is displayed, asking you to create a hierarchical governance model. Create an SDI and a DWI model for the new workspace by referring to the **ER Modeling** page of the old workspace, and then click **OK**.

Figure 3-37 Creating a hierarchical governance model



Create Hierarchical Governance Model

SDI
* Model Name: sdi
* Data Connection Type: MRS_HIVE

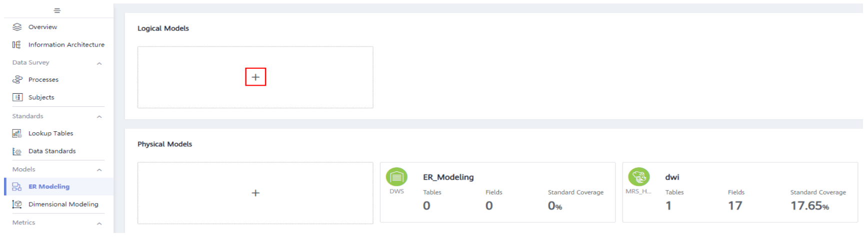
DWI
* Model Name: dwi
* Data Connection Type: MRS_HIVE

DWR
Dimensional model

OK | Cancel

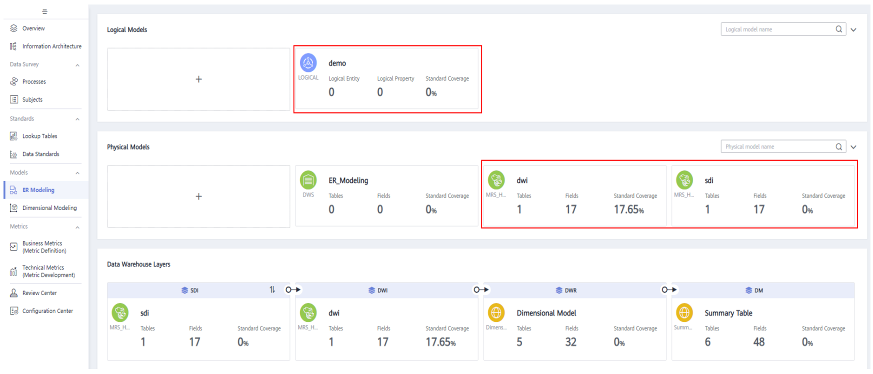
Step 3 If a logical model has been created in the old workspace, click + to create a logical model in the new workspace.

Figure 3-38 Creating a logical model



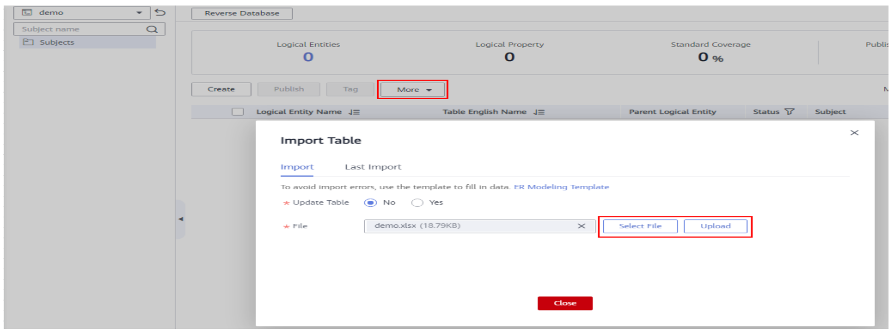
Step 4 Access a logical or physical model to import tables/entities to the model. This section uses logical model **demo** as an example.

Figure 3-39 Accessing the model



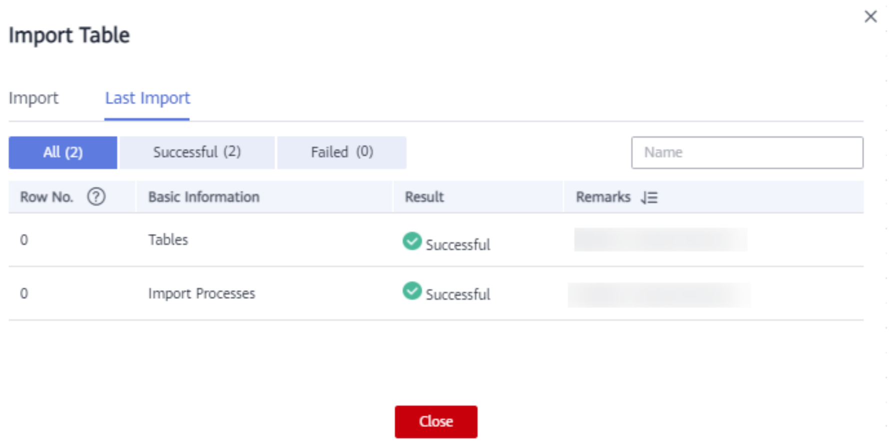
Step 5 Click **More** above the list and select **Import** from the drop-down list box. In the displayed dialog box, select and upload the table/entity file to import.

Figure 3-40 Importing ER modeling tables/entities



Step 6 After the file is uploaded, the system automatically starts importing it. After the file is imported, the import result is displayed.

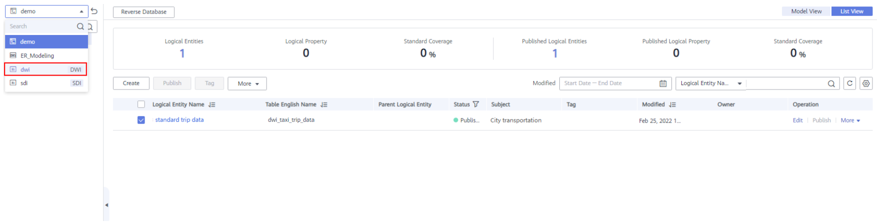
Figure 3-41 Successful import of ER modeling tables/entities



Row No.	Basic Information	Result	Remarks
0	Tables	Successful	
0	Import Processes	Successful	

Step 7 In the subject list, select other models and repeat **Step 5** to **Step 6** to import tables/entities to other models.

Figure 3-42 Importing tables/entities to other models



Step 8 After the import is successful, click **Publish**. The lookup table status will change to **Published**.

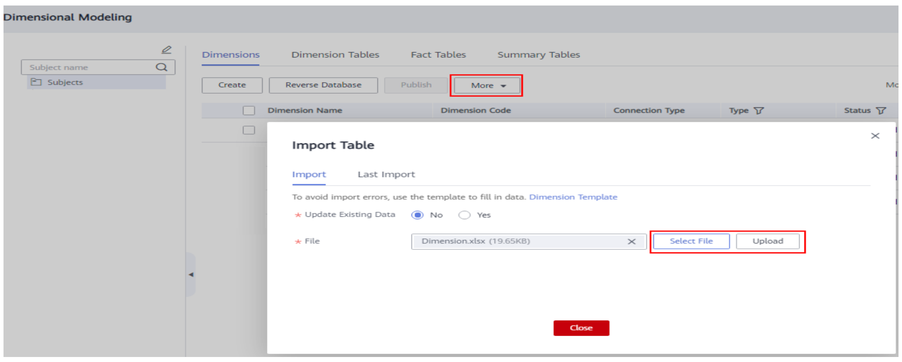
----End

Importing dimensions/fact tables

Step 1 On the **DataArts Architecture** page, choose **Dimensional Modeling** in the left navigation pane.

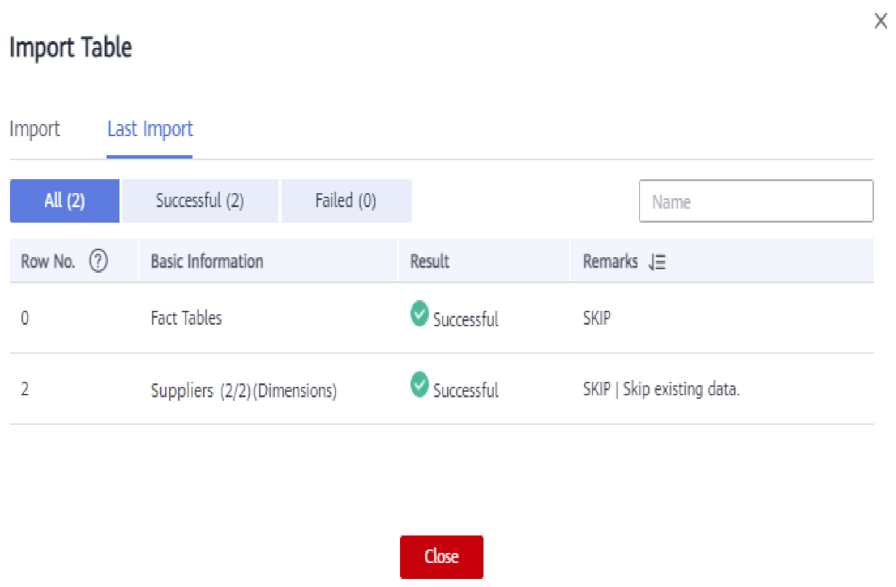
Step 2 On the displayed **Dimensions** page, click **More** above the list and select **Import** from the drop-down list box. In the displayed dialog box, select and upload the dimension file to import.

Figure 3-43 Importing dimensions



Step 3 After the file is uploaded, the system automatically starts importing it. After the file is imported, the import result is displayed.

Figure 3-44 Successful dimension import


Import Table

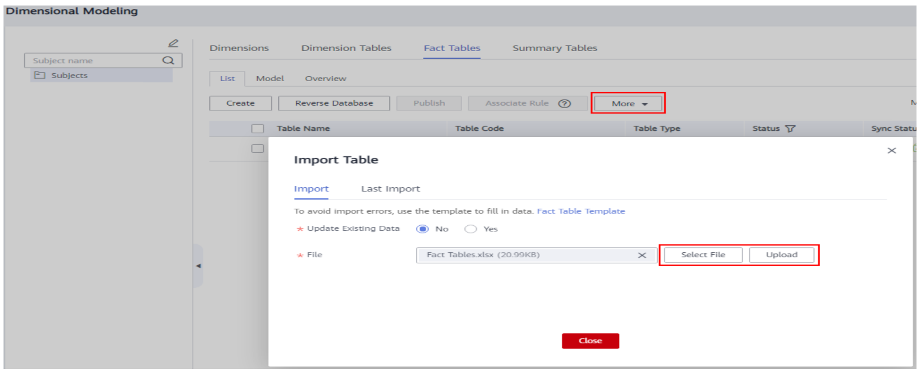
Import | Last Import

All (2) | Successful (2) | Failed (0)

Row No.	Basic Information	Result	Remarks
0	Fact Tables	✓ Successful	SKIP
2	Suppliers (2/2)(Dimensions)	✓ Successful	SKIP | Skip existing data.

Step 4 Click the **Fact Tables** page, click **More** above the list, and select **Import** from the drop-down list box. In the displayed dialog box, select and upload the fact table file to import.

Figure 3-45 Importing fact tables



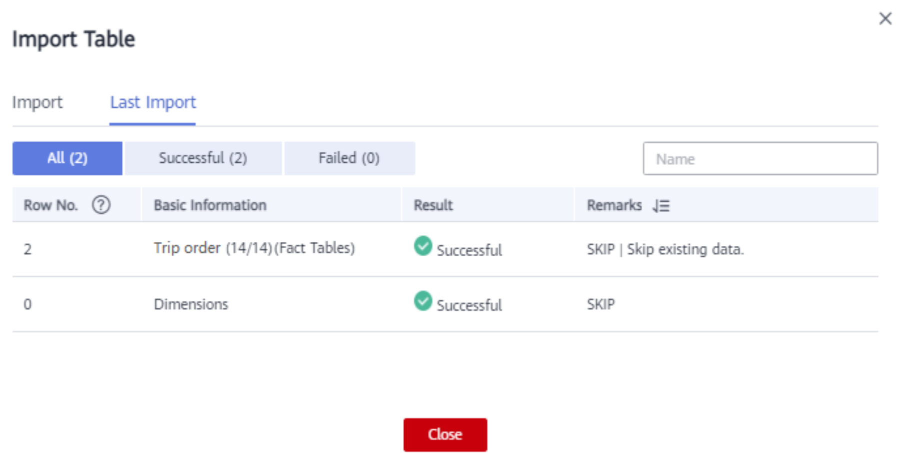
To avoid import errors, use the template to fill in data. Fact Table Template

* Update Existing Data [✓ No] Yes

* File Fact Tables.xlsx (20.99KB)

Step 5 After the file is uploaded, the system automatically starts importing it. After the file is imported, the import result is displayed.

Figure 3-46 Successful import of fact tables



Row No.	Basic Information	Result	Remarks
2	Trip order (14/14)(Fact Tables)	✓ Successful	SKIP | Skip existing data.
0	Dimensions	✓ Successful	SKIP

Step 6 After the import is successful, click **Publish**. The lookup table status will change to **Published**.

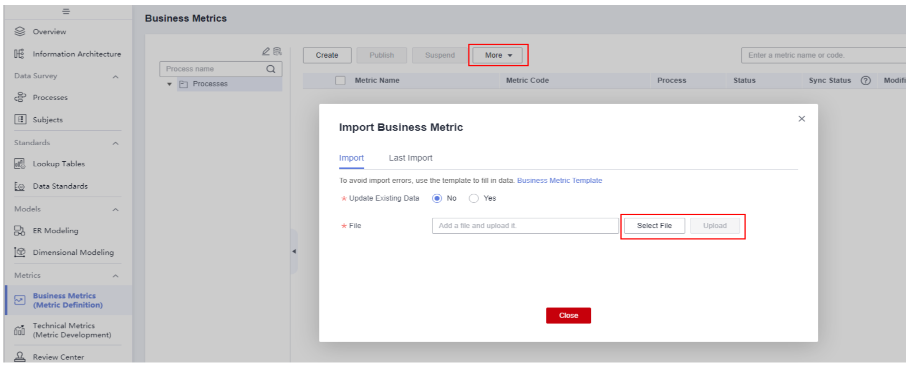
----End

Importing business metrics

Step 1 On the **DataArts Architecture** page, choose **Business Metrics** in the left navigation pane.

Step 2 Click **More** above the list and select **Import** from the drop-down list box. In the displayed dialog box, select and upload the business metric file to import.

Figure 3-47 Importing business metrics



Import Business Metric

Import | Last Import

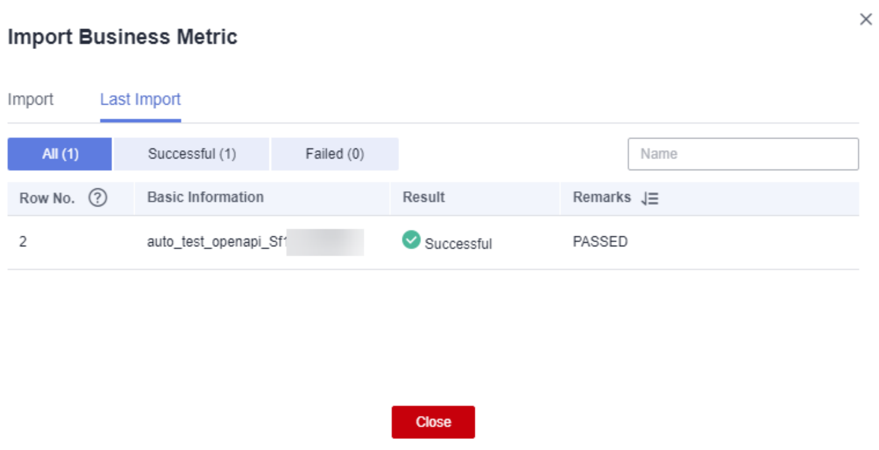
To avoid import errors, use the template to fill in data. Business Metric Template

* Update Existing Data [✓ No] Yes

* File Add a file and upload it.

Step 3 After the file is uploaded, the system automatically starts importing it. After the file is imported, the import result is displayed.

Figure 3-48 Successful import of business metrics



Import Business Metric

Import | Last Import

All (1) | Successful (1) | Failed (0)

Row No.	Basic Information	Result	Remarks
2	auto_test_openapi_Sf[redacted]	Successful	PASSED

Close

Step 4 After the import is successful, click **Publish**. The lookup table status will change to **Published**.

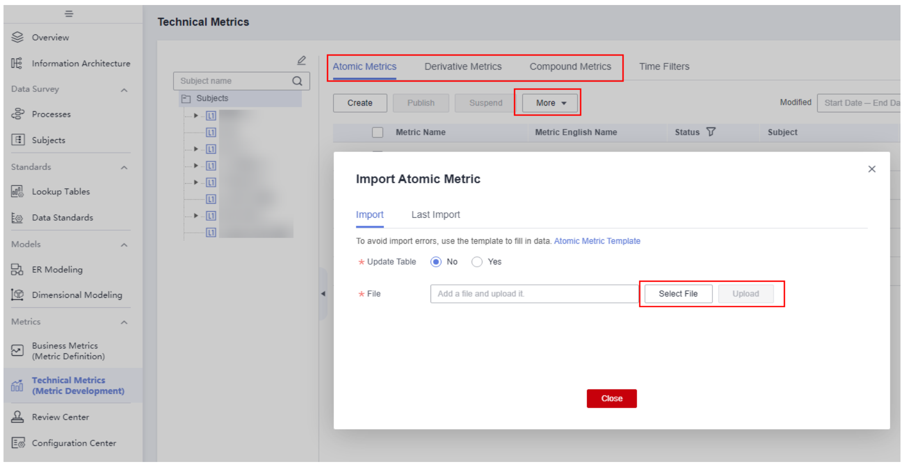
----End

Importing technical metrics

Step 1 On the **DataArts Architecture** page, choose **Technical Metrics** in the left navigation pane.

Step 2 Click the **Atomic Metrics**, **Derivative Metrics**, and **Compound Metrics** tab, respectively. Click **More** and select **Import**. In the displayed dialog box, select and upload a technical metric file.

Figure 3-49 Importing technical metrics



Import Atomic Metric

Import | Last Import

To avoid import errors, use the template to fill in data. Atomic Metric Template

* Update Table [✓ No] Yes

* File Add a file and upload it. Select File Upload

Close

Step 3 After the file is uploaded, the system automatically starts importing it. After the file is imported, the import result is displayed.

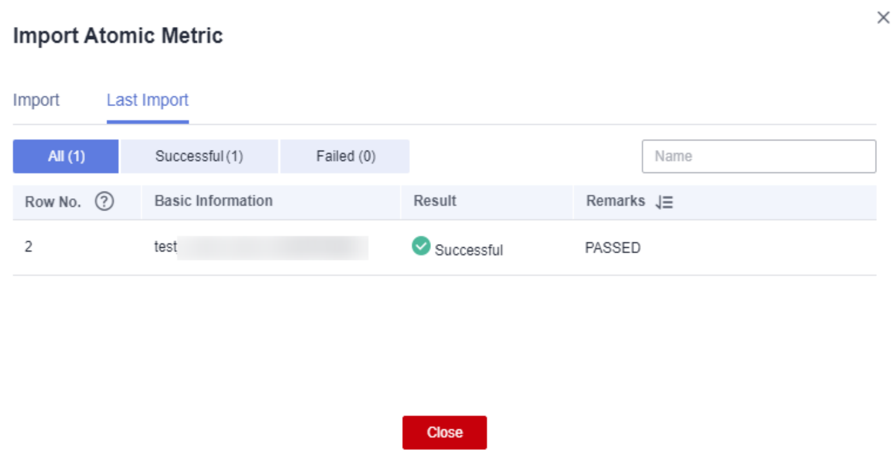


Figure 3-50 Successful import of technical metrics

Import Atomic Metric

Import | Last Import

All (1) | Successful (1) | Failed (0) | Name

Row No.	Basic Information	Result	Remarks
2	test	✓ Successful	PASSED

Step 4 After the import is successful, click **Publish**. The lookup table status will change to **Published**.

----End

Importing summary tables

Step 1 On the **DataArts Architecture** page, choose **Dimensional Modeling** in the left navigation pane.

Step 2 Click the **Summary Tables** page, click **More** above the list, and select **Import** from the drop-down list box. In the displayed dialog box, select and upload the summary table file to import.

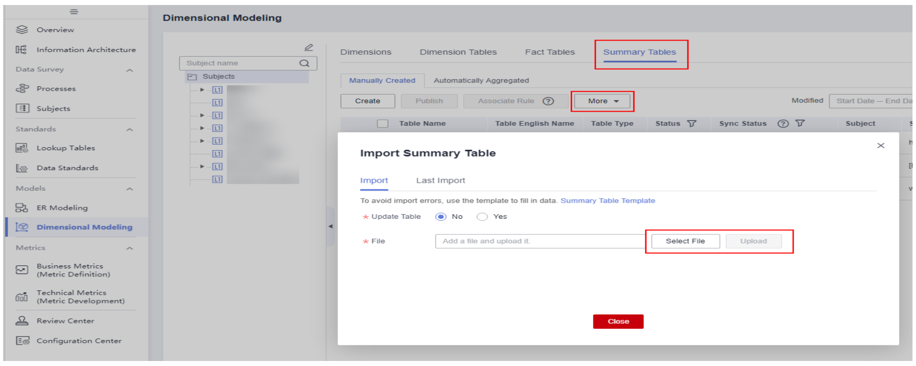


Figure 3-51 Importing summary tables

Import Summary Table

Import | Last Import

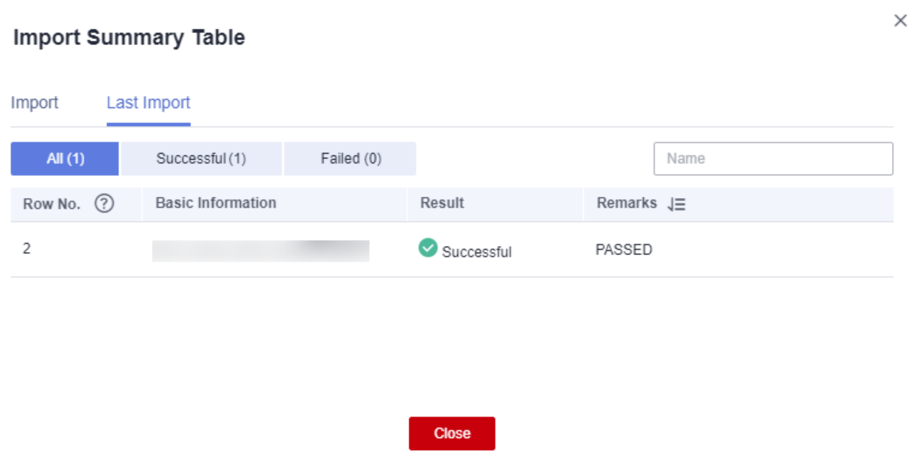
To avoid import errors, use the template to fill in data. Summary Table Template

* Update Table [✓ No] Yes

* File: Add a file and upload it. Select File Upload

Step 3 After the file is uploaded, the system automatically starts importing it. After the file is imported, the import result is displayed.

Figure 3-52 Successful import of summary tables



Row No.	Basic Information	Result	Remarks
2		Successful	PASSED

Step 4 After the import is successful, click **Publish**. The lookup table status will change to **Published**.

----End

Verifying the Migration

Check whether the models and table data imported to the new workspace are consistent with those in the old workspace. If they are consistent, the migration is successful.

3.5 DataArts Factory Data Migration

This function depends on the script, job, environment variable, and resource import and export functions of DataArts Factory.

Constraints

- **Management Center Data Migration** is complete.
- Data such as notifications, backups, job tags, agencies, and default items cannot be imported or exported. If such data needs to be migrated, you need to manually synchronize the data.
- Importing scripts, jobs, environment variables, and resources depends on the OBS service.

Exporting Data from the Old Workspace

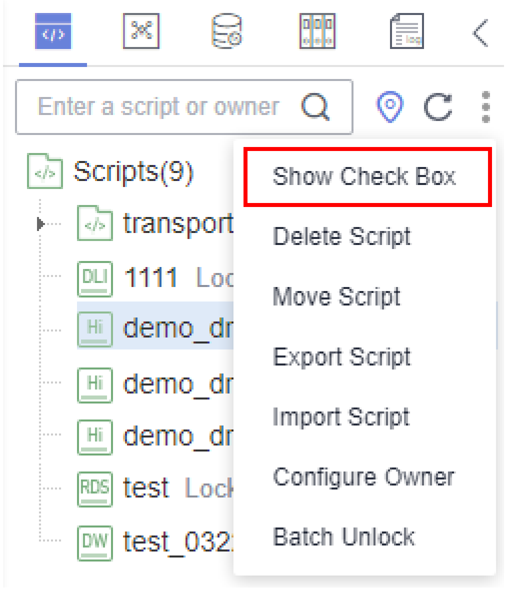
Log in to the console, access the DataArts Factory module of the old workspace, and perform the following operations in sequence to export **scripts**, **jobs**, **environment variables**, and **resources**:

Exporting scripts

Step 1 On the **DataArts Factory** page, choose **Develop Script** in the left navigation pane.

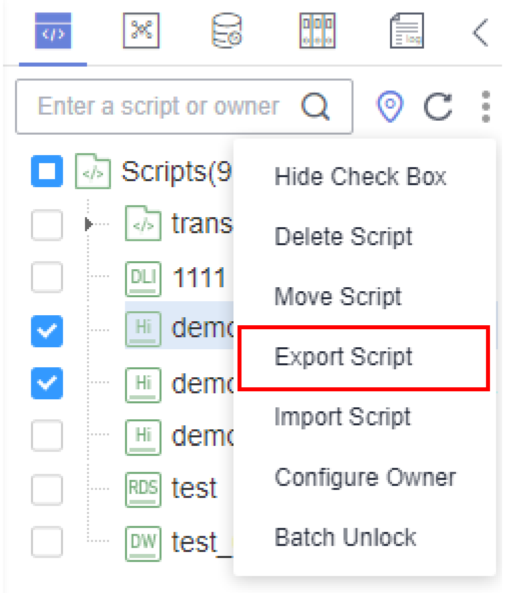
Step 2 Click ⋮ in the script directory and select **Show Check Box**.

Figure 3-53 Clicking Show Check Box



Step 3 Select the scripts to be exported, click ⋮, and choose **Export Script**. After the export is successful, you can obtain the exported .zip file.

Figure 3-54 Selecting and exporting scripts



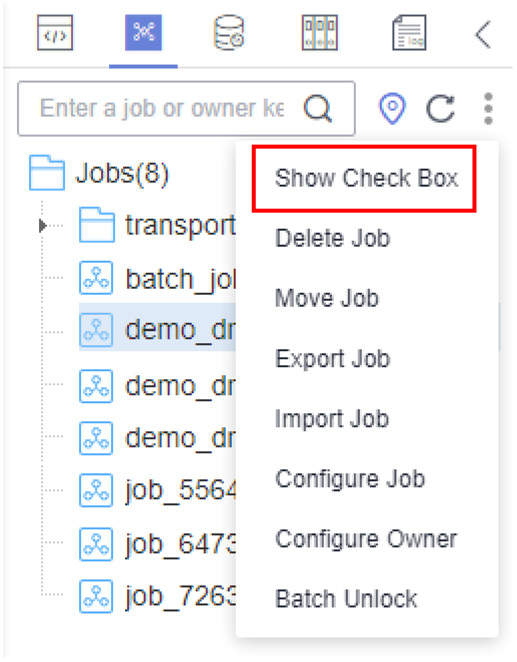
----End

Exporting jobs

Step 1 Click the job icon above the script tree to switch to the job directory.

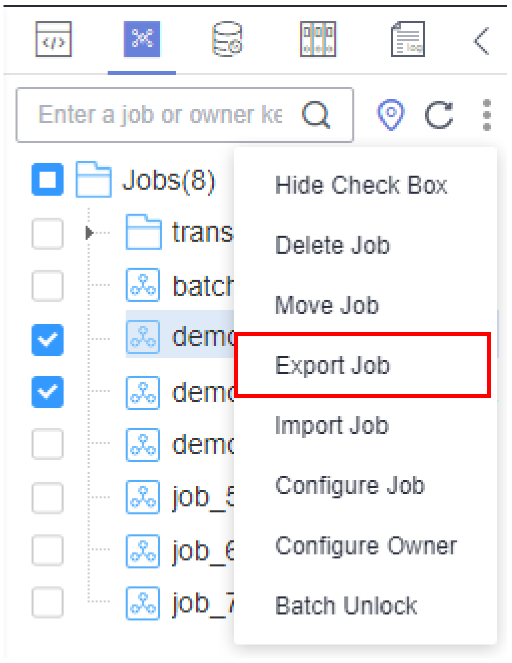
Step 2 Click ⋮ in the job directory and select **Show Check Box**.

Figure 3-55 Clicking Show Check Box



Step 3 Select the jobs to export, click ⋮, and select **Export Job**. In the displayed dialog box, select **Export jobs only** or **Export jobs and their dependency scripts and resource definitions**. After the export is successful, you can obtain the exported .zip file.

Figure 3-56 Selecting and exporting a job



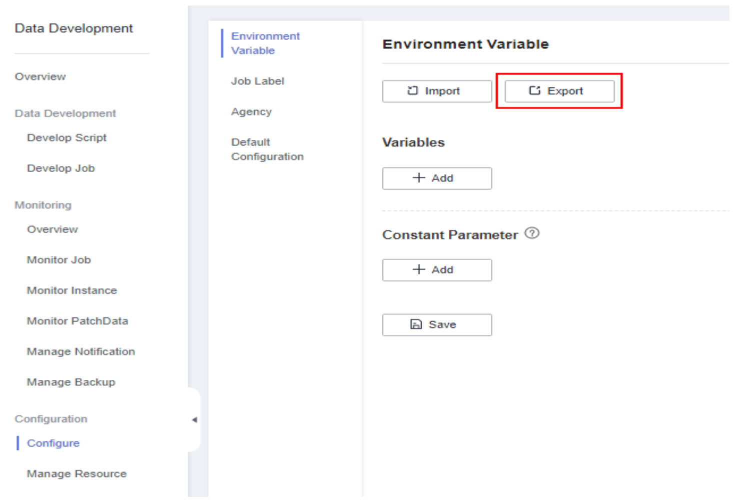
----End

Exporting environment variables

Step 1 In the navigation pane on the left, choose **Configure**.

Step 2 On the **Environment Variable** page, click **Export**.

Figure 3-57 Exporting environment variables



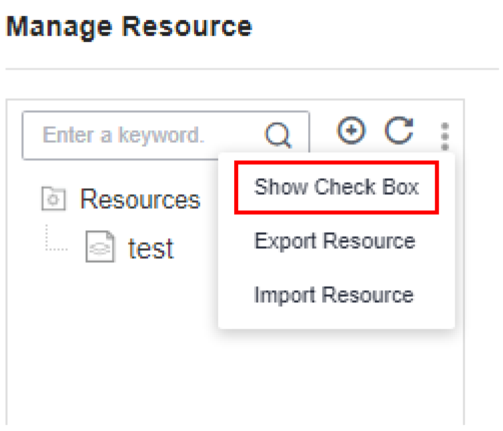
----End

Exporting resources

Step 1 In the left navigation pane, choose **Manage Resource**.

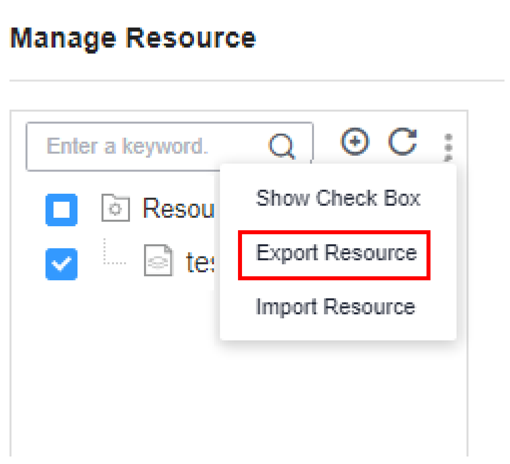
Step 2 Click ⋮ in the resource directory and select **Show Check Box**.

Figure 3-58 Showing the check box



Step 3 Select the resources to export, click ⋮, and select **Export Resource**. After the export is successful, you can obtain the exported .zip file.

Figure 3-59 Selecting and exporting resources



----End

Importing Data to the New Workspace

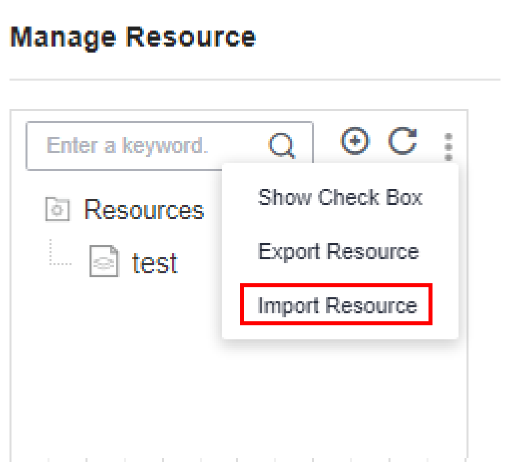
Log in to the console, access the **DataArts Factory** module of the new workspace, and perform the following operations in sequence to import **resources**, **environment variables**, **scripts**, and **jobs**:

Importing resources

Step 1 On the **DataArts Factory** page, choose **Manage Resource** in the left navigation pane.

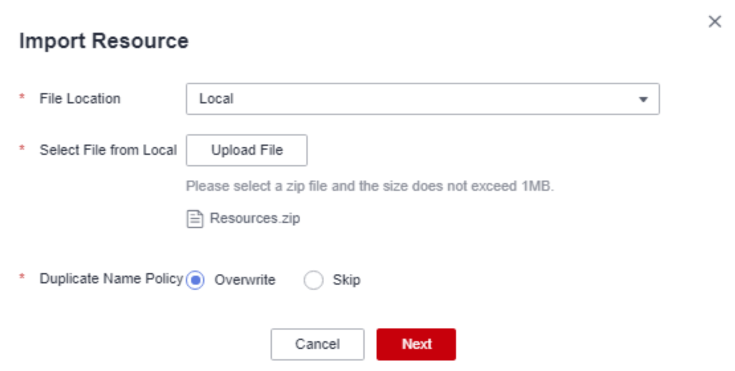
Step 2 Click ⋮ in the resource directory and select **Import Resource**.

Figure 3-60 Selecting Import Resource



Step 3 In the displayed dialog box, select **Local** for **File Location**, select the resource file exported from the old workspace, select **Overwrite** for **Duplicate Name Policy**, and click **Next**.

Figure 3-61 Importing resources


Import Resource

* File Location: Local

* Select File from Local: Upload File

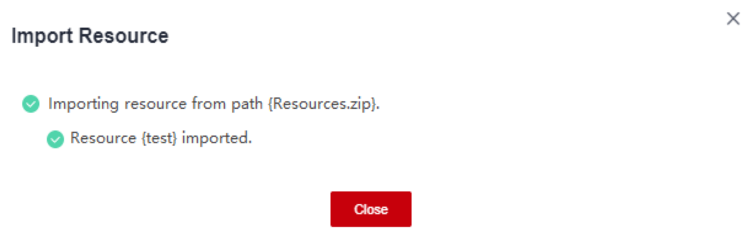
Please select a zip file and the size does not exceed 1MB.

Resources.zip

* Duplicate Name Policy: [✓ Overwrite] Skip

Step 4 The system starts to import resources. After the import is successful, the names of the imported resources are displayed.

Figure 3-62 Successful resource import


Import Resource

Importing resource from path {Resources.zip}.

Resource {test} imported.

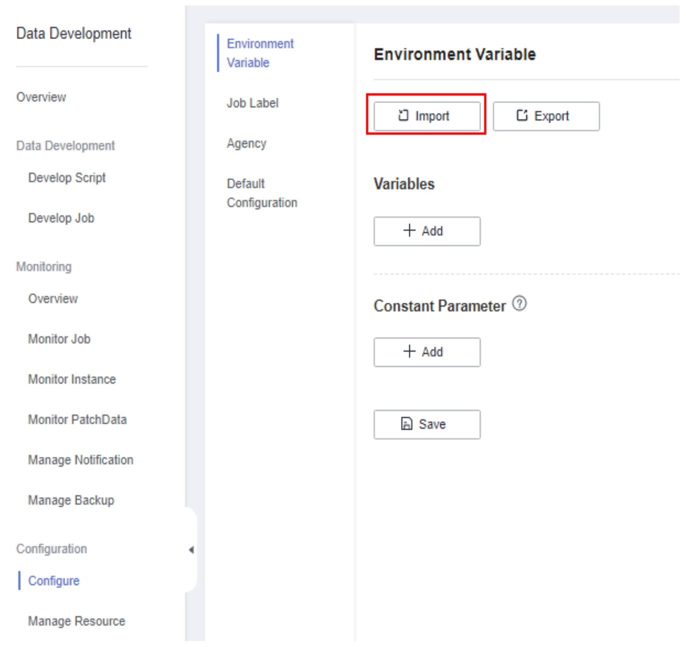
----End

Importing environment variables

Step 1 In the navigation pane on the left, choose **Configure**.

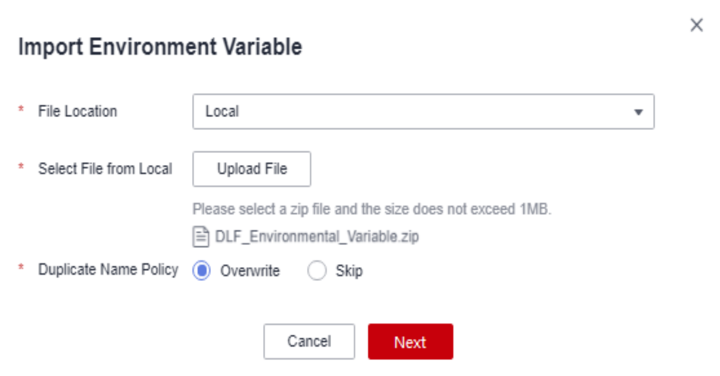
Step 2 On the **Environment Variable** page, click **Import**.

Figure 3-63 Clicking Import



Step 3 In the displayed dialog box, select **Local** for **File Location**, select the environment variable file exported from the old workspace, select **Overwrite** for **Duplicate Name Policy**, and click **Next**.

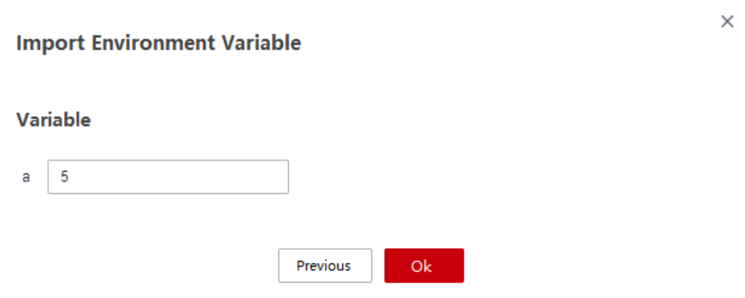
Figure 3-64 Importing environment variables


Import Environment Variable

* File Location	Local
* Select File from Local	Upload File
	Please select a zip file and the size does not exceed 1MB.
	DLF_Environmental_Variable.zip
* Duplicate Name Policy	[✓ Overwrite] Skip

Step 4 The system starts to import the environment variables. You can choose whether to change the values of the variables.

Figure 3-65 Confirming the import result


Import Environment Variable

Variable

a 5

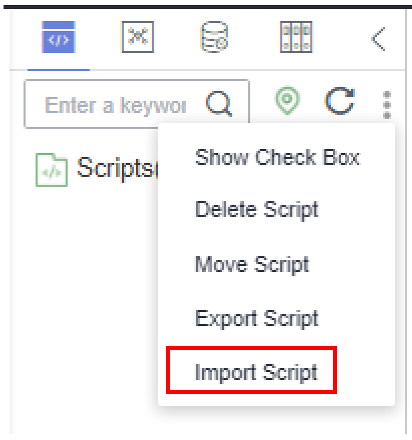
----End

Importing scripts

Step 1 In the left navigation pane, choose **Develop script**.

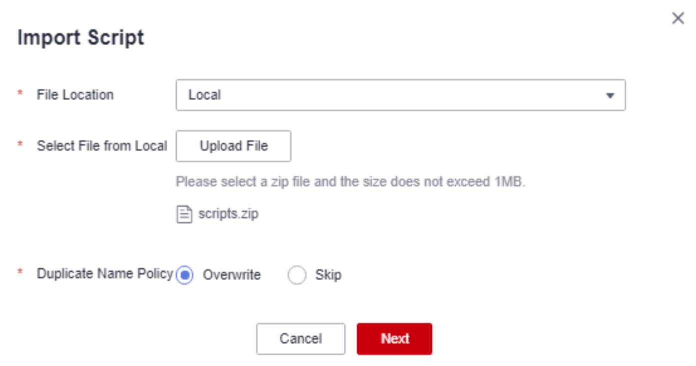
Step 2 Click ⋮ in the script directory and select **Import Script**.

Figure 3-66 Selecting Import Script



Step 3 In the displayed dialog box, select **Local** for **File Location**, select the script file exported from the old workspace, select **Overwrite** for **Duplicate Name Policy**, and click **Next**.

Figure 3-67 Importing scripts

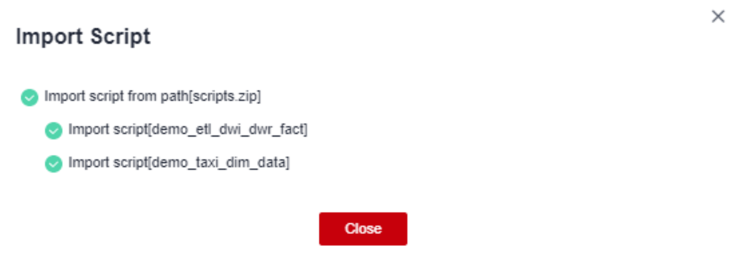


Import Script

* File Location: Local
* Select File from Local: Upload File
Please select a zip file and the size does not exceed 1MB.
scripts.zip
* Duplicate Name Policy: [✓ Overwrite] Skip

Step 4 The system starts to import scripts. After the import is successful, the names of the imported scripts are displayed.

Figure 3-68 Successful script import



Import Script

Import script from path[scripts.zip]
Import script[demo_etl_dwi_dwr_fact]
Import script[demo_taxi_dim_data]

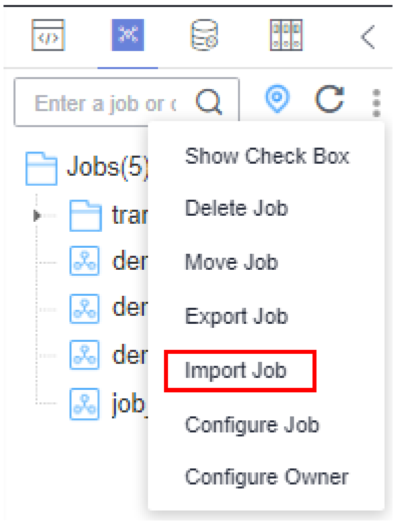
----End

Importing jobs

Step 1 Click above the script tree to switch to the job directory.

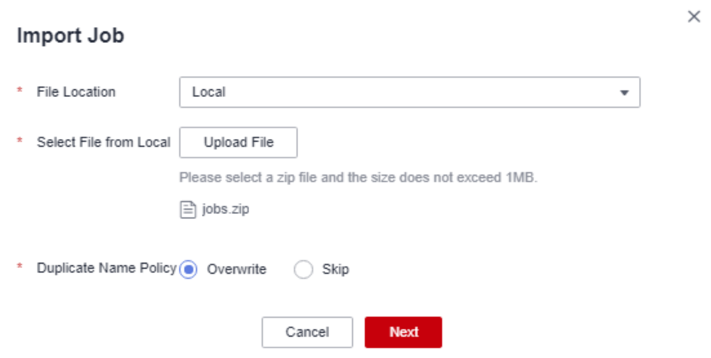
Step 2 Click in the job directory and select **Import Job**.

Figure 3-69 Selecting Import Job



Step 3 In the displayed dialog box, select **Local** for **File Location**, select the job file exported from the old workspace, and click **Next**.

Figure 3-70 Importing jobs

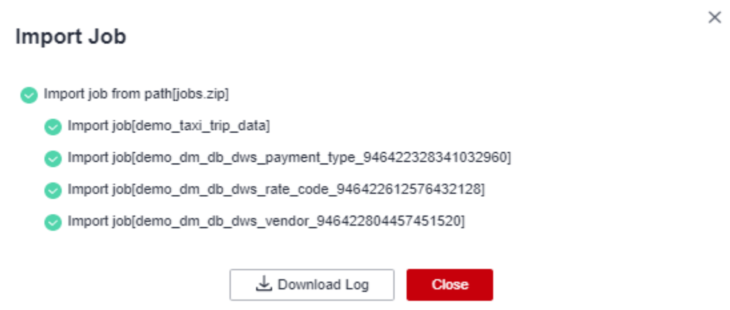


Import Job

* File Location	Local
* Select File from Local	Upload File
	Please select a zip file and the size does not exceed 1MB.
	jobs.zip
* Duplicate Name Policy	[✓ Overwrite] Skip

Step 4 The system starts to import jobs. After the import is successful, the names of the imported jobs are displayed.

Figure 3-71 Successful job import



Import Job

✓ Import job from path[jobs.zip]
✓ Import job[demo_taxi_trip_data]
✓ Import job[demo_dm_db_dws_payment_type_946422328341032960]
✓ Import job[demo_dm_db_dws_rate_code_946422612576432128]
✓ Import job[demo_dm_db_dws_vendor_946422804457451520]

----End

Verifying the Migration

Check whether the scripts, jobs, environment variables, and resources imported to the new workspace are consistent with those in the old workspace. If they are consistent, the migration is successful.

3.6 DataArts Quality Data Migration

This function depends on the import and export of rule templates, quality jobs, and comparison jobs of the DataArts Quality module.

Constraints

- **Management Center Data Migration** is complete.
- Metrics, rules, and scenarios in metric monitoring cannot be imported or exported. If they need to be migrated, you need to manually synchronize them.
- To export custom rule templates, perform the following steps (you can export a maximum of 200 rule templates at a time):
- You can import a file containing a maximum of 4 MB data.
- You can export a maximum of 200 quality jobs.
- You can import a file containing a maximum of 4 MB data.
- You can export a maximum of 200 comparison jobs.
- You can import a file containing a maximum of 4 MB data.

Exporting Data from the Old Workspace

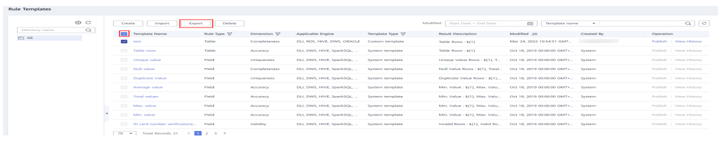
Log in to the console, access the **DataArts Quality** module of the old workspace, and perform the following operations in sequence to export **rule templates**, **quality jobs**, and **comparison jobs**:

Exporting rule templates

Step 1 On the **DataArts Quality** page, choose **Rule Templates** in the left navigation pane.

Step 2 In the rule template list, select the custom rule templates you want to export and click **Export**.

Figure 3-72 Exporting rule templates

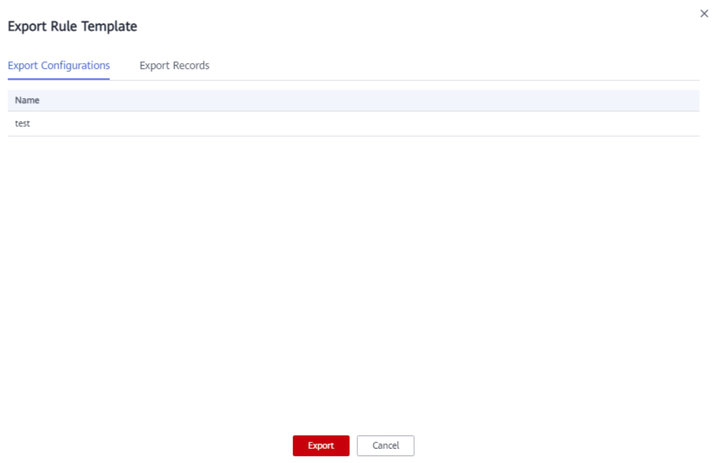


Template Name	Rule Type	Dimension	Applicable Engine	Template Type	Result Description	Modified	Created By	Operation
test	Table	Completeness	DLI, RDS, HIVE, DWS, ORACLE	Custom template	Table Rows : ${1}	Mar 24, 2022 10:54:51 GMT...		Publish | View History
Table rows	Table	Accuracy	DLI, DWS, HIVE, SparkSQL, ...	System template	Table Rows : ${1}	Oct 18, 2019 00:00:00 GMT+...	System	Publish | View History
Unique value	Field	Uniqueness	DLI, DWS, HIVE, SparkSQL, ...	System template	Unique Value Rows : ${1}, T...	Oct 18, 2019 00:00:00 GMT+...	System	Publish | View History
Null value	Field	Completeness	DLI, DWS, HIVE, SparkSQL, ...	System template	Null Value Rows : ${1}, Total...	Oct 18, 2019 00:00:00 GMT+...	System	Publish | View History
Duplicate Value	Field	Uniqueness	DLI, DWS, HIVE, SparkSQL, ...	System template	Duplicate Value Rows : ${1},...	Oct 18, 2019 00:00:00 GMT+...	System	Publish | View History
Average value	Field	Accuracy	DLI, DWS, HIVE, SparkSQL, ...	System template	Min. Value : ${1}, Max. Valu...	Oct 18, 2019 00:00:00 GMT+...	System	Publish | View History
Total values	Field	Accuracy	DLI, DWS, HIVE, SparkSQL, ...	System template	Min. Value : ${1}, Max. Valu...	Oct 18, 2019 00:00:00 GMT+...	System	Publish | View History
Max. value	Field	Accuracy	DLI, DWS, HIVE, SparkSQL, ...	System template	Min. Value : ${1}, Max. Valu...	Oct 18, 2019 00:00:00 GMT+...	System	Publish | View History
Min. value	Field	Accuracy	DLI, DWS, HIVE, SparkSQL, ...	System template	Min. Value : ${1}, Max. Valu...	Oct 18, 2019 00:00:00 GMT+...	System	Publish | View History
ID card number verification[...	Field	Validity	DLI, DWS, HIVE, SparkSQL, ...	System template	Invalid Rows : ${1}, Valid Ro...	Oct 18, 2019 00:00:00 GMT+...	System	Publish | View History

10 Total Records: 21 < 1 2 3 >

Step 3 In the displayed box, confirm the rule templates and click **Export**.

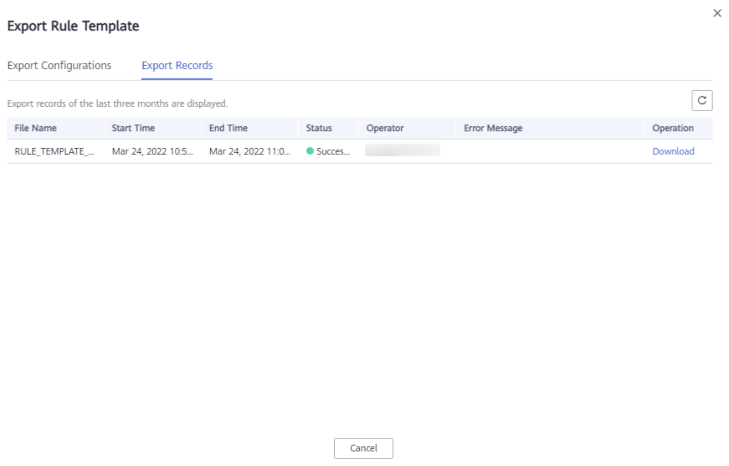
Figure 3-73 Confirming the rule templates to export



Name
test

Step 4 After the export is successful, click **Download** on the **Export Records** page to obtain the exported .xlsx file.

Figure 3-74 Obtaining the exported file of rule templates



Export records of the last three months are displayed.

File Name	Start Time	End Time	Status	Operator	Error Message	Operation
RULE_TEMPLATE_...	Mar 24, 2022 10:5...	Mar 24, 2022 11:0...	● Succes...			Download

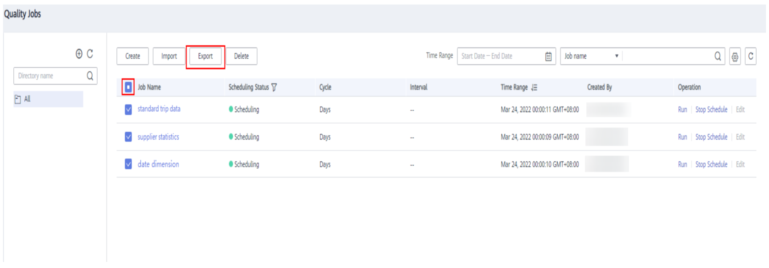
----End

Exporting quality Jobs

Step 1 In the navigation pane on the left, choose **Quality Jobs**.

Step 2 In the quality job list, select the jobs you want to migrate and click **Export**.

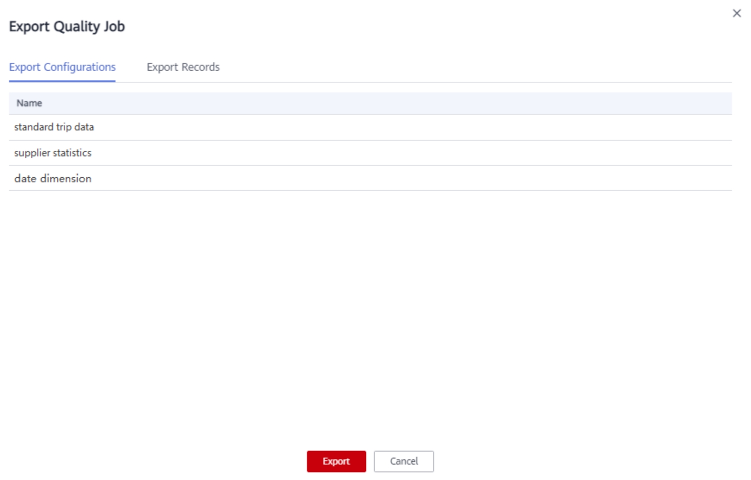
Figure 3-75 Exporting quality jobs



Job Name	Scheduling Status	Cycle	Interval	Time Range	Created By	Operation
standard trip data	● Scheduling	Days	--	Mar 24, 2022 00:00:11 GMT+08:00		Run | Stop Schedule | Edit
supplier statistics	● Scheduling	Days	--	Mar 24, 2022 00:00:09 GMT+08:00		Run | Stop Schedule | Edit
date dimension	● Scheduling	Days	--	Mar 24, 2022 00:00:10 GMT+08:00		Run | Stop Schedule | Edit

Step 3 In the displayed box, confirm the jobs and click **Export**.

Figure 3-76 Confirm the quality jobs to export



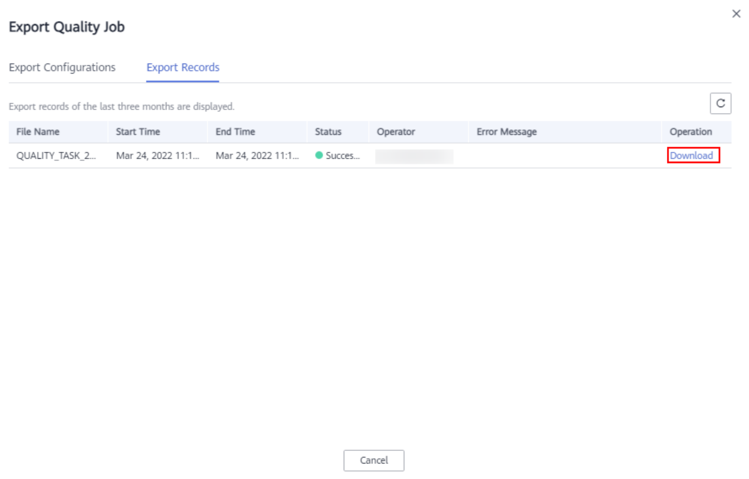
Export Quality Job

Export Configurations | Export Records

Name
standard trip data
supplier statistics
date dimension

Step 4 After the export is successful, click **Download** on the **Export Records** page to obtain the exported .xlsx file.

Figure 3-77 Obtaining the exported file of quality jobs



Export Quality Job

Export Configurations | Export Records

Export records of the last three months are displayed.

File Name	Start Time	End Time	Status	Operator	Error Message	Operation
QUALITY_TASK_2...	Mar 24, 2022 11:1...	Mar 24, 2022 11:1...	● Succes...	[redacted]		Download

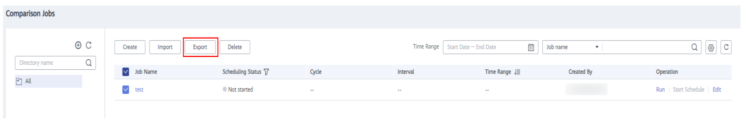
----End

Exporting comparison jobs

Step 1 In the navigation pane on the left, choose **Comparison Jobs**.

Step 2 In the comparison job list, select the jobs you want to migrate and click **Export**.

Figure 3-78 Exporting comparison jobs

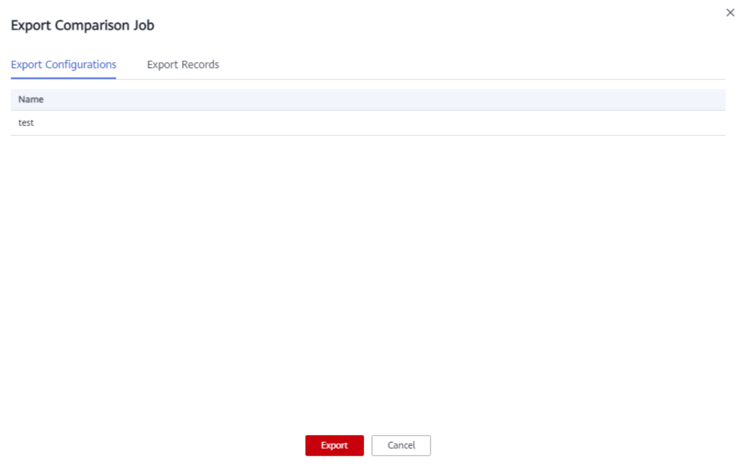


Comparison Jobs

Job Name	Scheduling Status	Cycle	Interval	Time Range	Created By	Operation
test	Not started	--	--	--	[redacted]	Run | Start Schedule | Edit

Step 3 In the displayed box, confirm the jobs and click **Export**.

Figure 3-79 Confirming the comparison jobs to export

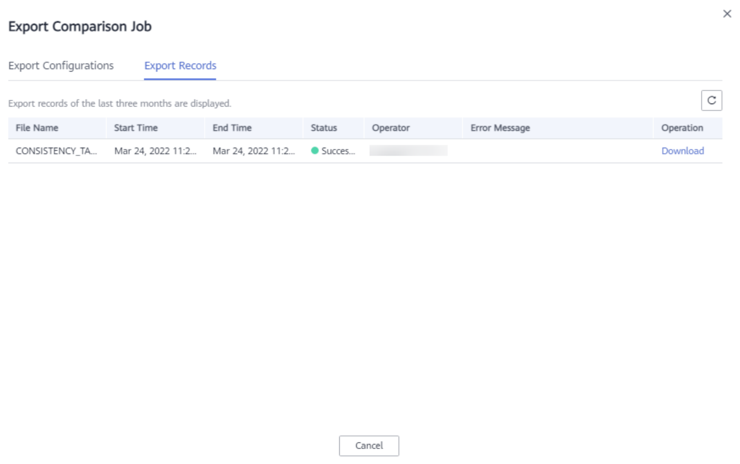

Export Comparison Job

Export Configurations | Export Records

Name
test

Step 4 After the export is successful, click **Download** on the **Export Records** page to obtain the exported .xlsx file.

Figure 3-80 Obtaining the exported file of comparison jobs


Export Comparison Job

Export Configurations | Export Records

Export records of the last three months are displayed.

File Name	Start Time	End Time	Status	Operator	Error Message	Operation
CONSISTENCY_TA...	Mar 24, 2022 11:2...	Mar 24, 2022 11:2...	● Succes...			Download

----End

Importing Data to the New Workspace

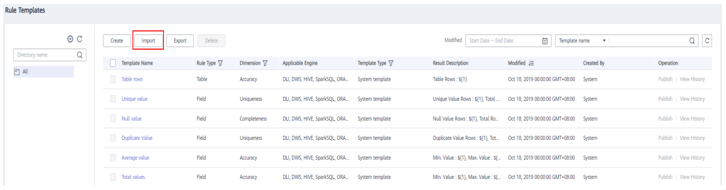
Log in to the console, access the **DataArts Quality** module of the new workspace, and perform the following operations in sequence to import **rule templates**, **quality jobs**, and **comparison jobs**:

Importing rule templates

Step 1 On the **DataArts Quality** page, choose **Rule Templates** in the left navigation pane.

Step 2 Click **Import** above the rule template list.

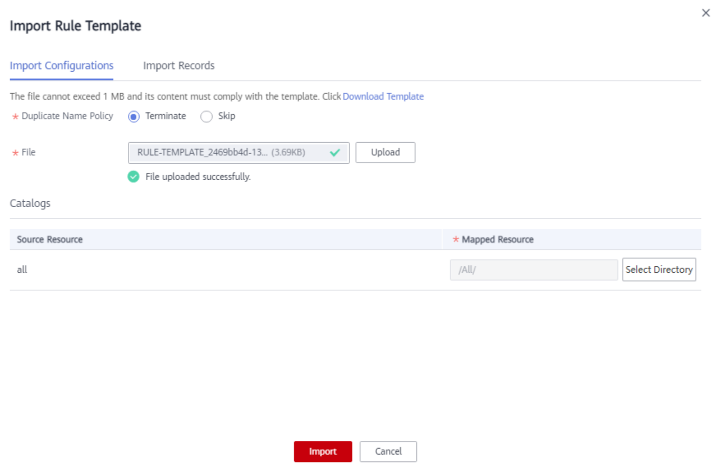
Figure 3-81 Importing rule templates



Template Name	Rule Type	Dimension	Applicable Engine	Template Type	Result Description	Modified	Created By	Operation
Table rows	Table	Accuracy	DLI, DWS, HIVE, SparkSQL, ORA...	System template	Table Rows : ${1}	Oct 18, 2019 00:00:00 GMT+08:00	System	Publish | View History
Unique value	Field	Uniqueness	DLI, DWS, HIVE, SparkSQL, ORA...	System template	Unique Value Rows : ${1}, Total ...	Oct 18, 2019 00:00:00 GMT+08:00	System	Publish | View History
Null value	Field	Completeness	DLI, DWS, HIVE, SparkSQL, ORA...	System template	Null Value Rows : ${1}, Total Ro...	Oct 18, 2019 00:00:00 GMT+08:00	System	Publish | View History
Duplicate Value	Field	Uniqueness	DLI, DWS, HIVE, SparkSQL, ORA...	System template	Duplicate Value Rows : ${1}, Tot...	Oct 18, 2019 00:00:00 GMT+08:00	System	Publish | View History
Average value	Field	Accuracy	DLI, DWS, HIVE, SparkSQL, ORA...	System template	Min. Value : ${1}, Max. Value : ${...	Oct 18, 2019 00:00:00 GMT+08:00	System	Publish | View History
Total values	Field	Accuracy	DLI, DWS, HIVE, SparkSQL, ORA...	System template	Min. Value : ${1}, Max. Value : ${...	Oct 18, 2019 00:00:00 GMT+08:00	System	Publish | View History

Step 3 In the displayed dialog box, select the rule template file exported from the old workspace, select the mapping directory, select **Terminate** for **Duplicate Name Policy**, and click **Import**.

Figure 3-82 Importing rule templates



The file cannot exceed 1 MB and its content must comply with the template. Click Download Template

* Duplicate Name Policy [✓ Terminate] Skip

* File RULE-TEMPLATE_2469bb4d-13... (3.69KB)

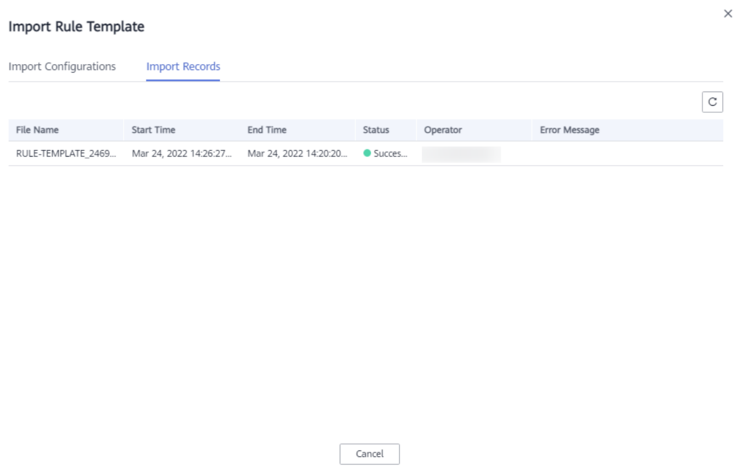
File uploaded successfully.

Catalogs

Source Resource	* Mapped Resource
all	/All/

Step 4 Click the **Import Records** tab to check the import status. The import is successful if the value in the **Status** column is **Successful**.

Figure 3-83 Checking the rule template import result



File Name	Start Time	End Time	Status	Operator	Error Message
RULE-TEMPLATE_2469...	Mar 24, 2022 14:26:27...	Mar 24, 2022 14:20:20...	● Succes...	[redacted]	

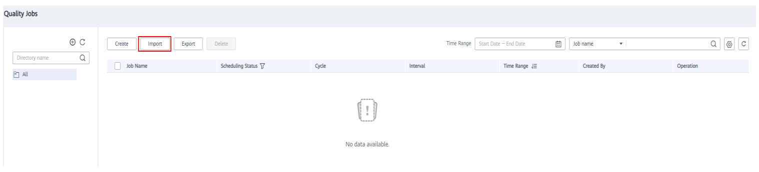
----End

Importing quality jobs

Step 1 In the navigation pane on the left, choose **Quality Jobs**.

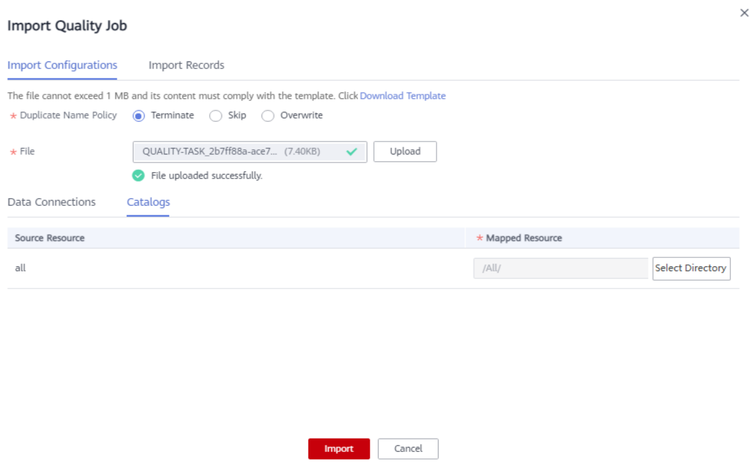
Step 2 Click **Import** above the quality job list.

Figure 3-84 Clicking Import



Step 3 In the displayed dialog box, select the quality job file exported from the old workspace, select the mapping directories for data connections, clusters, and directories, select **Terminate** for **Duplicate Name Policy**, and click **Import**.

Figure 3-85 Importing quality jobs



The file cannot exceed 1 MB and its content must comply with the template. Click Download Template

* Duplicate Name Policy [✓ Terminate] Skip Overwrite

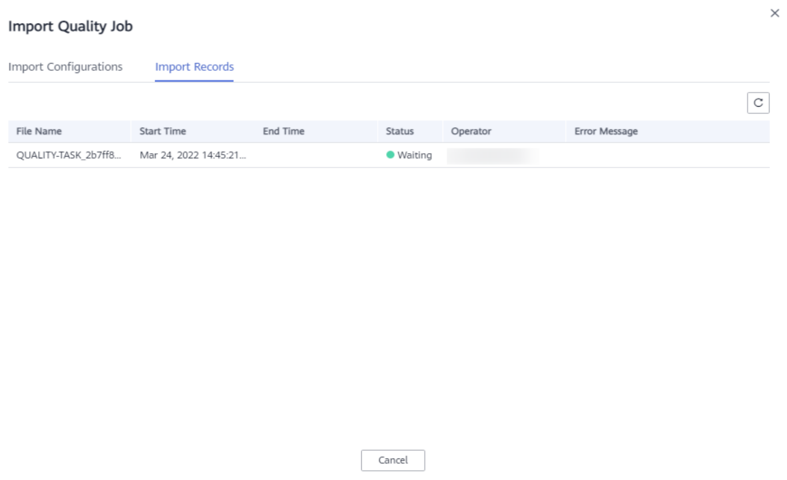
* File QUALITY-TASK_2b7ff88a-ace7... (7.40KB)

File uploaded successfully.

Source Resource	* Mapped Resource
all	/All/

Step 4 Click the **Import Records** tab to check the import status. The import is successful if the value in the **Status** column is **Successful**.

Figure 3-86 Checking the quality job import result



File Name	Start Time	End Time	Status	Operator	Error Message
QUALITY-TASK_2b7ff8...	Mar 24, 2022 14:45:21...		Waiting	[redacted]	

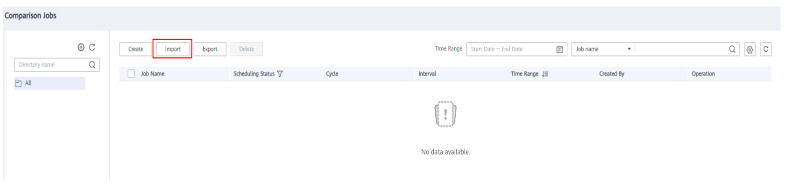
----End

Importing comparison jobs

Step 1 In the navigation pane on the left, choose **Comparison Jobs**.

Step 2 Click **Import** above the comparison job list.

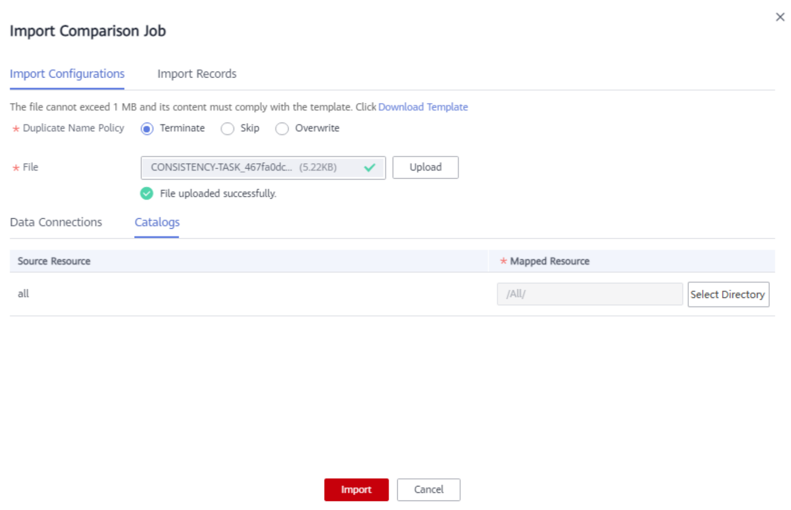
Figure 3-87 Clicking Import



Job Name	Scheduling Status	Cycle	Interval	Time Range	Created By	Operation

Step 3 In the displayed dialog box, select the comparison job file exported from the old workspace, select the mapping directories for data connections, clusters, and directories, select **Terminate** for **Duplicate Name Policy**, and click **Import**.

Figure 3-88 Importing comparison jobs



The file cannot exceed 1 MB and its content must comply with the template. Click Download Template

* Duplicate Name Policy [✓ Terminate] Skip Overwrite

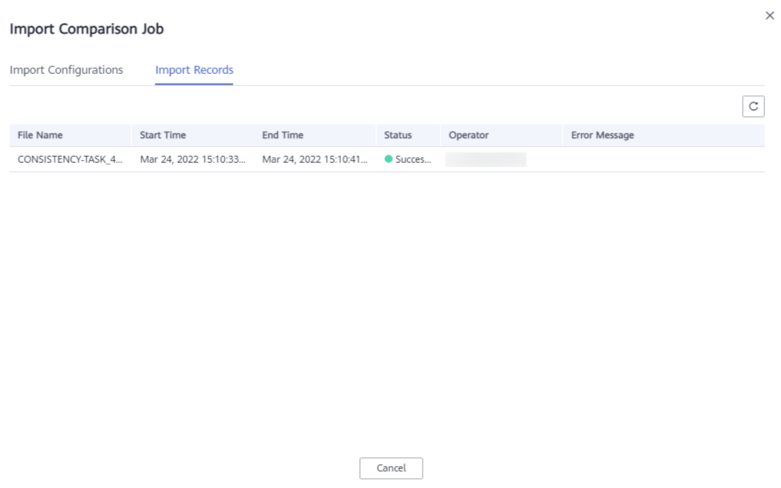
* File CONSISTENCY-TASK_467fa0dc... (5.22KB)

File uploaded successfully.

Source Resource	* Mapped Resource
all	/All/

Step 4 Click the **Import Records** tab to check the import status. The import is successful if the value in the **Status** column is **Successful**.

Figure 3-89 Checking the comparison job import result



File Name	Start Time	End Time	Status	Operator	Error Message
CONSISTENCY-TASK_4...	Mar 24, 2022 15:10:33...	Mar 24, 2022 15:10:41...	● Succes...	[redacted]	

----End

Verifying the Migration

Check whether the rule templates, quality jobs, and comparison jobs imported to the new workspace are consistent with those in the old workspace. If they are consistent, the migration is successful.

3.7 DataArts Catalog Data Migration

This function depends on the resource migration function of Management Center. For details, see **Management Center Data Migration**.

3.8 DataArts Security Data Migration

Data of the DataArts Security module cannot be imported or exported. You need to manually synchronize configurations and tasks.

3.9 DataArts DataService Data Migration

This function depends on the resource migration function of Management Center. For details, see **Management Center Data Migration**.

4 Preventing an IAM User from Logging In to DataArts Studio by Setting Specific Conditions

This section describes how to prevent an IAM user from logging in to DataArts Studio by setting specific conditions. This section uses the time condition as an example, which prevents an IAM user from logging in to DataArts Studio after a specified time.

Prerequisites

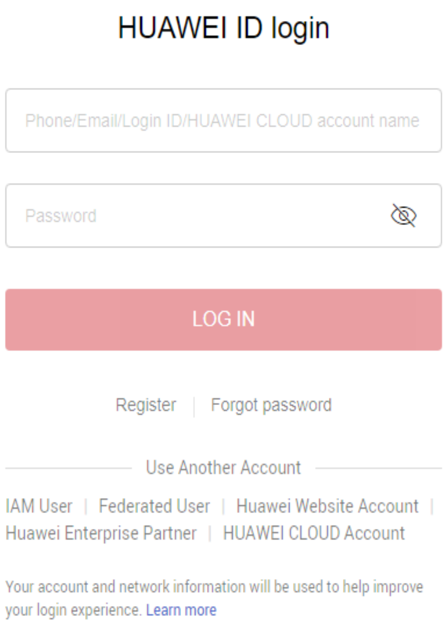
You have created an IAM user. For details, see **Creating an IAM User**.

Creating a Custom Policy

The system default permissions cannot be modified. To set specific conditions, you need to create a custom authorization policy.

Step 1 Log in to the console as the administrator.

Figure 4-1 Logging in to the console



HUAWEI ID login

Phone/Email/Login ID/HUAWEI CLOUD account name

Password

LOG IN

Register | Forgot password

Use Another Account

IAM User | Federated User | Huawei Website Account | Huawei Enterprise Partner | HUAWEI CLOUD Account

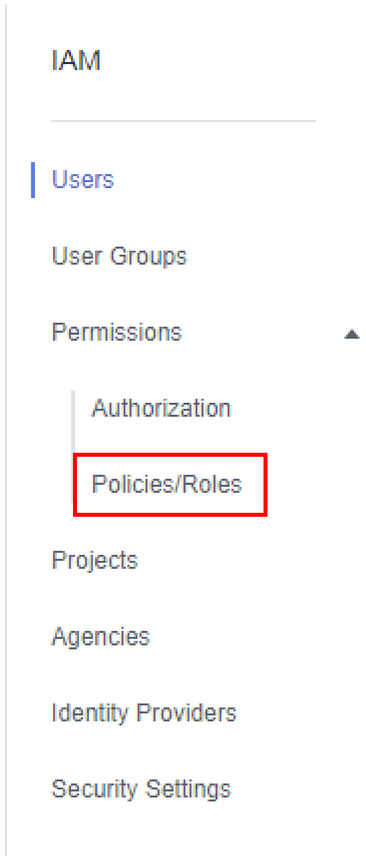
Your account and network information will be used to help improve your login experience. Learn more

Step 2 Hover the mouse over the username in the upper right corner and select **Identity and Access Management**.

Figure 4-2 Identity and Access Management

Step 3 In the navigation pane, choose **Permissions** > **Policies/Roles**.

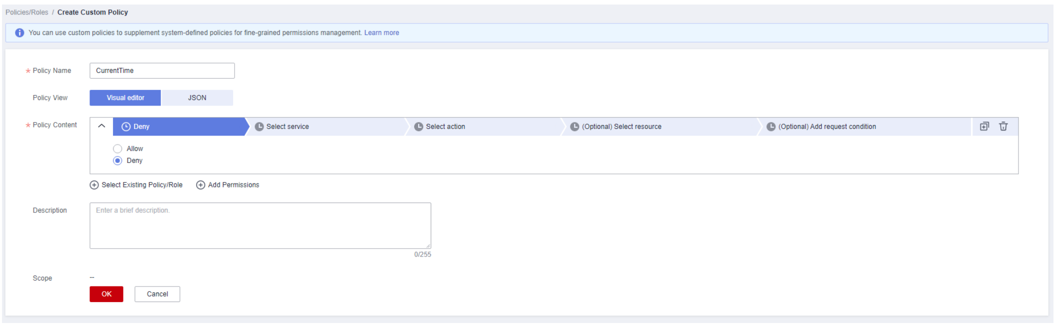
Figure 4-3 Policies/Roles



Step 4 Click **Create Custom Policy** in the upper right corner. For details about how to configure custom permissions, see **Policy Syntax**.

Step 5 On the **Create Custom Policy** page, enter **CurrentTime** for **Policy Name** and select **Visual editor** for **Policy View**.

Figure 4-4 Create Custom Policy



Policies/Roles / Create Custom Policy

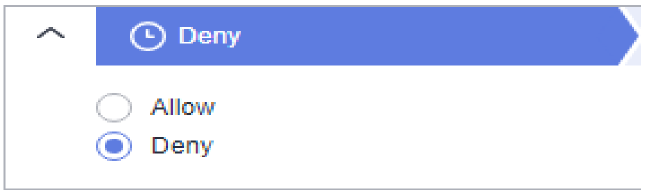
You can use custom policies to supplement system-defined policies for fine-grained permissions management. Learn more

* Policy Name	CurrentTime
Policy View	Visual editor / JSON
* Policy Content	Deny · Select service · Select action · (Optional) Select resource · (Optional) Add request condition Allow [✓ Deny]
	Select Existing Policy/Role · Add Permissions
Description	Enter a brief description. 0/255
Scope	--

Step 6 Configure the policy content.

1. **Allow/Deny**: Select **Deny**.

Figure 4-5 Allow/Deny

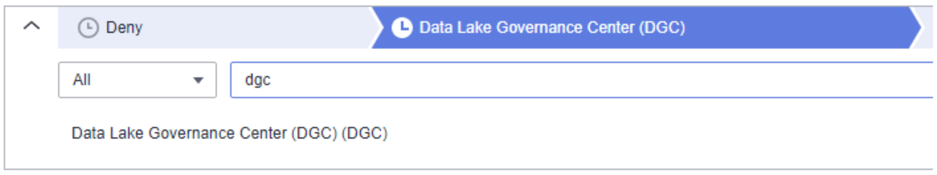


Deny

Allow [✓ Deny]

2. **Select service**: Enter a keyword to search for and select it.

Figure 4-6 Select service



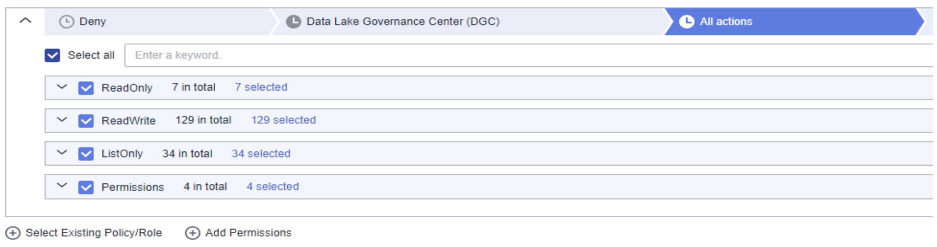
Deny | Data Lake Governance Center (DGC)

All | dgc

Data Lake Governance Center (DGC) (DGC)

3. **Select action**: Select all actions.

Figure 4-7 Select action



Deny | Data Lake Governance Center (DGC) | All actions

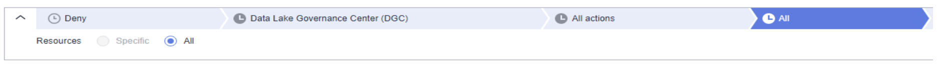
✓ Select all | Enter a keyword.

✓ ReadOnly	7 in total	7 selected
✓ ReadWrite	129 in total	129 selected
✓ ListOnly	34 in total	34 selected
✓ Permissions	4 in total	4 selected

Select Existing Policy/Role | Add Permissions

4. **(Optional) Select resource**: By default, all resources are selected.

Figure 4-8 Select resource

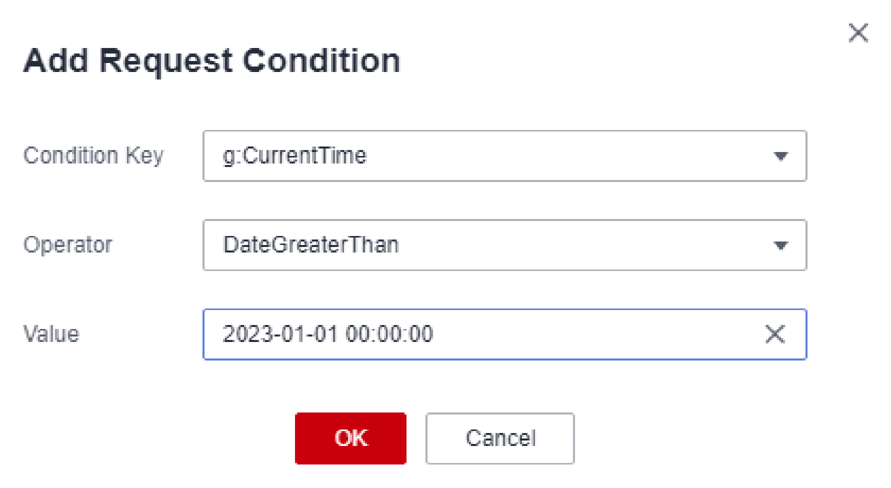


Deny | Data Lake Governance Center (DGC) | All actions | All

Resources: Specific [✓ All]

5. **(Optional) Add request condition**: Click **Add Request Condition**. In the displayed dialog box, select **g:CurrentTime** for **Condition Key**, **DateGreaterThan** for **Qualifier**, and set the time.

Figure 4-9 Add Request Condition

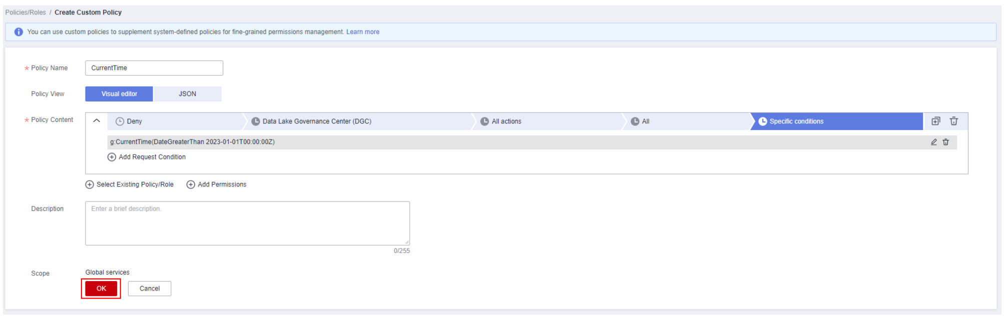


Add Request Condition

Condition Key: g:CurrentTime
Operator: DateGreaterThan
Value: 2023-01-01 00:00:00

6. Click **OK**.

Figure 4-10 Clicking OK



Policies/Roles / Create Custom Policy

You can use custom policies to supplement system-defined policies for fine-grained permissions management. Learn more

* Policy Name: CurrentTime
Policy View: Visual editor | JSON
* Policy Content: Deny | Data Lake Governance Center (DGC) | All actions | All | Specific conditions
g:CurrentTime(DateGreaterThan 2023-01-01T00:00:00Z)
Description: Enter a brief description.
0/255
Scope: Global services

7. Select **JSON** for **Policy View** to view the code for the custom policy.

```
{
    "Version": "1.1",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "DGC:*:*"
            ],
            "Condition": {
                "DateGreaterThan": {
                    "g:CurrentTime": [
                        "2023-01-01T00:00:00Z"
                    ]
                }
            }
        }
    ]
}
```

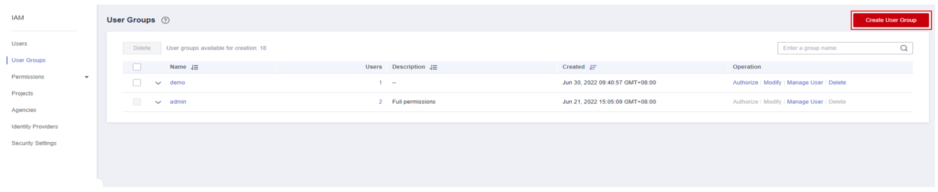
----End

Granting the Custom Policy to an IAM User

The custom policy cannot be directly granted to an IAM user. You must grant the policy to a user group and then add the IAM user to the user group.

Step 1 On the IAM console, choose **User Groups** in the navigation pane. Then click **Create User Group** in the upper right corner.

Figure 4-11 Create User Group

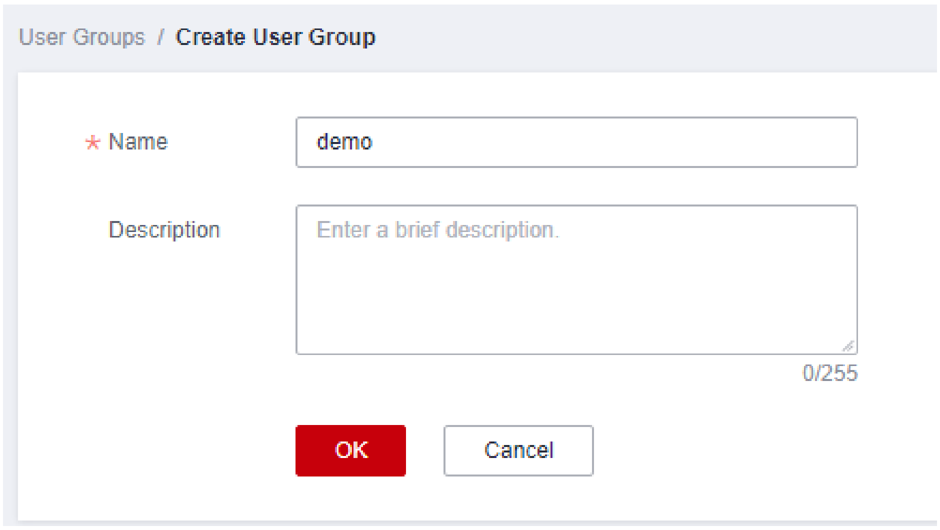


User groups available for creation: 18

Name	Users	Description	Created	Operation
demo	1	--	Jun 30, 2022 09:40:57 GMT+08:00	Authorize | Modify | Manage User | Delete
admin	2	Full permissions	Jun 21, 2022 15:05:09 GMT+08:00	Authorize | Modify | Manage User | Delete

Step 2 Enter the user group name and click **OK**.

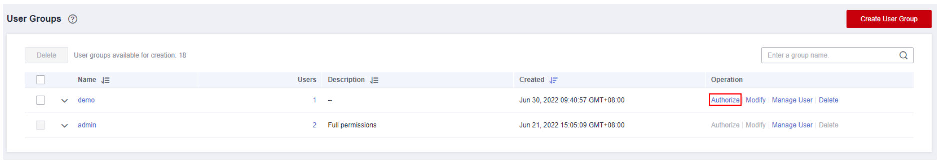
Figure 4-12 Entering the user group name



* Name	demo
Description	

Step 3 On the **User Groups** page, locate the create user group and click **Authorize** in the **Operation** column.

Figure 4-13 Authorize

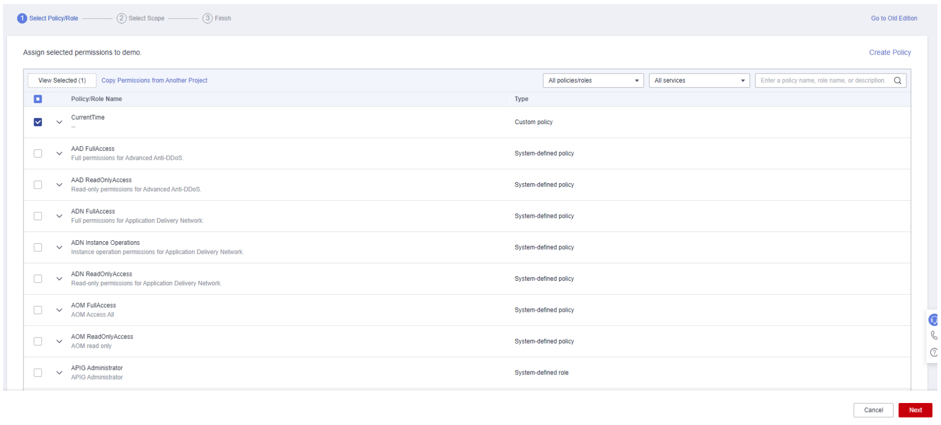


User groups available for creation: 18

Name	Users	Description	Created	Operation
demo	1	--	Jun 30, 2022 09:40:57 GMT+08:00	Authorize | Modify | Manage User | Delete
admin	2	Full permissions	Jun 21, 2022 15:05:09 GMT+08:00	Authorize | Modify | Manage User | Delete

Step 4 Select the created custom policy and click **Next**.

Figure 4-14 Selecting the custom policy

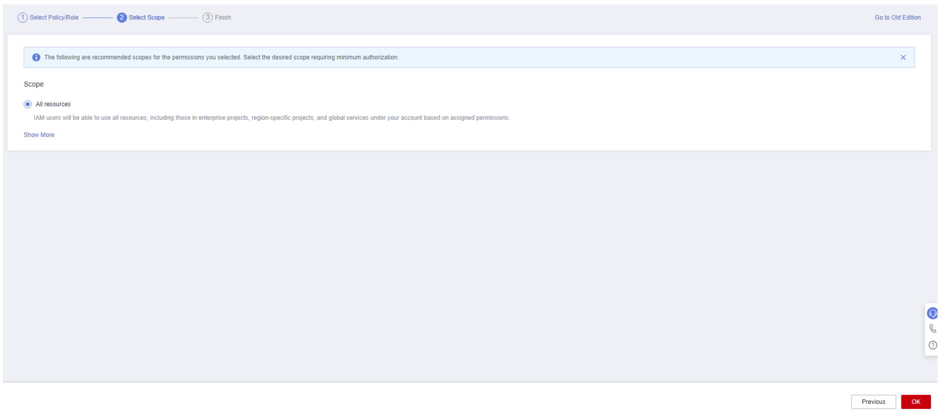


Assign selected permissions to demo.

Policy/Role Name	Type
✓ CurrentTime --	Custom policy
AAD FullAccess Full permissions for Advanced Anti-DDoS.	System-defined policy
AAD ReadOnlyAccess Read-only permissions for Advanced Anti-DDoS.	System-defined policy
ADN FullAccess Full permissions for Application Delivery Network.	System-defined policy
ADN Instance Operations Instance operation permissions for Application Delivery Network.	System-defined policy
ADN ReadOnlyAccess Read-only permissions for Application Delivery Network.	System-defined policy
AOM FullAccess AOM Access All	System-defined policy
AOM ReadOnlyAccess AOM read only	System-defined policy
APIG Administrator APIG Administrator	System-defined role

Step 5 Set the authorization scope. By default, **All resources** are selected. Then click **OK**.

Figure 4-15 Setting the authorization scope



The following are recommended scopes for the permissions you selected. Select the desired scope requiring minimum authorization.

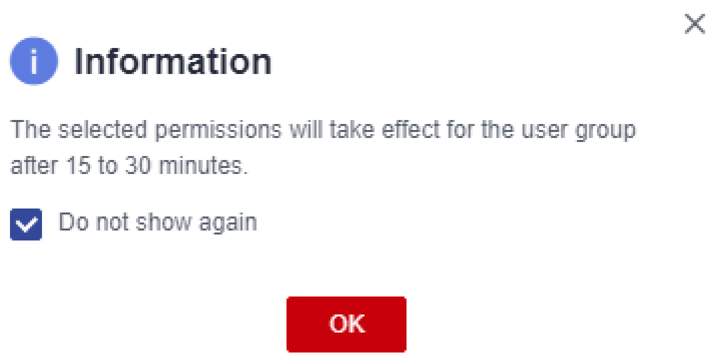
Scope

● All resources

IAM users will be able to use all resources, including those in enterprise projects, region-specific projects, and global services under your account based on assigned permissions.

Step 6 You can select **Do not show again** so that the following dialog box will not be displayed when you grant permissions to the user group in the future.

Figure 4-16 Effective time notification



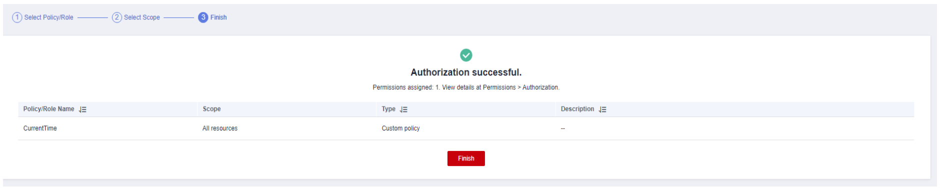
Information

The selected permissions will take effect for the user group after 15 to 30 minutes.

✓ Do not show again

Step 7 Click **Finish** to go back to the **User Groups** page.

Figure 4-17 Successful authorization



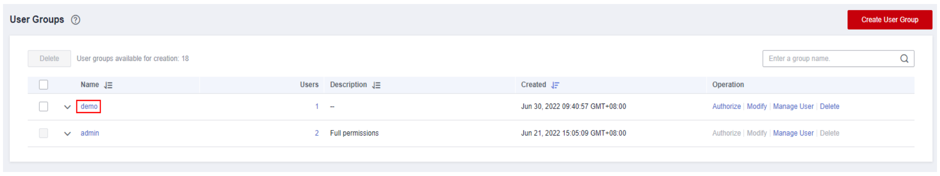
Authorization successful.

Permissions assigned: 1. View details at Permissions > Authorization.

Policy/Role Name	Scope	Type	Description
CurrentTime	All resources	Custom policy	--

Step 8 Click the name of the created user group.

Figure 4-18 User group

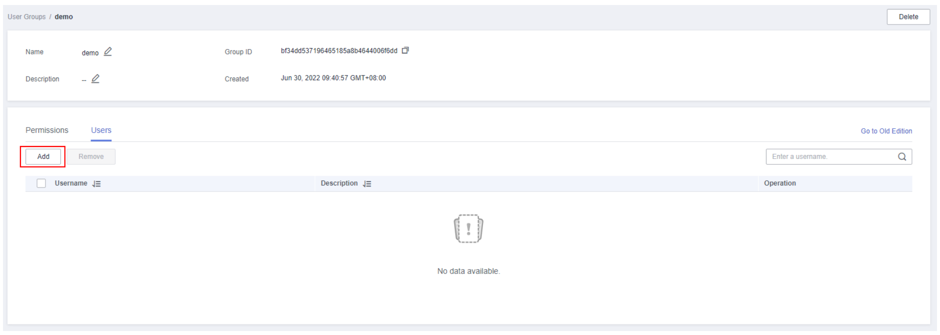


User groups available for creation: 18

Name	Users	Description	Created	Operation
demo	1	--	Jun 30, 2022 09:40:57 GMT+08:00	Authorize | Modify | Manage User | Delete
admin	2	Full permissions	Jun 21, 2022 15:05:09 GMT+08:00	Authorize | Modify | Manage User | Delete

Step 9 Click the **Users** tab and then **Add**.

Figure 4-19 Adding a user



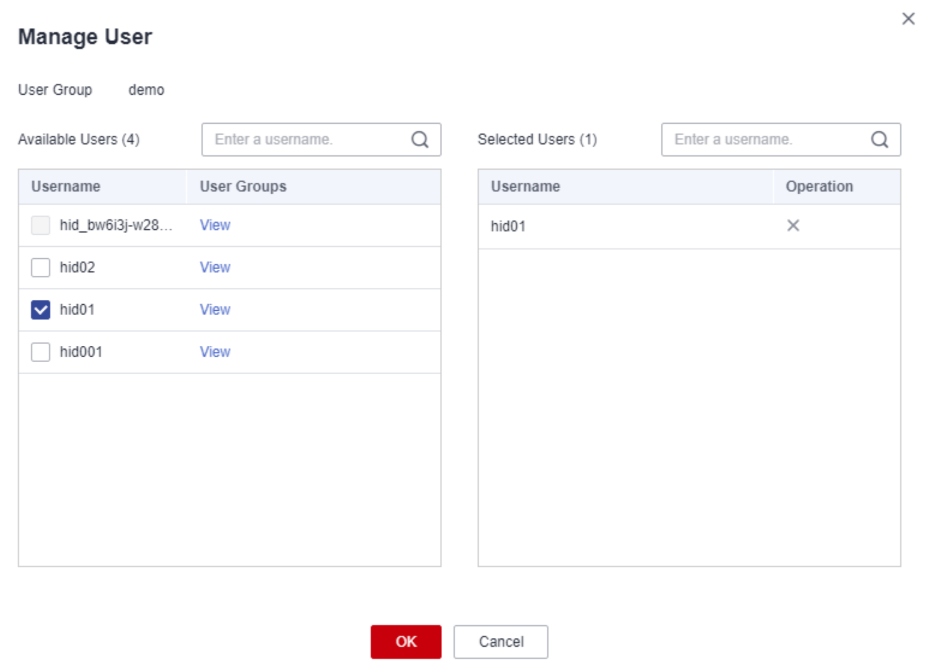
Name	demo	Group ID	bf34dd537196465185a8b4644006f6dd
Description	--	Created	Jun 30, 2022 09:40:57 GMT+08:00

Username	Description	Operation

No data available.

Step 10 Select an IAM user and click **OK**.

Figure 4-20 Selecting an IAM user



Manage User

User Group demo

Available Users (4)

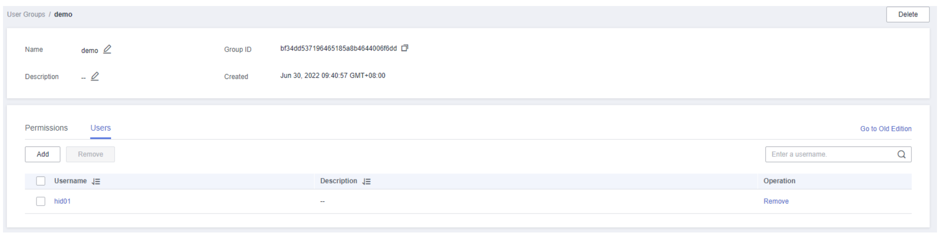
Username	User Groups
hid_bw6i3j-w28...	View
hid02	View
[✓] hid01	View
hid001	View

Selected Users (1)

Username	Operation
hid01	×

Step 11 The IAM user is added successfully, and the custom policy has been granted to the IAM user.

Figure 4-21 IAM user added



User Groups / demo

Name	demo	Group ID	bf34dd537196465185a8b4644006f6dd
Description	--	Created	Jun 30, 2022 09:40:57 GMT+08:00

Username	Description	Operation
hid01	--	Remove

----End

5 Scheduling a CDM Job by Transferring Parameters Using DataArts Factory

You can use EL expressions in DataArts Factory to transfer parameters to a CDM job to schedule it.

NOTE

- The parameter transfer function is supported by CDM 2.8.6 or later versions.
- This section uses a CDM job for migrating data from Oracle to MRS Hive as an example.

Prerequisites

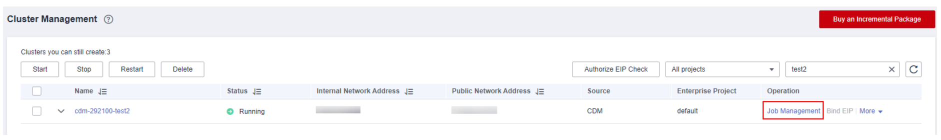
A CDM incremental package is available.

Creating a CDM Migration Job

Step 1 Log in to the console, locate an instance, click **Access**, and click **DataArts Migration**.

Step 2 On the **Cluster Management** page, click **Job Management** in the **Operation** column.

Figure 5-1 Cluster Management



Clusters you can still create:3

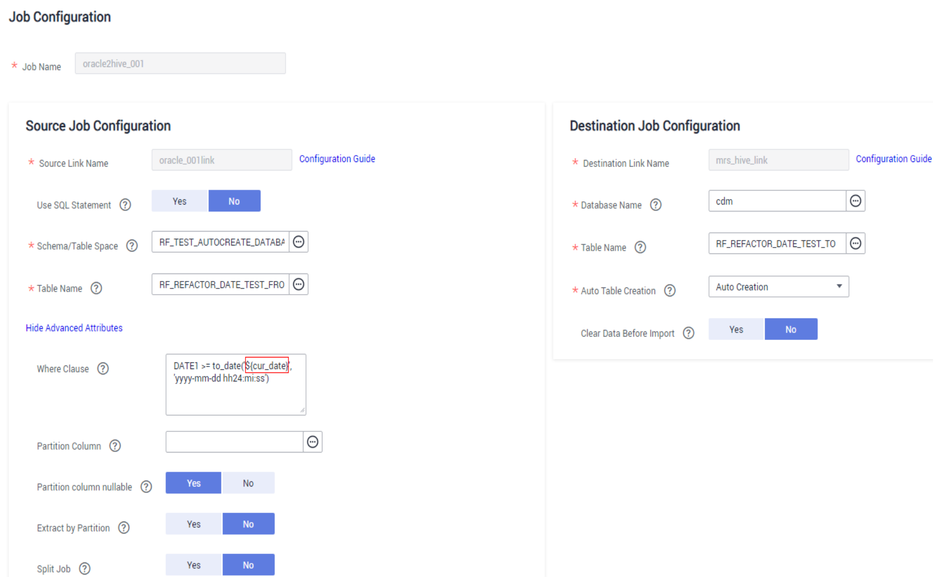
Name	Status	Internal Network Address	Public Network Address	Source	Enterprise Project	Operation
cdm-292100-test2	Running	[redacted]	[redacted]	CDM	default	Job Management Bind EIP | More

Step 3 Click the **Links** tab and then **Create Link** to create an Oracle link and an MRS Hive link. For details, see **Link to an Oracle Database** and **Link to Hive**.

Step 4 Click the **Table/File Migration** tab and then **Create Job** to create a data migration job.

Step 5 Configure parameters for the source Oracle link and destination MRS Hive link, and configure the parameter to transfer in **${*varName*}** format (**${cur_date}** in this example).

Figure 5-2 Creating a job



Job Configuration

* Job Name	oracle2hive_001

Source Job Configuration

* Source Link Name	oracle_001link
Use SQL Statement	Yes [✓ No]
* Schema/Table Space	RF_TEST_AUTOCREATE_DATABA
* Table Name	RF_REFACTOR_DATE_TEST_FRO
Where Clause	DATE1 >= to_date('${cur_date}', 'yyyy-mm-dd hh24:mi:ss')
Partition Column	
Partition column nullable	[✓ Yes] No
Extract by Partition	Yes [✓ No]
Split Job	Yes [✓ No]

Destination Job Configuration

* Destination Link Name	mrs_hive_link
* Database Name	cdm
* Table Name	RF_REFACTOR_DATE_TEST_TO
* Auto Table Creation	Auto Creation
Clear Data Before Import	Yes [✓ No]

NOTE

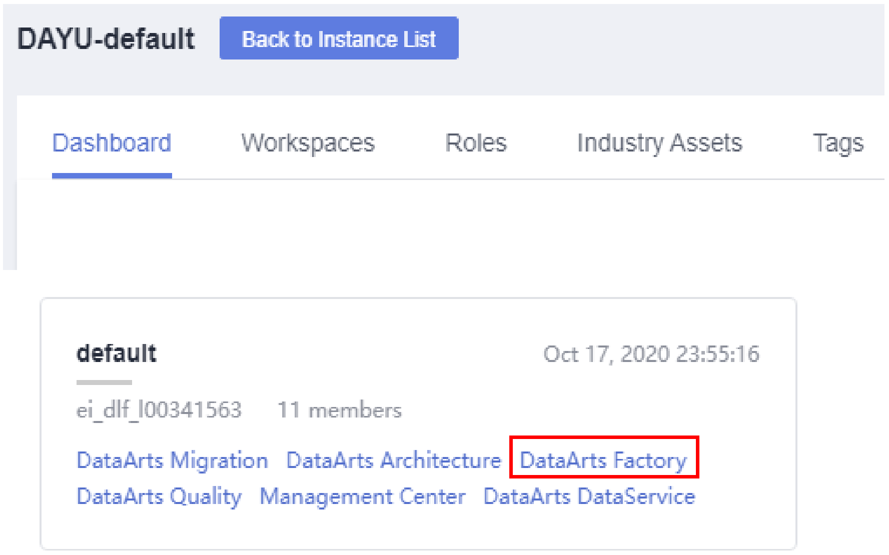
The **Retry upon Failure** parameter is unavailable in the CDM migration job. You can configure this parameter on the CDM node in DataArts Factory.

----End

Creating and Executing a Data Development Job

Step 1 Log in to the console. Locate an instance and click **Access**. On the displayed page, locate a workspace and click **DataArts Factory**.

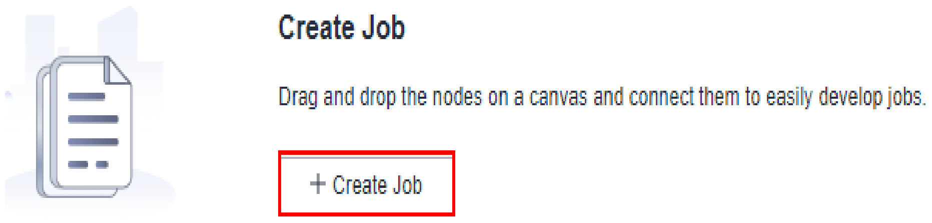
Figure 5-3 Accessing the console



Step 2 In the navigation pane of the DataArts Factory homepage, choose **Data Development** > **Develop Job**.

Step 3 On the **Develop Job** page, click **Create Job**.

Figure 5-4 Create Job


Create Job
Drag and drop the nodes on a canvas and connect them to easily develop jobs.
+ Create Job

Step 4 In the displayed dialog box, configure job parameters and click **OK**.

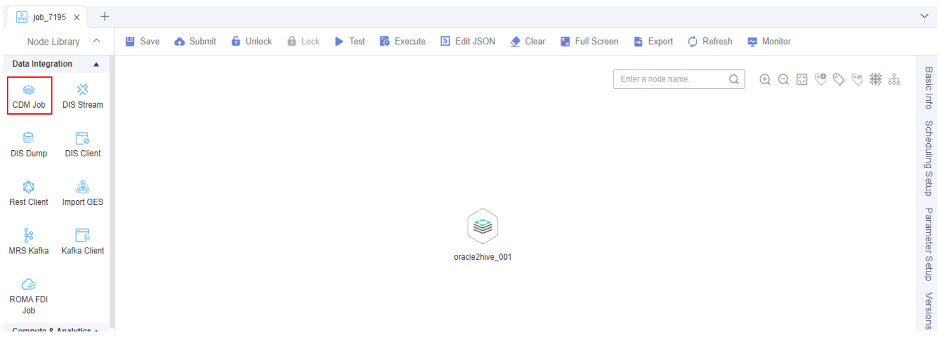
Table 5-1 Job parameters

Parameter	Description
Job Name	Name of the job. The name must contain 1 to 128 characters, including only letters, numbers, hyphens (-), underscores (_), and periods (.).
Job Type	Type of the job. ● **Batch processing**: Data is processed periodically in batches based on the scheduling plan, which is used in scenarios with low real-time requirements. This type of job is a pipeline that consists of one or more nodes and is scheduled as a whole. It cannot run for an unlimited period of time, that is, it must end after running for a certain period of time. You can configure job-level scheduling tasks for batch processing jobs. For details, see **Setting Up Scheduling for a Job Using the Batch Processing Mode**. ● **Real-time processing**: Data is processed in real time, which is used in scenarios with high real-time performance. This type of job is a business relationship that consists of one or more nodes. You can configure scheduling policies for each nodes, and the tasks started by nodes can keep running for an unlimited period of time. In this type of job, lines with arrows represent only service relationships, rather than task execution processes or data flows. You can configure node-level scheduling tasks for real-time processing jobs. For details, see **Setting Up Scheduling for Nodes of a Job Using the Real-Time Processing Mode**.
Creation Method	Job creation method ● **Create Empty Job**: Create an empty job. ● **Create Based on Template**: Use a template provided by DataArts Factory to create a job.

Parameter	Description
Select Directory	Directory to which the job belongs. The default value is the root directory.
Owner	Owner of the job
Priority	Priority of the job. The options are **High**, **Medium**, and **Low**.
Agency	After an agency is configured, the job interacts with other services as an agency during job execution. **NOTE** A job-level agency takes precedence over a workspace-level agency.
Log Path	Path of the OBS bucket for storing job logs. By default, logs are stored in an OBS bucket named **dlf-log-**{*Projectid*}. **NOTE** ● If you want to customize a storage path, select the bucket that you have created on OBS by following the instructions provided in **(Optional) Changing a Job Log Storage Path**. ● Ensure that you have the read and write permissions on the OBS bucket specified by this parameter, or the system cannot write or display logs.

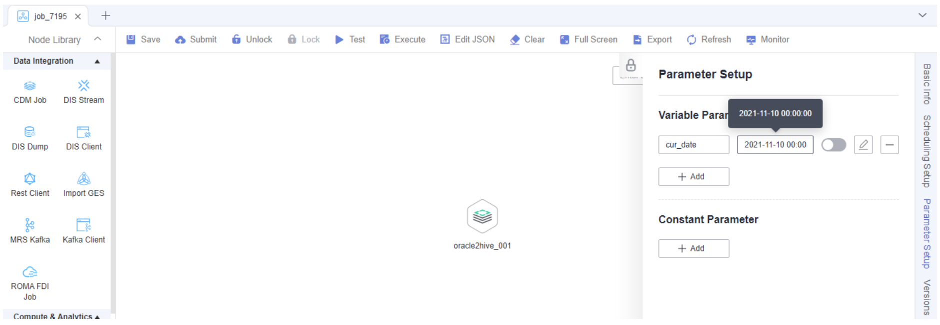
Step 5 Add a CDM Job node in the data development job and associate the node with the created CDM job.

Figure 5-5 Associating the CDM Job node with the created CDM job



Step 6 Configure the parameter to be transferred to the CDM job.

Figure 5-6 Configuring the parameter to be transferred



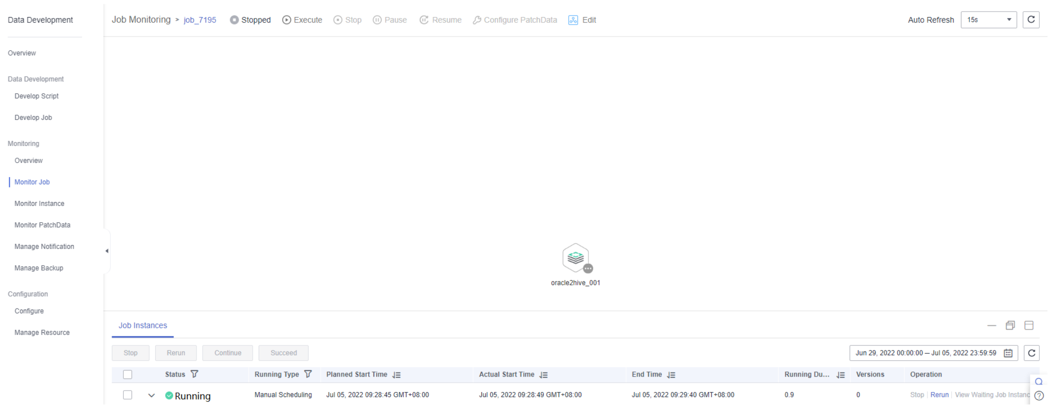
NOTE

When the job is scheduled and executed, the value of the configured parameter will be transferred to the CDM job. The value of the parameter **cur_date** can be set to a fixed value (for example, **2021-11-10 00:00:00**) or an EL expression (for example, **#{DateUtil.format(DateUtil.addDays(Job.planTime,-1),"yyyy-MM-dd")}** which means the day before the scheduled job execution date. For more EL expressions, see **EL expressions**.

Step 7 Save and submit a job version and click **Test** to execute the data development job.

Step 8 After the data development job is executed, click **Monitor** in the upper right corner to go to the **Monitor Job** page and check whether the generated task or instance meets requirements.

Figure 5-7 Viewing the execution result



	Status	Running Type	Planned Start Time	Actual Start Time	End Time	Running Du...	Versions	Operation
	Running	Manual Scheduling	Jul 05, 2022 09:28:45 GMT+08:00	Jul 05, 2022 09:28:49 GMT+08:00	Jul 05, 2022 09:29:40 GMT+08:00	0.9	0	Stop | Rerun | View Waiting Job Instanc

----End

6 Incremental Migration on CDM Supported by DLF

The Data Lake Factory (DLF) component of DataArts Studio is a one-stop big data collaboration development platform. With DLF's online script editing, CDM jobs can be scheduled to implement incremental migration.

This section describes how DLF works with CDM to implement incremental migration from DWS to OBS.

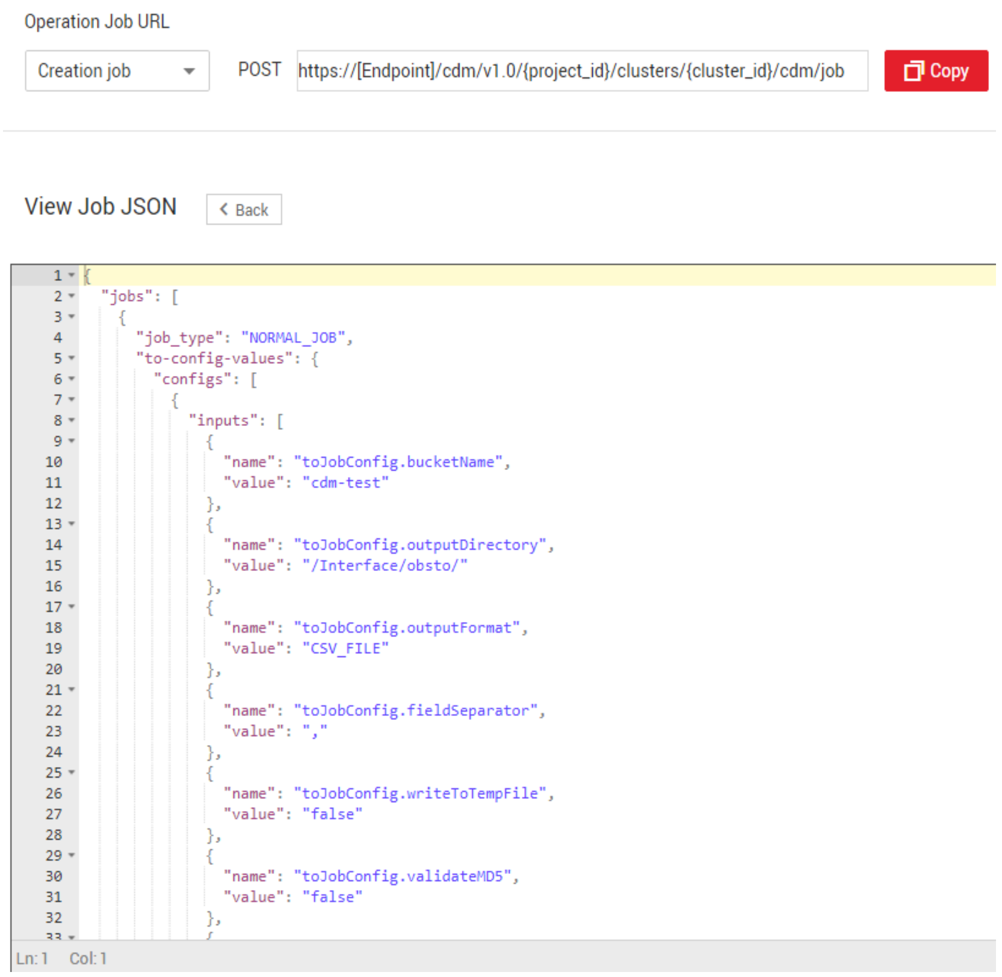
1. **Obtaining the CDM Job JSON**
2. **Modifying JSON**
3. **Creating a DLF Job**

Obtaining the CDM Job JSON

1. On the CDM console, create a table/file migration job from DWS to OBS.
2. On the **Table/File Migration** tab page of the **Job Management** page, locate the created job, click **More** in the **Operation** column, and select **View Job JSON** from the drop-down list.

   You can also view JSON of any other CDM job.

Figure 6-1 Viewing job JSON



https://[Endpoint]/cdm/v1.0/{project_id}/clusters/{cluster_id}/cdm/job

```
{
  "jobs": [
    {
      "job_type": "NORMAL_JOB",
      "to-config-values": {
        "configs": [
          {
            "inputs": [
              {
                "name": "toJobConfig.bucketName",
                "value": "cdm-test"
              },
              {
                "name": "toJobConfig.outputDirectory",
                "value": "/Interface/obsto/"
              },
              {
                "name": "toJobConfig.outputFormat",
                "value": "CSV_FILE"
              },
              {
                "name": "toJobConfig.fieldSeparator",
                "value": ","
              },
              {
                "name": "toJobConfig.writeToTempFile",
                "value": "false"
              },
              {
                "name": "toJobConfig.validateMD5",
                "value": "false"
              },
              {
```

3. The job JSON is the request body template for creating a CDM job. Replace **[Endpoint]**, **{project_id}**, and **{cluster_id}** in the URL with the actual values.
   – [Endpoint]: indicates the endpoint.

     An endpoint is the **request address** for calling an API. Endpoints vary depending on services and regions. You can obtain endpoints of the service from **Endpoints**.
   – **{project_id}**: indicates the project ID.
   – **{cluster_id}**: Indicates the cluster ID. You can click the cluster name on the **Cluster Management** page to view the cluster ID.

Modifying JSON

You can modify the JSON body as required. In this example, the period is one day, and the WHERE clause is used for filtering the incremental data to be migrated (generally, the time range is used for filtering data). The data generated on the previous day is migrated every day.

1. Modify the WHERE clause to add incremental data in a certain period.

```
{
  "name": "fromJobConfig.whereClause",
  "value": "_timestamp >= '${startTime}' and _timestamp < '${currentTime}'"
}
```

NOTE

- If the source database is DWS or MySQL, the value can be set to:
  ```
  _timestamp >= '2018-10-10 00:00:00' and _timestamp < '2018-10-11 00:00:00'
  Or
  _timestamp between '2018-10-10 00:00:00' and '2018-10-11 00:00:00'
  ```
- If the source database is Oracle, the value should be set to:
  ```
  _timestamp >= to_date (2018-10-10 00:00:00' , 'yyyy-mm-dd hh24:mi:ss' ) and _timestamp <
  to_date (2018-10-10 00:00:00' , 'yyyy-mm-dd hh24:mi:ss' )
  ```

2. Import incremental data in each period to different directories.
   ```
     {
       "name": "toJobConfig.outputDirectory",
       "value": "dws2obs/${currentTime}"
     }
   ```
3. Change the job name to a dynamic one. Otherwise, the job cannot be created because the job name is duplicate.
   ```
     "to-connector-name": "obs-connector",
     "from-link-name": "dws_link",
     "name": "dws2obs-${currentTime}"
   ```

For details about how to modify more parameters, see ***Cloud Data Migration API Reference***. The following is an example of the modified JSON file:

```
{
  "jobs": [
    {
      "job_type": "NORMAL_JOB",
      "to-config-values": {
        "configs": [
          {
            "inputs": [
              {
                "name": "toJobConfig.bucketName",
                "value": "cdm-test"
              },
              {
                "name": "toJobConfig.outputDirectory",
                "value": "dws2obs/${currentTime}"
              },
              {
                "name": "toJobConfig.outputFormat",
                "value": "CSV_FILE"
              },
              {
                "name": "toJobConfig.fieldSeparator",
                "value": ","
              },
              {
                "name": "toJobConfig.writeToTempFile",
                "value": "false"
              },
              {
                "name": "toJobConfig.validateMD5",
                "value": "false"
              },
              {
                "name": "toJobConfig.encodeType",
                "value": "UTF-8"
              },
              {
                "name": "toJobConfig.duplicateFileOpType",
                "value": "REPLACE"
              },
              {
                "name": "toJobConfig.kmsEncryption",
                "value": "false"
              }
```

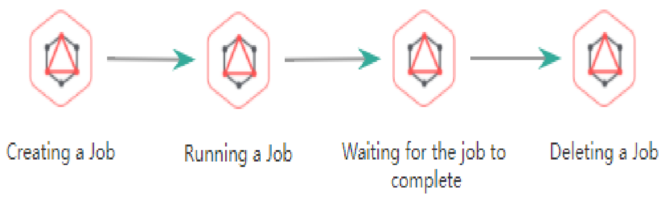
```
          ],
          "name": "toJobConfig"
        }
      ]
    },
    "from-config-values": {
      "configs": [
        {
          "inputs": [
            {
              "name": "fromJobConfig.schemaName",
              "value": "dws_database"
            },
            {
              "name": "fromJobConfig.tableName",
              "value": "dws_from"
            },
            {
              "name": "fromJobConfig.whereClause",
              "value": "_timestamp >= '${startTime}' and _timestamp < '${currentTime}'"
            },
            {
              "name": "fromJobConfig.columnList",
              "value": 
"_tiny&_small&_int&_integer&_bigint&_float&_double&_date&_timestamp&_char&_varchar&_text"
            }
          ],
          "name": "fromJobConfig"
        }
      ]
    },
    "from-connector-name": "generic-jdbc-connector",
    "to-link-name": "obs_link",
    "driver-config-values": {
      "configs": [
        {
          "inputs": [
            {
              "name": "throttlingConfig.numExtractors",
              "value": "1"
            },
            {
              "name": "throttlingConfig.submitToCluster",
              "value": "false"
            },
            {
              "name": "throttlingConfig.numLoaders",
              "value": "1"
            },
            {
              "name": "throttlingConfig.recordDirtyData",
              "value": "false"
            },
            {
              "name": "throttlingConfig.writeToLink",
              "value": "obs_link"
            }
          ],
          "name": "throttlingConfig"
        },
        {
          "inputs": [],
          "name": "jarConfig"
        },
        {
          "inputs": [],
          "name": "schedulerConfig"
        },
        {
          "inputs": [],
          "name": "transformConfig"
        },
        {
          "inputs": [],
          "name": "smnConfig"
        },
        {
          "inputs": [],
          "name": "retryJobConfig"
        }
      ]
    },
    "to-connector-name": "obs-connector",
    "from-link-name": "dws_link",
    "name": "dws2obs-${currentTime}"
  }
 ]
}
```

Creating a DLF Job

1. In DLF, create the jobs shown in **Figure 6-2**. For details, see **Creating a Job** in *DataArts Studio User Guide*.

   For details about how to configure the nodes and the job, see the following steps.

   Figure 6-2 DLF job

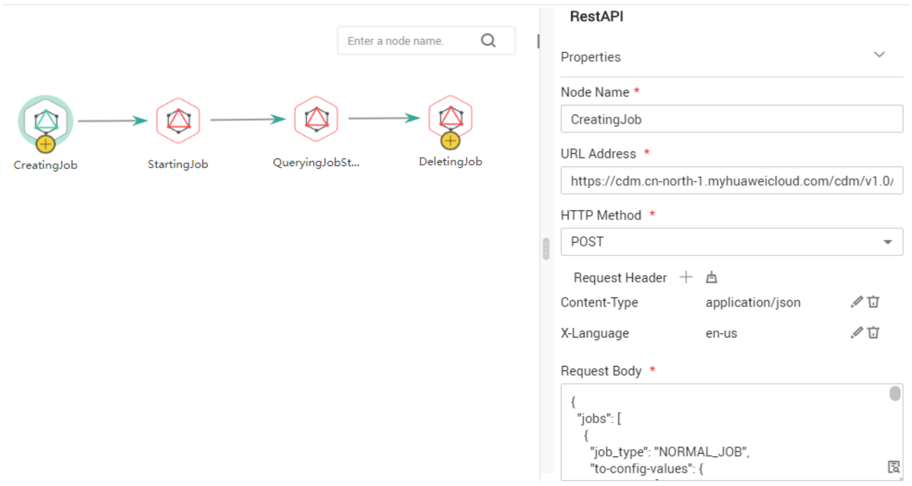
   

2. Configure the **CreatingJob** node.

   DLF uses a RestAPI node to call a RESTful API to create a CDM migration job. Configure the properties of the RestAPI node.

   a. **Node Name**: Enter a custom name, for example, **CreatingJob**. Note that the CDM job is only used as a node in the DLF job.
   b. **URL Address**: Set it to the URL obtained in **Obtaining the CDM Job JSON**. The format is https://{*Endpoint*}/cdm/v1.0/{*project_id*}/clusters/{*cluster_id*}/cdm/job.
   c. **HTTP Method**: Enter **POST**.
   d. Add the following request headers:
      - Content-Type = application/json
      - X-Language = en-us
   e. **Request Body**: Enter the modified JSON of the CDM job in **Modifying JSON**.

Figure 6-3 Properties of the node for creating the CDM job



Node Name *: CreatingJob
URL Address *: https://cdm.cn-north-1.myhuaweicloud.com/cdm/v1.0/
HTTP Method *: POST

Request Header

Content-Type	application/json
X-Language	en-us

Request Body *

```
{
  "jobs": [
   {
    "job_type": "NORMAL_JOB",
    "to-config-values": {
```

3. Configure the **StartingJob** node.

   After configuring the RESTful API node for creating a CDM job, you must add the RESTful API node for running the CDM job. For details, see section "Starting a Job" in ***Cloud Data Migration API Reference***. Configure the properties of the RestAPI node.

   a. **Node Name**: Enter the name of the node where the job is to be run.

   b. **URL Address**: Keep the values of **project_id** and **cluster_id** consistent with those in 2. Set the job name to **dws2obs-${currentTime}**. The format is https://{*Endpoint*}/cdm/v1.0/{*project_id*}/clusters/{*cluster_id*}/cdm/job/{*job_name*}/start.

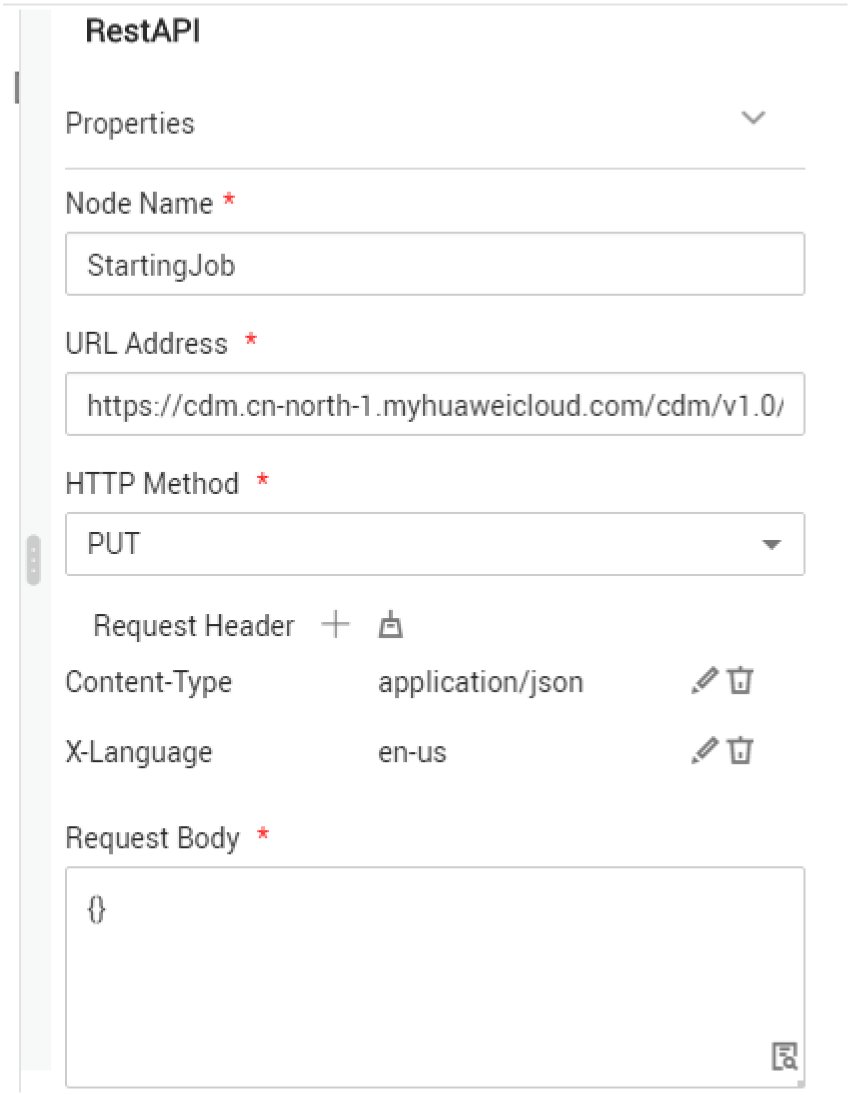
   c. **HTTP Method**: Enter **PUT**.

   d. **Request Header**:

      ▪ Content-Type = application/json

      ▪ X-Language = en-us

Figure 6-4 Properties of the node for running the CDM job



RestAPI

Properties

Node Name *

StartingJob

URL Address *

https://cdm.cn-north-1.myhuaweicloud.com/cdm/v1.0/

HTTP Method *

PUT

Request Header

Content-Type	application/json
X-Language	en-us

Request Body *

```
{}
```

4. Configure the **WaitingJobCompletion** node.

   CDM jobs are run asynchronously. Therefore, even if the REST request for running the job returns 200, it does not mean that the data has been migrated successfully. If a computing job depends on the CDM job, a RestAPI node is required to periodically check whether the migration is successful. Computing is performed only when the migration is successful. For details about the API used to check whether the CDM migration is successful, see section "Querying Job Status" in ***Cloud Data Migration API Reference***.

   After configuring the RestAPI node for running the CDM job, add the node for waiting for the CDM job completion. The node properties are as follows:

   a. Node Name: Wait until the job is complete.
   b. **URL Address**: The format is https://{*Endpoint*}/cdm/v1.0/{*project_id*}/clusters/{*cluster_id*}/cdm/job/{*job_name*}/status. Keep the values of **project_id** and **cluster_id** consistent with those in 2. Set the job name to **dws2obs-${currentTime}**.
   c. **HTTP Method**: Enter **GET**.

d. **Request Header**:
   - Content-Type = application/json
   - X-Language = en-us
e. **Check Return Value**: Select **YES**.
f. **Property Path**: Enter **submissions[0].status**.
g. **Request Success Flag**: Set this parameter to **SUCCEEDED**.
h. Retain default values for other parameters.

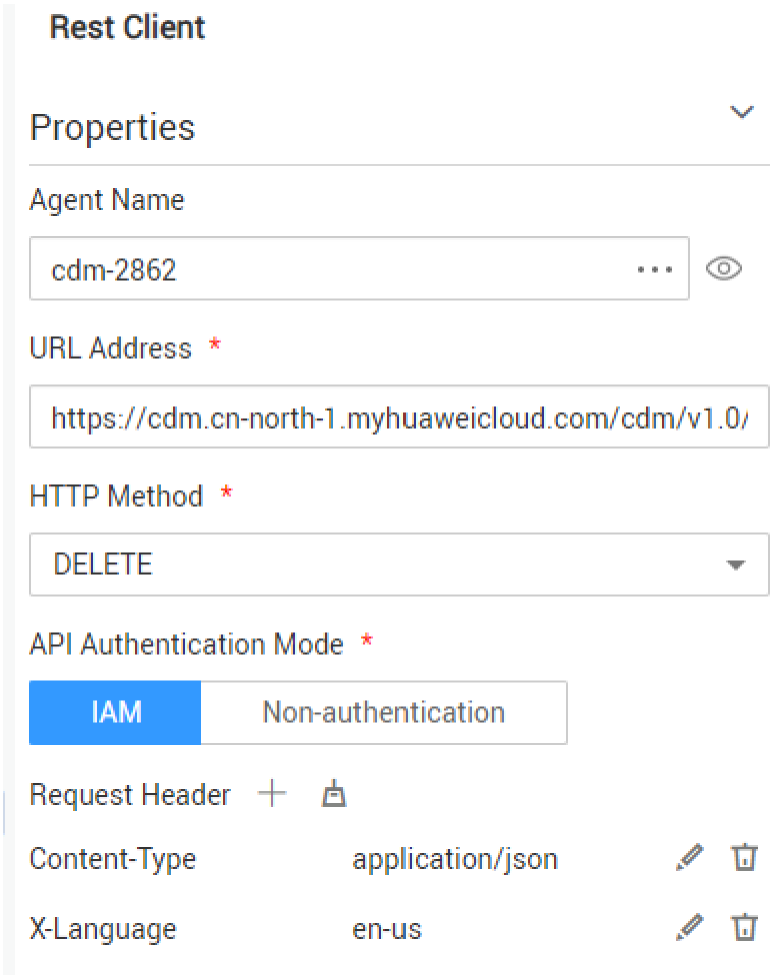
5. (Optional) Configure the **DeletingJob** node.

   You can delete jobs as required. DLF periodically creates CDM jobs to implement incremental migration. Therefore, a large number of jobs exist in the CDM cluster. After the migration is successful, you can delete the jobs that have been successfully executed. To delete a CDM job, add a RestAPI node for deleting CDM jobs after the node for querying the CDM job status. DLF calls the API for deleting a job described in ***Cloud Data Migration API Reference***.

   Properties of the node for deleting the CDM job are as follows:

   a. **Node Name**: Enter **DeletingJob**.
   b. **URL Address**: The format is https://{*Endpoint*}/cdm/v1.0/{*project_id*}/clusters/{*cluster_id*}/cdm/job/{*job_name*}. Keep the values of **project_id** and **cluster_id** consistent with those in **2**. Set the job name to **dws2obs-${currentTime}**.
   c. **HTTP Method**: Enter **DELETE**.
   d. **Request Header**:
      - Content-Type = application/json
      - X-Language = en-us
   e. Retain default values for other parameters.

Figure 6-5 Properties of the node for deleting the CDM job


Rest Client

Properties

Agent Name: cdm-2862

URL Address *: https://cdm.cn-north-1.myhuaweicloud.com/cdm/v1.0/

HTTP Method *: DELETE

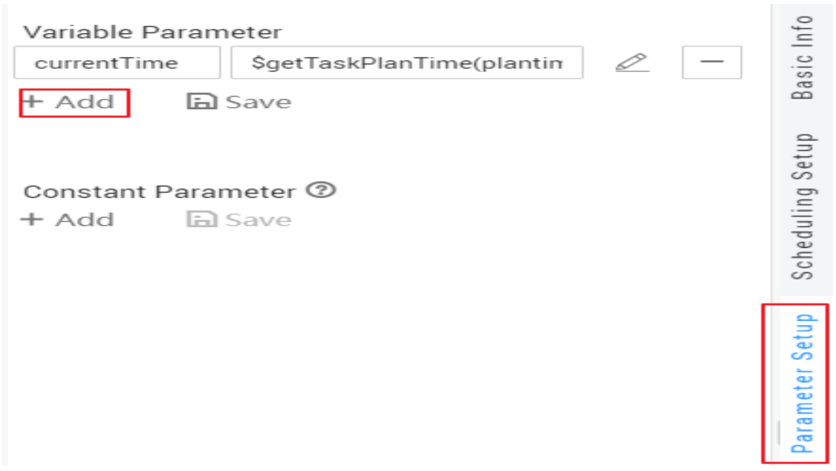
API Authentication Mode *: IAM | Non-authentication

Request Header

Content-Type	application/json
X-Language	en-us

6. To perform computing operations after the migration is complete, you can add various computing nodes.
7. Configure DLF job parameters.
   a. Configure the DLF job parameters shown in **Figure 6-6**.
      - startTime = $getTaskPlanTime(plantime,@@yyyyMMddHHmmss@@,-24*60*60)
      - currentTime = $getTaskPlanTime(plantime,@@yyyyMMdd-HHmm@@,0)

Figure 6-6 Configuring DLF job parameters



b. After saving the DLF job, choose **Scheduling Configuration** > **Periodic Scheduling** and set the scheduling period to one day.

In this way, DLF works with CDM to migrate data generated on the previous day every day.

7 Creating Table Migration Jobs in Batches Using CDM Nodes

Scenario

In a service system, data sources are usually stored in different tables to reduce the size of a single table in complex application scenarios.

In this case, you need to create a data migration job for each table when using CDM to integrate data. This tutorial describes how to use the For Each and CDM nodes provided by the DataArts Factory module to create table migration jobs in batches.

In this tutorial, the source MySQL database has three tables, mail01, mail02, and mail03. The tables have the same structure but different data content. The destination is MRS Hive.

Prerequisites

- You have created a CDM cluster.
- MRS Hive has been enabled.
- Databases and tables have been created in MRS Hive.

Creating a Link

Step 1 Log in to the DataArts Studio console, locate the target DataArts Studio instance, and click **Access** on the instance card.

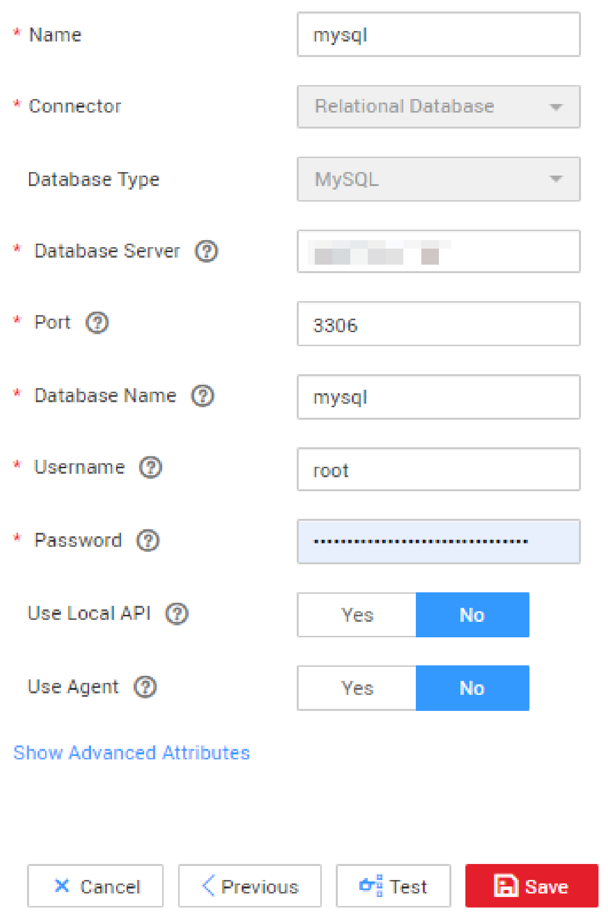
Step 2 Locate a workspace and click **DataArts Migration**.

Step 3 In the **Operation** column, click **Job Management**.

Step 4 Click the **Links** tab and then **Driver Management**. Upload the MySQL database driver by following the instructions in **Managing Drivers**.

Step 5 Click the **Links** tab and then **Create Link**. Select **MySQL** and click **Next** to configure parameters for the link. After the configuration is complete, click **Save** to return to the **Links** page.

Figure 7-1 Configuring the MySQL link



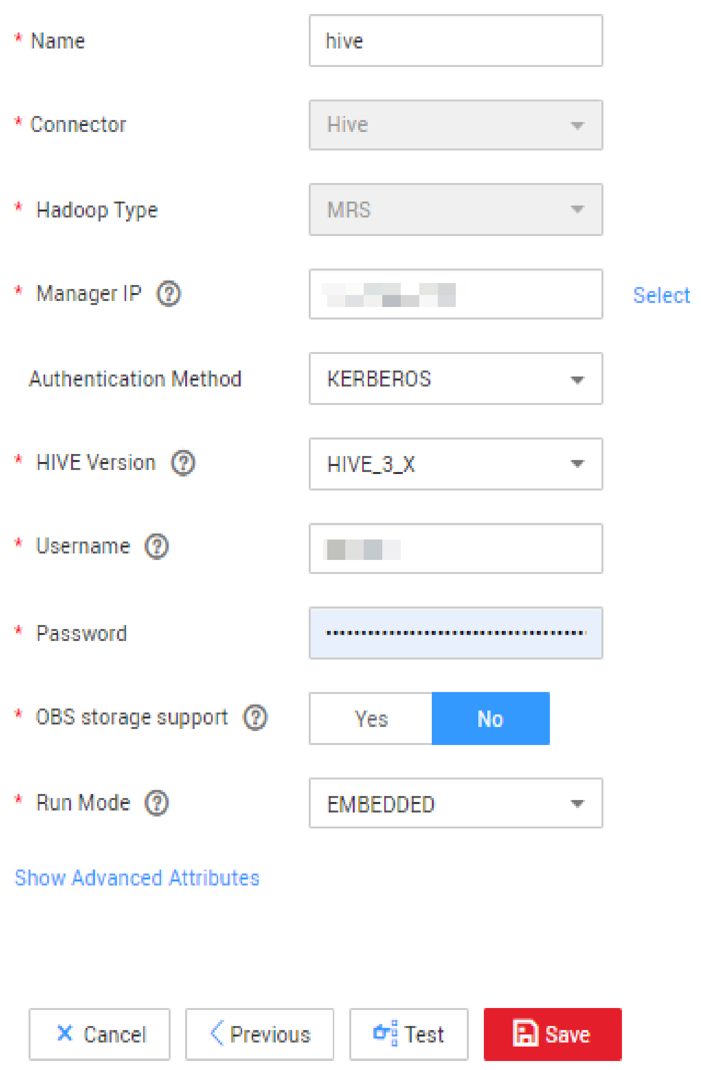
* Name	mysql
* Connector	Relational Database
Database Type	MySQL
* Database Server	[redacted]
* Port	3306
* Database Name	mysql
* Username	root
* Password	••••••••••••••••••••••••••••••••••
Use Local API	Yes [✓ No]
Use Agent	Yes [✓ No]

Table 7-1 MySQL link parameters

Parameter	Description	Example
Name	Link name, which should be defined based on the data source type, so it is easier to remember what the link is for	mysql
Database Server	IP address or domain name of the database to connect	192.168.0.1
Port Number	Port of the database to connect	3306
Database	Name of the database to connect	mysql
Username	Username used for accessing the database This account must have the permissions required to read and write data tables and metadata.	root
Password	Password of the username	-
Use Agent	Whether to extract data from the data source through an agent	Disabled

Step 6 Click the **Links** tab and then **Create Link**. Select **MRS Hive** and click **Next** to configure parameters for the link. After the configuration is complete, click **Save** to return to the **Links** page.

Figure 7-2 Configuring the MRS Hive link



* Name	hive
* Connector	Hive
* Hadoop Type	MRS
* Manager IP	[redacted]
Authentication Method	KERBEROS
* HIVE Version	HIVE_3_X
* Username	[redacted]
* Password	••••••••••••••••••••••••••••••••••••
* OBS storage support	Yes [✓ No]
* Run Mode	EMBEDDED

Table 7-2 MRS Hive link parameters

Parameter	Remarks	Example
Metric Name	Link name, which should be defined based on the data source type, so it is easier to remember what the link is for	hive
Manager IP	Floating IP address of MRS Manager. Click **Select** next to the **Manager IP** text box to select an MRS cluster. CDM automatically fills in the authentication information.	127.0.0.1

Parameter	Remarks	Example
Authentication Method	Authentication method used for accessing MRS ● **SIMPLE**: Select this for non-security mode. ● **KERBEROS**: Select this for security mode.	KERBEROS
Hive Version	Hive version. Set it to the Hive version on the server.	HIVE_3_X
Username	If **Authentication Method** is set to **KERBEROS**, you must provide the username and password used for logging in to MRS Manager. If you need to create a snapshot when exporting a directory from HDFS, the user configured here must have the administrator permission on HDFS. To create a data connection for an MRS security cluster, do not use user **admin**. The **admin** user is the default management page user and cannot be used as the authentication user of the security cluster. You can create an MRS user and set **Username** and **Password** to the username and password of the created MRS user when creating an MRS data connection. **NOTE** ● If the CDM cluster version is 2.9.0 or later and the MRS cluster version is 3.1.0 or later, the created user must have the permissions of the **Manager_viewer** role to create links on CDM. To perform operations on databases, tables, and data of a component, you also need to add the user group permissions of the component to the user. ● If the CDM cluster version is earlier than 2.9.0 or the MRS cluster version is earlier than 3.1.0, the created user must have the permissions of **Manager_administrator**, **Manager_tenant**, or **System_administrator** to create links on CDM.	cdm
Password	Password for logging in to MRS Manager	-
OBS storage support	The server must support OBS storage. When creating a Hive table, you can store the table in OBS.	Disabled

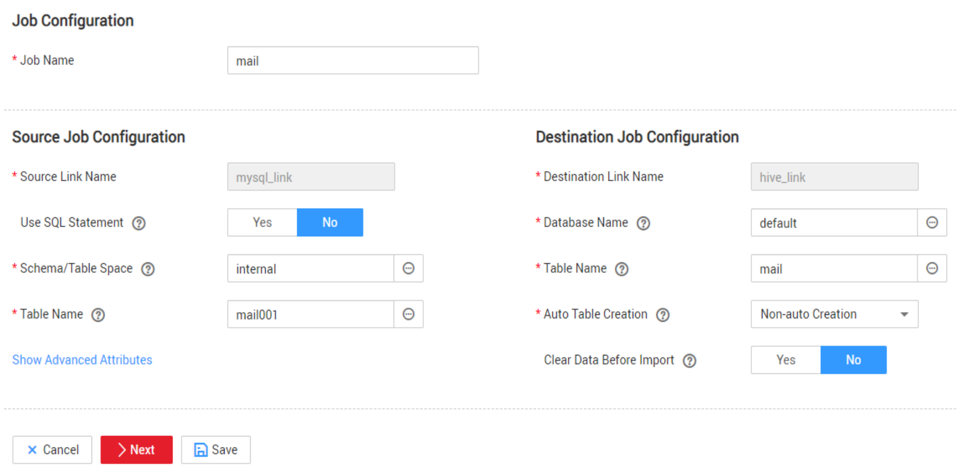
Parameter	Remarks	Example
Run Mode	This parameter is used only when the Hive version is **HIVE_3_X**. Possible values are: ● **EMBEDDED**: The link instance runs with CDM. This mode delivers better performance. ● **STANDALONE**: The link instance runs in an independent process. If CDM needs to connect to multiple Hadoop data sources (MRS, Hadoop, or CloudTable) with both Kerberos and Simple authentication modes, select **STANDALONE** or configure different agents. Note: The **STANDALONE** mode is used to solve the version conflict problem. If the connector versions of the source and destination ends of the same link are different, a JAR file conflict occurs. In this case, you need to place the source or destination end in the STANDALONE process to prevent the migration failure caused by the conflict.	EMBEDDED
Use Cluster Config	You can use the cluster configuration to simplify parameter settings for the Hadoop connection.	Disabled

----End

Creating a Sample Job

Step 1 Create a job for migrating the first MySQL table **mail001** to the MRS Hive table **mail**.

Note: Select **Do not Delete** for the **Delete Job After Completion** parameter.



Job Configuration

* Job Name	mail

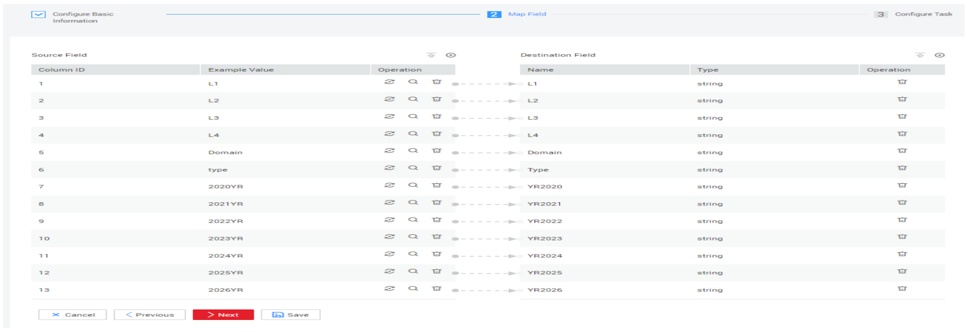
Source Job Configuration

* Source Link Name	mysql_link
Use SQL Statement	Yes [✓ No]
* Schema/Table Space	internal
* Table Name	mail001

Show Advanced Attributes

Destination Job Configuration

* Destination Link Name	hive_link
* Database Name	default
* Table Name	mail
* Auto Table Creation	Non-auto Creation
Clear Data Before Import	Yes [✓ No]

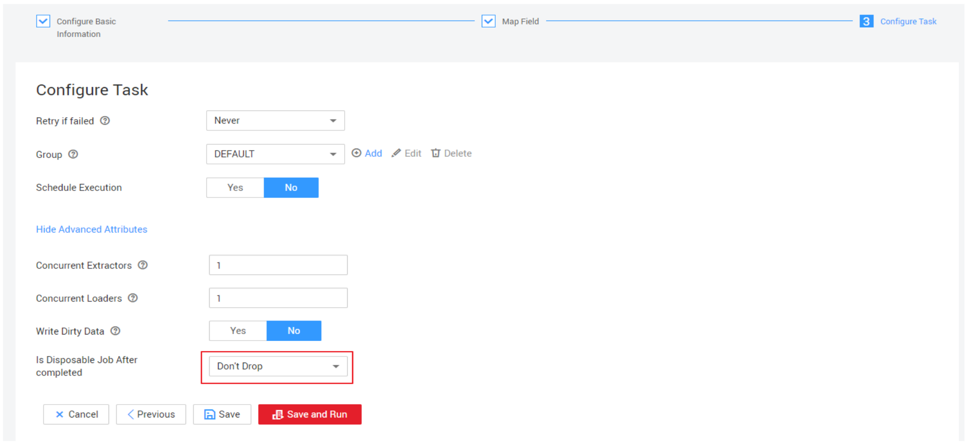


Source Field

Column ID	Example Value	Operation
1	L1	
2	L2	
3	L3	
4	L4	
5	Domain	
6	type	
7	2020YR	
8	2021YR	
9	2022YR	
10	2023YR	
11	2024YR	
12	2025YR	
13	2026YR	

Destination Field

Name	Type	Operation
L1	string	
L2	string	
L3	string	
L4	string	
Domain	string	
Type	string	
YR2020	string	
YR2021	string	
YR2022	string	
YR2023	string	
YR2024	string	
YR2025	string	
YR2026	string	



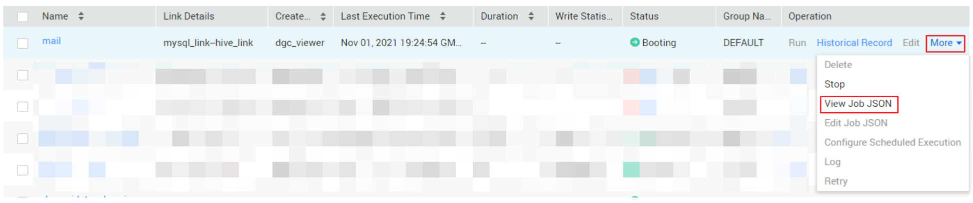
Configure Task

Retry if failed	Never
Group	DEFAULT
Schedule Execution	Yes [✓ No]

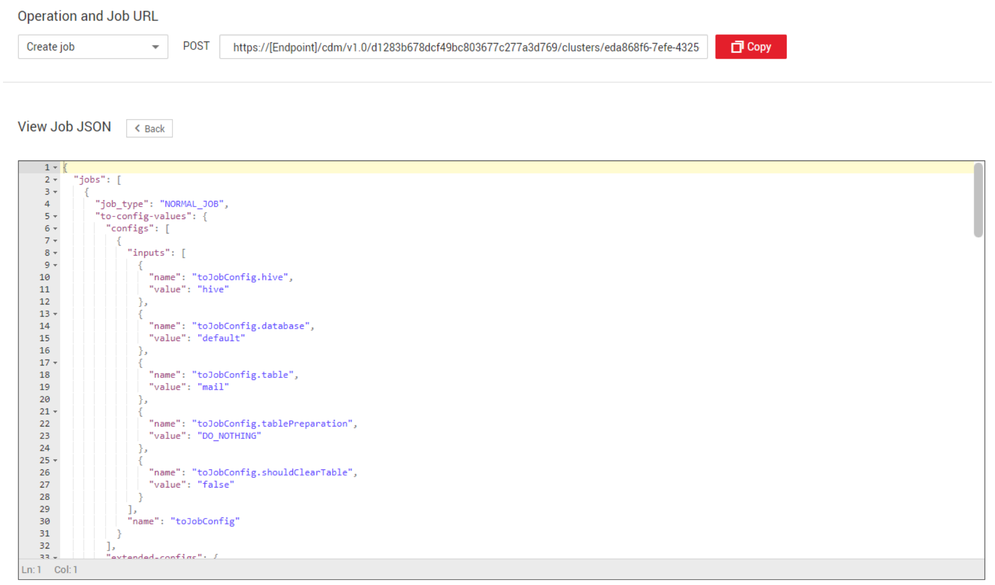
Hide Advanced Attributes

Concurrent Extractors	1
Concurrent Loaders	1
Write Dirty Data	Yes [✓ No]
Is Disposable Job After completed	Don't Drop

Step 2 After the sample job is created, view and copy the job JSON for subsequent configuration of data development jobs.



Name	Link Details	Create...	Last Execution Time	Duration	Write Statis...	Status	Group Na...	Operation
mail	mysql_link–hive_link	dgc_viewer	Nov 01, 2021 19:24:54 GM...	--	--	Booting	DEFAULT	Run Historical Record Edit More


Operation and Job URL

Create job POST https://[Endpoint]/cdm/v1.0/d1283b678dcf49bc803677c277a3d769/clusters/eda868f6-7efe-4325

View Job JSON

```
{
  "jobs": [
    {
      "job_type": "NORMAL_JOB",
      "to-config-values": {
        "configs": [
          {
            "inputs": [
              {
                "name": "toJobConfig.hive",
                "value": "hive"
              },
              {
                "name": "toJobConfig.database",
                "value": "default"
              },
              {
                "name": "toJobConfig.table",
                "value": "mail"
              },
              {
                "name": "toJobConfig.tablePreparation",
                "value": "DO_NOTHING"
              },
              {
                "name": "toJobConfig.shouldClearTable",
                "value": "false"
              }
            ],
            "name": "toJobConfig"
          }
        ],
        "extended-configs": {
```

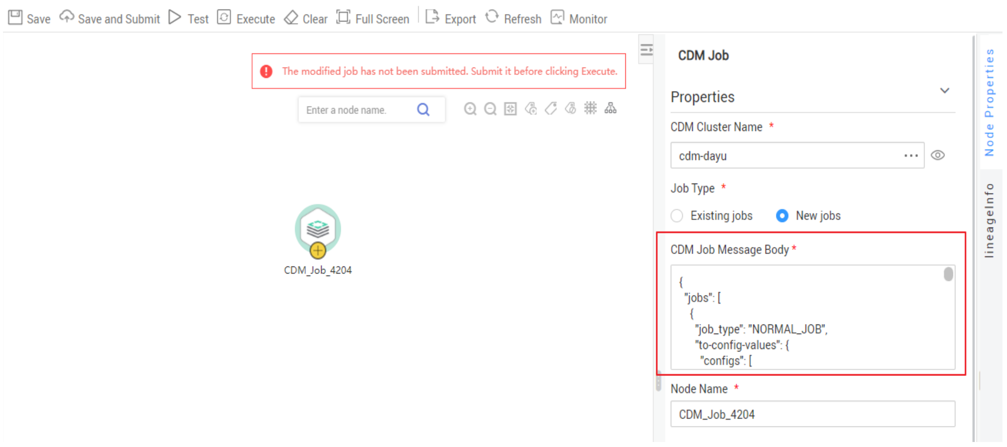
Ln: 1 Col: 1

----End

Creating a Data Development Job

Step 1 Locate a workspace and click **DataArts Factory**.

Step 2 Create a subjob named **table**, select the CDM node, select **New jobs** for **Job Type** in **Properties**, and copy and paste the JSON file in Step 2 to the **CDM Job Message Body**.



The modified job has not been submitted. Submit it before clicking Execute.

CDM Job

Properties

CDM Cluster Name * cdm-dayu

Job Type * Existing jobs [✓ New jobs]

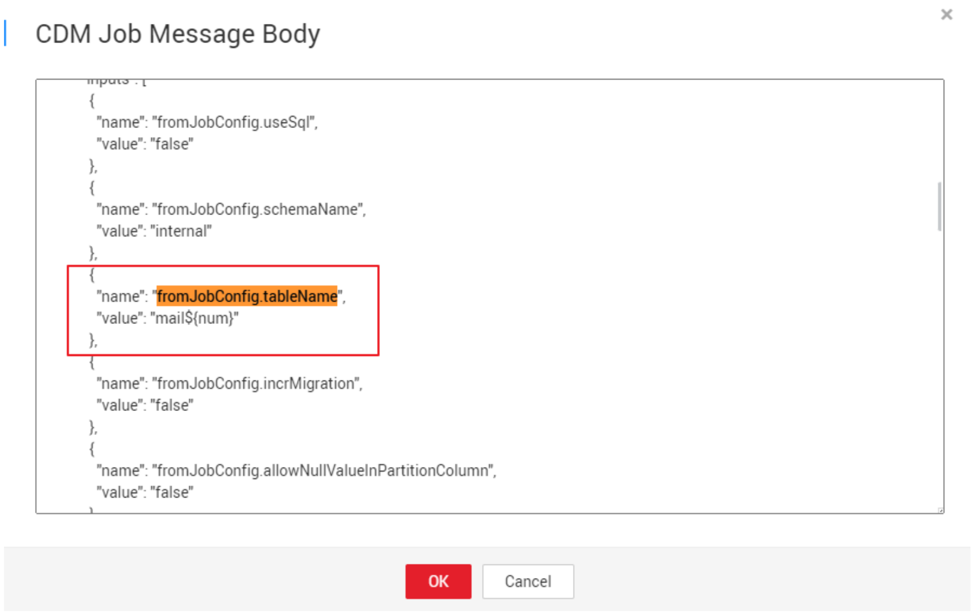
CDM Job Message Body *

```
{
 "jobs": [
  {
   "job_type": "NORMAL_JOB",
   "to-config-values": {
    "configs": [
```

Node Name * CDM_Job_4204

Step 3 Edit the CDM job message body.

1. Since there are three source tables **mail001**, **mail002**, and **mail003**, you need to set **fromJobConfig.tableName** to **mail${num}** in the JSON file of the job. The following figure shows the parameters for creating a main job.



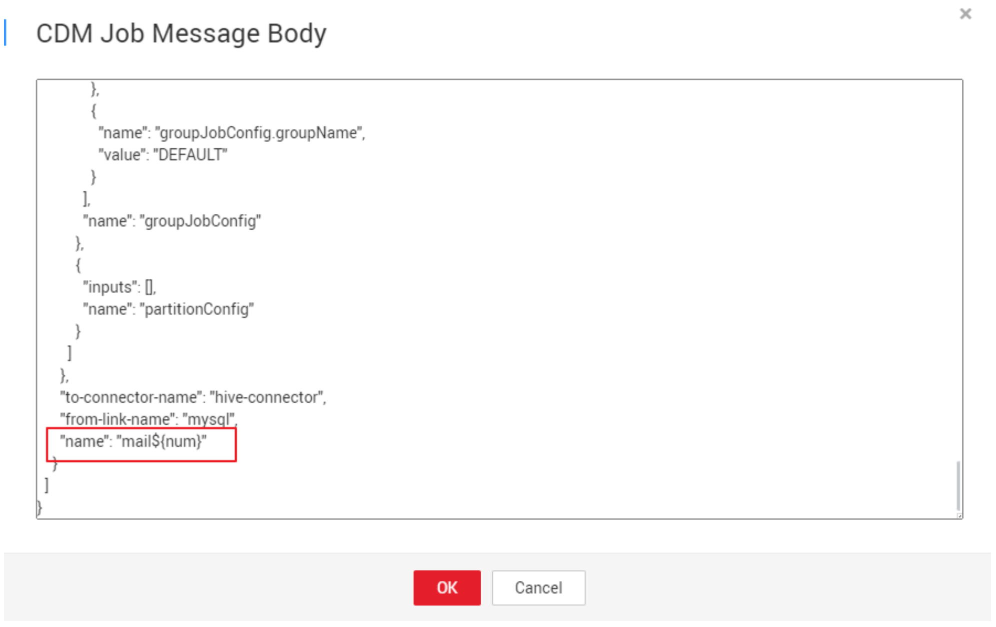
CDM Job Message Body

```
   {
    "name": "fromJobConfig.useSql",
    "value": "false"
   },
   {
    "name": "fromJobConfig.schemaName",
    "value": "internal"
   },
   {
    "name": "fromJobConfig.tableName",
    "value": "mail${num}"
   },
   {
    "name": "fromJobConfig.incrMigration",
    "value": "false"
   },
   {
    "name": "fromJobConfig.allowNullValueInPartitionColumn",
    "value": "false"
```

2. The name of each data migration job must be unique. Therefore, you need to change the value of **name** in the JSON file to **mail${num}** to create multiple CDM jobs. The following figure shows the parameters for creating a main job.

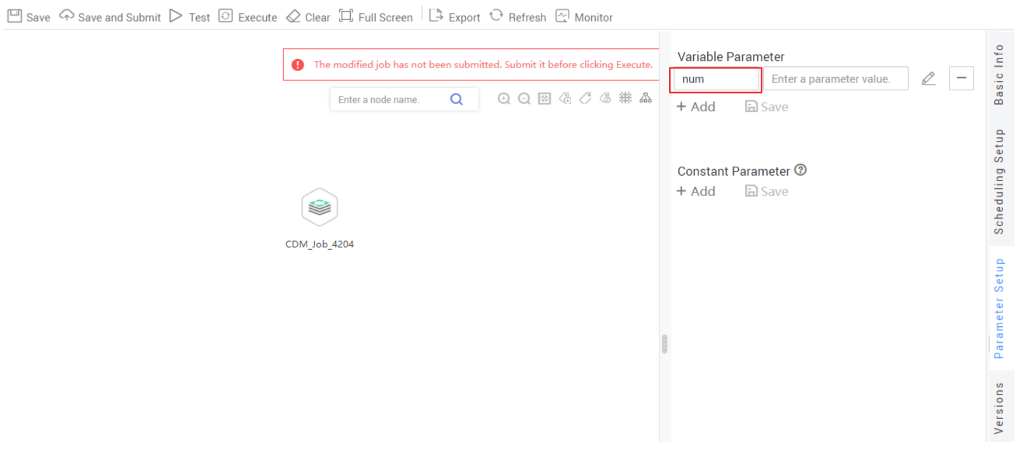
NOTE

To create a sharding job, you can change the source link in the job JSON file to a variable that can be easily replaced.



```
        },
        {
          "name": "groupJobConfig.groupName",
          "value": "DEFAULT"
        }
      ],
      "name": "groupJobConfig"
    },
    {
      "inputs": [],
      "name": "partitionConfig"
    }
  ]
},
"to-connector-name": "hive-connector",
"from-link-name": "mysql",
"name": "mail${num}"
}
]
}
```

Step 4 Add the **num** parameter, which is invoked in the job JSON file. The following figure shows the parameters for creating a main job.



Click **Save and Submit** to save the subjobs.

Step 5 Create the main job **integration_management**. Select the For Each node that executes the subjobs in a loop and transfers parameters **001**, **002**, and **003** to the subjobs to generate different table extraction tasks.

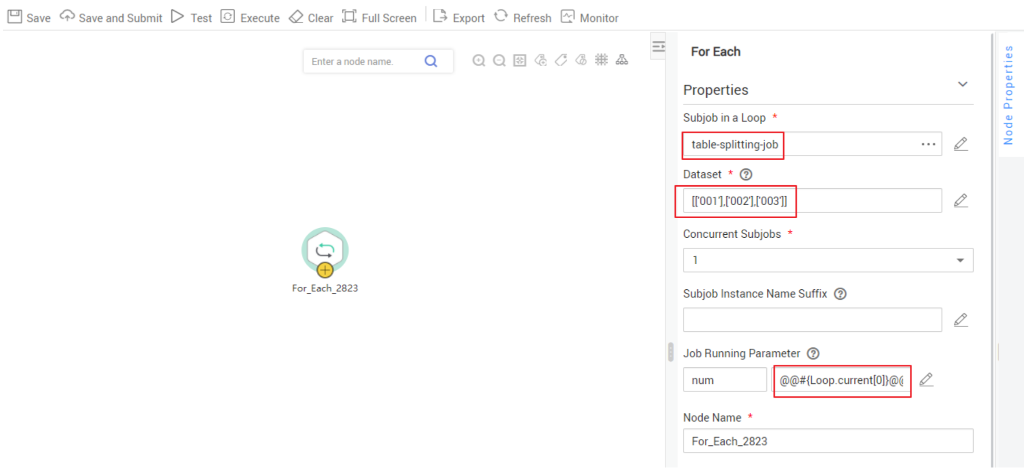
The key configurations are as follows:

- **Subjob in a Loop**: Select **table**.
- **Dataset**: Enter **[['001'],['002'],['003']]**.
- **Job Running Parameter**: Enter **@@#{Loop.current[0]}@@**.

NOTE

Add **@@** to the EL expression of the job running parameter. If **@@** is not added, dataset 001 will be identified as 1. As a result, the source table name does not exist.

The following figure shows the parameters for creating a main job.



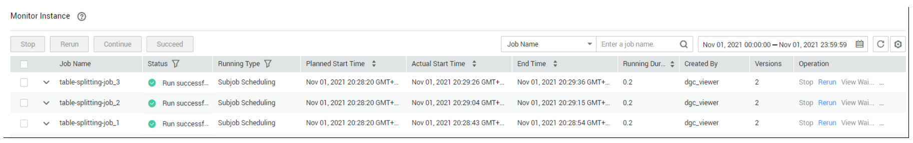
For Each

Properties

Subjob in a Loop *	table-splitting-job
Dataset *	[['001'],['002'],['003']]
Concurrent Subjobs *	1
Subjob Instance Name Suffix	
Job Running Parameter	num @@#{Loop.current[0]}@(
Node Name *	For_Each_2823

Click **Save and Submit** to save the main job.

Step 6 After the main job and subjobs are created, test and run the main job to check whether it is successfully created. If the job is successfully executed, the CDM subjobs are successfully created and executed.

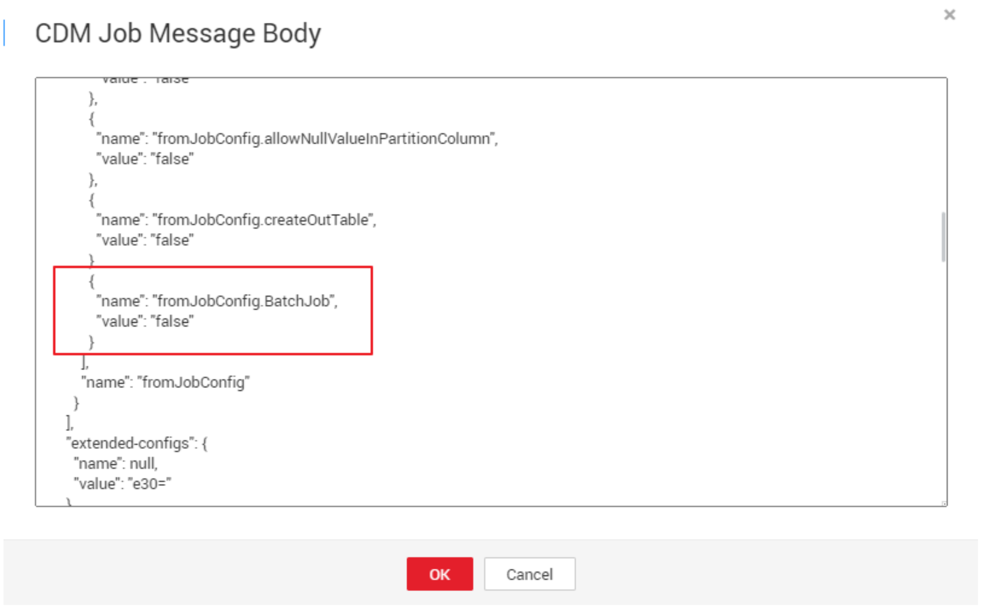


Job Name	Status	Running Type	Planned Start Time	Actual Start Time	End Time	Running Dur...	Created By	Versions	Operation
table-splitting-job_3	Run successf...	Subjob Scheduling	Nov 01, 2021 20:28:20 GMT+...	Nov 01, 2021 20:29:26 GMT+...	Nov 01, 2021 20:29:36 GMT+...	0.2	dgc_viewer	2	Stop Rerun View Wai... ...
table-splitting-job_2	Run successf...	Subjob Scheduling	Nov 01, 2021 20:28:20 GMT+...	Nov 01, 2021 20:29:04 GMT+...	Nov 01, 2021 20:29:15 GMT+...	0.2	dgc_viewer	2	Stop Rerun View Wai... ...
table-splitting-job_1	Run successf...	Subjob Scheduling	Nov 01, 2021 20:28:20 GMT+...	Nov 01, 2021 20:28:43 GMT+...	Nov 01, 2021 20:28:54 GMT+...	0.2	dgc_viewer	2	Stop Rerun View Wai... ...

----End

Important Notes

- Some attributes, such as **fromJobConfig.BatchJob**, may not be supported in some CDM versions. If an error is reported during task creation, you need to delete the attribute from the request body. The following figure shows the parameters for creating a main job.



```
      "value": "false"
    },
    {
      "name": "fromJobConfig.allowNullValueInPartitionColumn",
      "value": "false"
    },
    {
      "name": "fromJobConfig.createOutTable",
      "value": "false"
    }
    {
      "name": "fromJobConfig.BatchJob",
      "value": "false"
    }
  ],
  "name": "fromJobConfig"
 }
],
"extended-configs": {
  "name": null,
  "value": "e30="
```

- If a CDM node is configured to create a job, the node checks whether a CDM job with the same name is running.

- If the CDM job is not running, update the job with the same name based on the request body.
- If a CDM job with the same name is running, wait until the job is run. During this period, the CDM job may be started by other tasks. As a result, the extracted data may not be the same as expected (for example, the job configuration is not updated, or the macro of the running time is not correctly replaced). Therefore, do not start or create multiple jobs with the same name.

8 Practice: Data Development Based on E-commerce BI Reports

For details, see **Data Development Based on E-commerce BI Reports**.

9 Practice: Data Integration and Development Based on Movie Scores

For details, see **Data Integration and Development Based on Movie Scores**.

10 Practice: Data Governance Based on Taxi Trip Data

For details, see **Governance of Taxi Trip Data**.

11 Case: Trade Data Statistics and Analysis

11.1 Scenario

Consulting company H uses CDM to import local trade statistics to OBS, and Data Lake Insight (DLI) to analyze trade statistics. In this way, company H builds its big data analytics platform at an extremely low cost, allowing the company more time to focus on their businesses and make innovations continuously.

Background

Company H is a commercial organization in China that engages in collecting trade statistics of major trading nations and buyer data. It has a large-scale trade statistics database. The collected data is widely used in industry research, international trade promotion, and other fields.

In the past, company H used its own big data cluster with maintenance by dedicated personnel. Each year, company H purchased the dedicated bandwidth from China Telecom and China Unicom and invested heavily in equipment room, electric power, private networks, servers, and O&M. However, the company could not satisfy customers' ever-changing service requirements due to insufficient manpower and capability restrictions of its big data cluster. As a result, only 4% of 100 TB inventory data was useful.

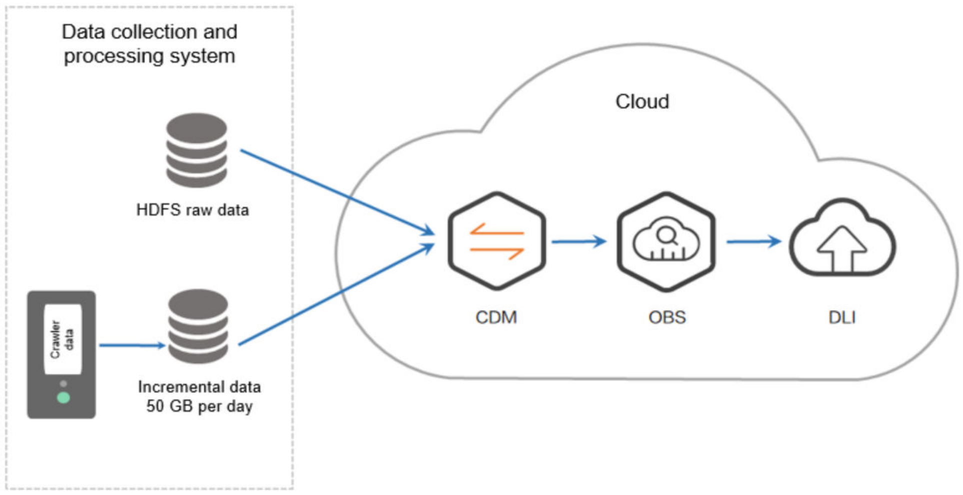
After migrating local trade statistics to HUAWEI CLOUD, company H can make full use of the 100 TB inventory data in maximizing asset monetization, without the need of constructing and maintaining infrastructures but relying on HUAWEI CLOUD's big data analysis capabilities.

CDM and DLI use the pay-per-use billing mode, so maintenance personnel are not required and the dedicated bandwidth cost is reduced. Compared with the offline data center, CDM and DLI save the maintenance cost by 70%. In addition, CDM and DLI have low skill demands for personnel and enable smooth migration of existing services, shortening the service rollout period by 50%.

Task

Use CDM, OBS, and DLI to complete trade statistics analysis using the existing data (for example, trade detail records and basic information) of company H's customer data collection and processing system.

Figure 11-1 Scenario scheme


Data collection and processing system
HDFS raw data
Crawler data
Incremental data 50 GB per day
Cloud
CDM
OBS
DLI

NOTE

When creating an OBS foreign table on DLI, the data storage format of the OBS table must meet the following requirements:

- When you use the DataSource syntax to create an OBS table, the ORC, Parquet, JSON, CSV, Carbon, and Avro formats are supported.
- When you use the Hive syntax to create an OBS table, the Text file, Avro, ORC, SequenceFile, RCFile, Parquet, Carbon formats are supported.

If the storage format of the raw data table does not meet the requirements, you can use CDM to import the raw data to DLI for analysis without uploading the data to OBS.

Data Types

- Trade detail records

  Trade detail records include trade statistics of major trading nations.

Table 11-1 Trade detail records

Field Name	Field Type	Field Description
hs_code	string	List of import and export offering code
country	smallint	Basic information about countries
dollar_value	double	Transaction amount
quantity	double	Transaction volume

Field Name	Field Type	Field Description
unit	smallint	Measurement unit
b_country	smallint	Basic information about the target country
imex	smallint	Import or export
y_year	smallint	Year
m_month	smallint	Month

- Basic information

  The basic information indicates the dictionary data corresponding to the fields in the trade detail records.

Table 11-2 Basic information about countries (description of **country**)

Field Name	Field Type	Field Description
countryid	smallint	Country code
country_en	string	English name of a country
country_cn	string	Chinese name of a country

Table 11-3 Information about the update time (description of **updatetime**)

Field Name	Field Type	Field Description
countryid	smallint	Country code
imex	smallint	Import or export
hs_len	smallint	Length of the offering code
minstartdate	string	Minimum start time
startdate	string	Start time
newdate	string	Update time
minnewdate	string	Last update time

Table 11-4 Information about import and export offering code (description of **hs246**)

Field Name	Field Type	Field Description
id	bigint	ID
hs	string	Offering code
hs_cn	string	Chinese name of an offering
hs_en	string	English name of an offering

Table 11-5 Information about units (description of **unit_general**)

Field Name	Field Type	Field Description
id	smallint	Measurement unit code
unit_en	string	English name of a measurement unit
unit_cn	string	Chinese name of a measurement unit

11.2 Analysis Process

Introduction

To use CDM, OBS, and DLI to analyze trade statistics, you need to perform the following steps:

1. **Using CDM to Upload Data to OBS**
   a. Use CDM to upload the inventory data of company H to OBS.
   b. Configure a scheduled task of CDM to automatically upload incremental data to OBS every day.
2. **Using DLI to Analyze Data**
   Use DLI to directly analyze the service data in OBS to support the customers of company H for trade statistics analysis.

11.3 Using CDM to Upload Data to OBS

11.3.1 Uploading Inventory Data

1. Use **Direct Connect** to establish a Direct Connect connection between the local data center and Huawei Cloud Virtual Private Cloud (VPC).

2. Create an OBS bucket and record the access domain name, port number, access key ID (AK), and secret access key (SK) of the OBS bucket.
3. Create a CDM cluster.

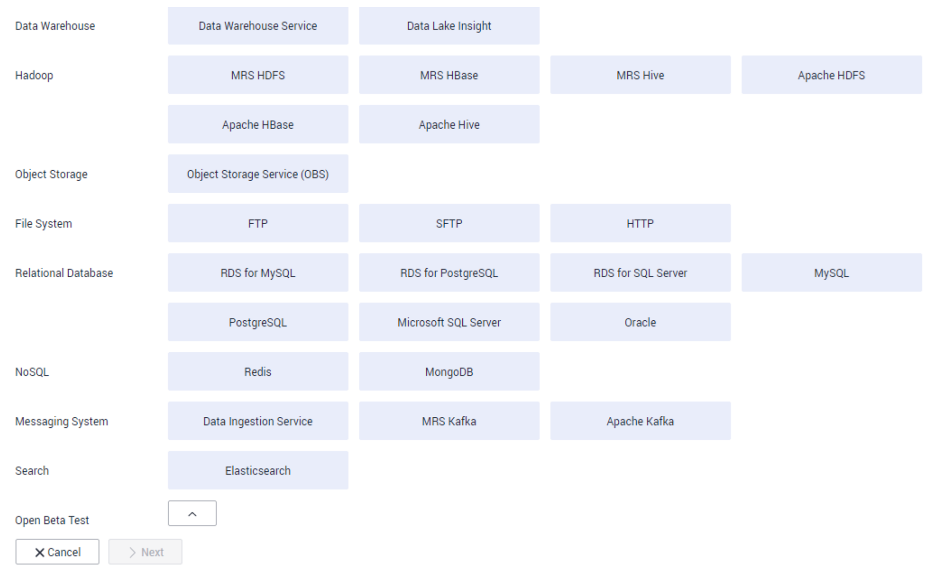
   NOTE

   If a DataArts Studio instance includes a CDM cluster (except the trial version) and the cluster meets your requirements, you do not need to buy a DataArts Migration incremental package.

   If you need to create another CDM cluster, buy a DataArts Studio incremental package by referring to **Buying a DataArts Studio Incremental Package**.

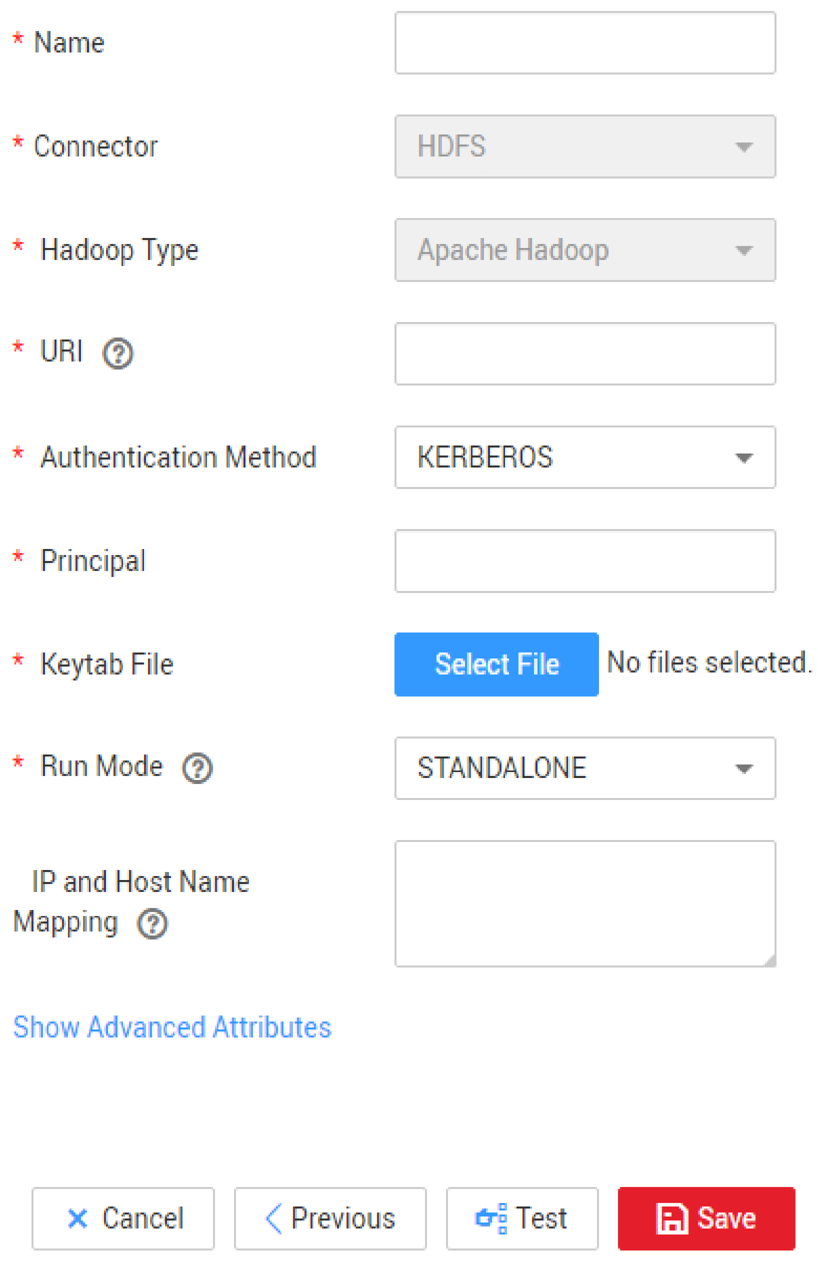
   – **Instance Type**: Select **cdm.xlarge**, which applies to most migration scenarios.
   – **VPC**: VPC of the CDM cluster. Select the VPC that connects to the local data center through Direct Connect.
   – (Optional) **Subnet** and **Security Group**: You can configure either of them.
4. After the cluster is created, choose **Job Management** > **Link Management** > **Create Link**. The page for selecting a link type is displayed. See **Figure 11-2**.

Figure 11-2 Selecting a connector



5. To connect to the local Apache HDFS of company *H*, select **Apache HDFS**, and click **Next**.

Figure 11-3 Creating an HDFS link



* Name	
* Connector	HDFS
* Hadoop Type	Apache Hadoop
* URI	
* Authentication Method	KERBEROS
* Principal	
* Keytab File	Select File No files selected.
* Run Mode	STANDALONE
IP and Host Name Mapping	

NOTE

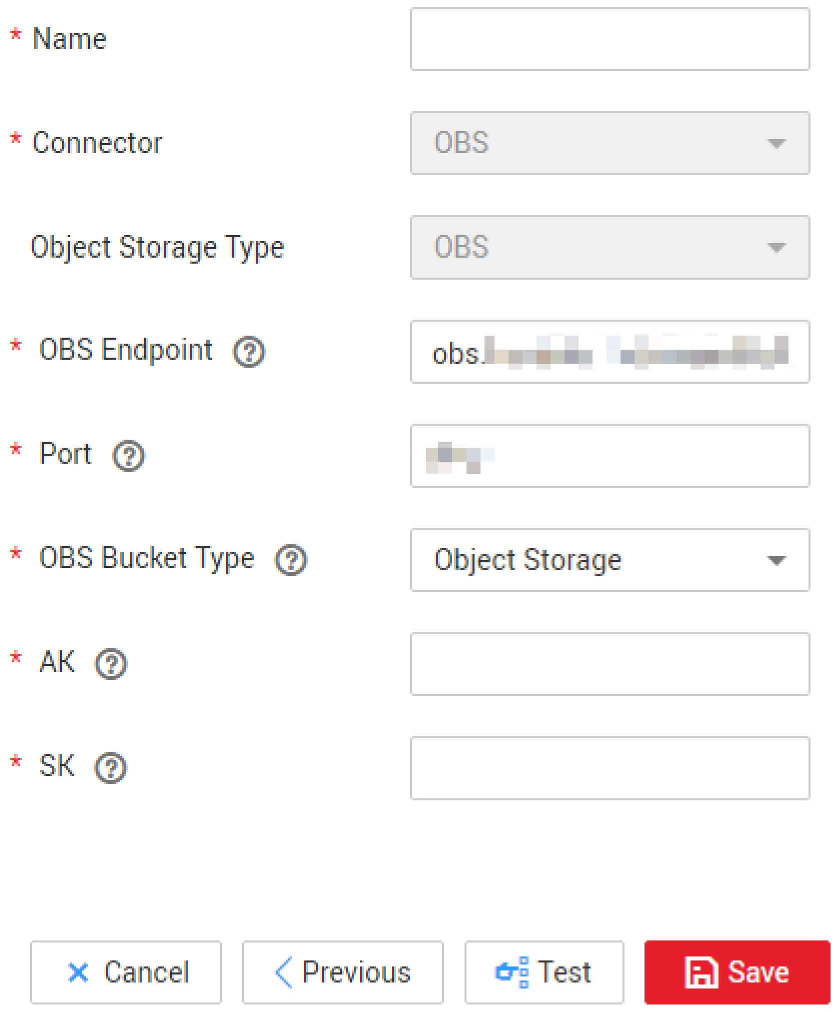
- **Name**: Enter a custom link name, for example, **hdfs_link**.
- **URI**: Enter the NameNode URI of HDFS of company *H*.
- **Authentication Method**: Select **KERBEROS** if Hadoop is in security mode to obtain the **principal** and **keytab** files from the client for authentication.
- **Principal** and **Keytab File**: Obtain the **principal** account and **keytab** file from the Hadoop administrator.

6. Click **Save**. CDM automatically checks whether the link is available.
   – If the link is available, a message is displayed, indicating that the link is successfully saved, and the link management page is displayed.

   – If the link is unavailable, check whether the link parameters are correctly configured or whether the firewall of company *H* allows the elastic IP address (EIP) of the CDM cluster to access the data source.

7. Click **Create Link** to create an OBS link. On the page that is displayed, select **Object Storage Service (OBS)** and click **Next**. Set the OBS link parameters as required. See **Figure 11-4**.

Figure 11-4 Creating an OBS link



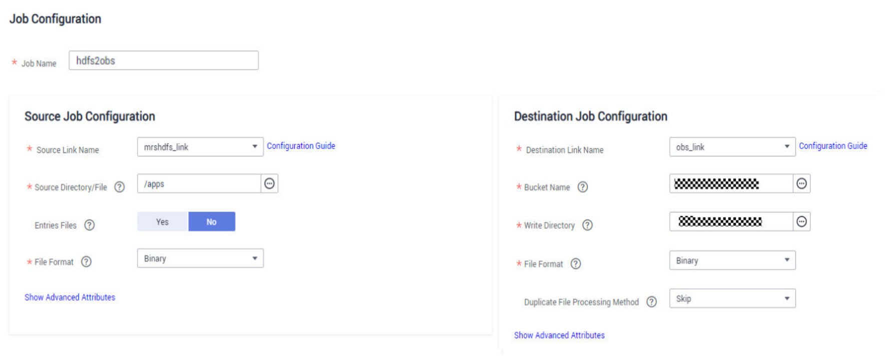
* Name	
* Connector	OBS
Object Storage Type	OBS
* OBS Endpoint	obs.
* Port	
* OBS Bucket Type	Object Storage
* AK	
* SK	

NOTE

- **Name**: Enter a custom link name, for example, **obslink**.
- **OBS Endpoint**: Enter the domain name or IP address of OBS, for example, **obs.myhuaweicloud.com**.
- **Port**: Enter the port number of OBS, for example, **443**.
- **OBS Bucket Type**: Select a value from the drop-down list box as required.
- **AK** and **SK**: Enter the AK and SK used for accessing the OBS database. To obtain the AK and SK, log in to the management console, click the username in the upper right corner, and select **My Credentials** from the drop-down list. On the displayed page, choose **Access Keys** in the left navigation pane.

8. Click **Save**. The **Link Management** page is displayed.
9. Choose **Table/File Migration** > **Create Job** to create a job for migrating trade statistics of company *H* to OBS. See **Figure 11-5**.

Figure 11-5 Creating a job



Job Configuration

* Job Name	hdfs2obs

Source Job Configuration

* Source Link Name	mrshdfs_link
* Source Directory/File	/apps
Entries Files	Yes [✓ No]
* File Format	Binary

Destination Job Configuration

* Destination Link Name	obs_link
* Bucket Name	[redacted]
* Write Directory	[redacted]
* File Format	Binary
Duplicate File Processing Method	Skip

NOTE

- **Job Name**: Enter a user-defined job name.
- **Source Link Configuration**:
  - **Source Link Name**: Select the HDFS link **hdfs_link** created in **5**.
  - **Source Directory/File**: Set this parameter to the local storage path of company *H*'s trade statistics. The value can be either a directory or a file. Set this parameter to a directory. CDM migrates all files in the directory to OBS.
  - **File Format**: Select **Binary**. The file format refers to the format used by CDM to transmit data. The formats of the original files are not changed. **Binary** is recommended for migration between files because the transmission efficiency and performance are optimal.
- **Destination Link Configuration**:
  - **Destination Link Name**: Select the OBS link **obslink** created in **7**.
  - **Bucket Name** and **Write Directory**: Enter the path for storing trade statistics in OBS. CDM writes the files to this path.
  - **File Format**: Select **Binary**. Similar to the source link, the formats of the original files are not changed.
  - **Duplicate File Processing Method**: Select **Skip**. CDM determines that a file is a duplicate file only when the file name and file size are the same on the source and destination ends. In this case, CDM skips the file and does not migrate the file to OBS.

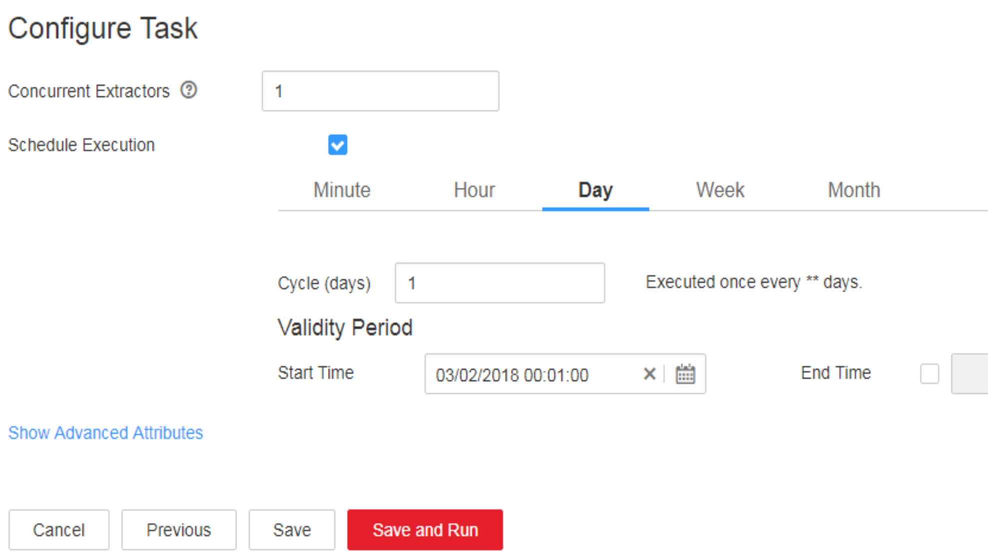
10. Click **Next** to go to the tab page for configuring the task parameters. For the migration of inventory data, retain the default values of the parameters.
11. Click **Save and Run**. On the displayed job management page, you can view the job execution progress and result.
12. After the job is successfully executed, click **Historical Record** to view the number of written rows, number of read rows, number of written bytes, number of written files, and execution logs.

11.3.2 Uploading Incremental Data

1. After uploading inventory data using CDM, click **Edit** in the **Operation** column to modify a job.

2. Retain the values of the basic parameters, and click **Next** to modify the task parameters. See **Figure 11-6**.

Figure 11-6 Configuring a scheduled task


Configure Task

Concurrent Extractors: 1

Schedule Execution: ✓

Minute | Hour | **Day** | Week | Month

Cycle (days): 1 Executed once every ** days.

Validity Period

Start Time: 03/02/2018 00:01:00 End Time: ☐

3. Select **Schedule Execution** and configure the scheduled task.
   – Set **Cycle (days)** to 1 day.
   – Set **Start Time** to 00:01:00 every day.

   In this way, CDM automatically performs full migration in the early morning every day. However, because **Duplicate File Processing Method** is set to **Skip**, files with the same name and size are not migrated. Therefore, only new files are uploaded every day.
4. Click **Save**.

11.4 Analyzing Data

Use DLI to analyze the trade statistics stored in OBS buckets.

Prerequisites

When creating an OBS foreign table on DLI, the data storage format of the OBS table must meet the following requirements:

- When you use the DataSource syntax to create an OBS table, the ORC, Parquet, JSON, CSV, Carbon, and Avro formats are supported.
- When you use the Hive syntax to create an OBS table, the Text file, Avro, ORC, SequenceFile, RCFile, Parquet, Carbon formats are supported.

If the storage format of the raw data table does not meet the requirements, you can use CDM to import the raw data to DLI for analysis without uploading the data to OBS.

Procedure

1. Log in to the DLI console and create a database by referring to **Creating a Database**.
2. Create an OBS foreign table by referring to **Creating an OBS Table**, including the trade statistics database, trade detail record table, and basic information table.
3. Develop SQL scripts on the DLI console for trade statistics analysis to meet service requirements.