DataArts Studio

Best Practices

Issue	01
Date	2024-04-29

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


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

Notice

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

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

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: https://www.huaweicloud.com/intl/en-us/

Contents

1 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 at or 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

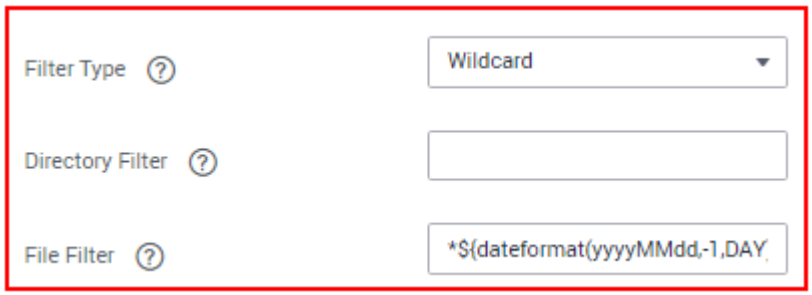
- Parameter position: When creating a table/file migration job, if the migration source is a file system, set **Filter Type** in advanced attributes of **Source Job Configuration** to **Wildcard** or **Regular expression**.
- Parameter principle: If you select **Wildcard** for **Filter Type**, CDM filters 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

  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: After you specify the start time and end time, only files that are modified between the start time (included) and end time (excluded) will be migrated.
- 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

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:

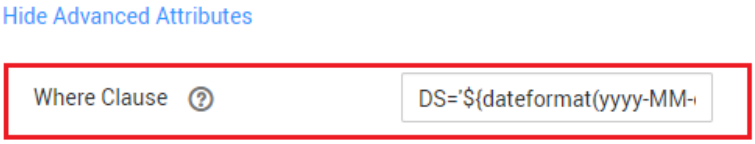
  Figure 1-3 Table data

  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

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

  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

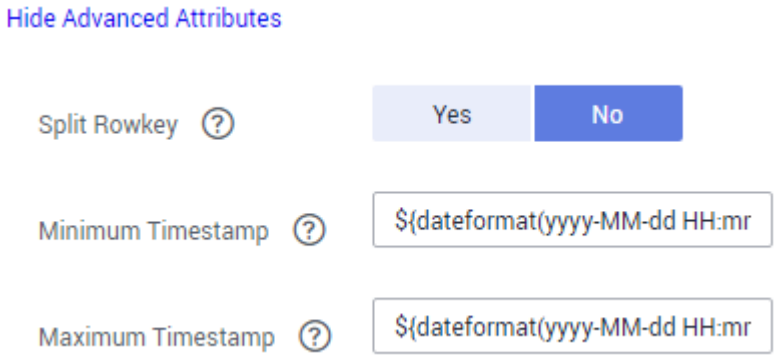
You can use CDM to export data in a specified period of time from HBase (including MRS HBase, FusionInsight HBase, and Apache HBase) and CloudTable. The CDM scheduled jobs can be used together to implement incremental migration of HBase and CloudTable.

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



- 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.

1.1.4 MongoDB/DDS Incremental Migration

By using CDM, you can export MongoDB or DDS data within a specified period. With the scheduled jobs of CDM, you can implement incremental migration of MongoDB and DDS.

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 MongoDB or DDS as the source link, you can set the query filters in advanced attributes.

Figure 1-6 Setting query filters

You can set this parameter to a **macro variable of date and time**, for example, **{"ts":{$gte:ISODate("${dateformat(yyyy-MM-dd'T'HH:mm:ss.SSS'Z',-1,DAY)}")}}**, which indicates searching for the values in the **ts** field that are greater than those after time macro conversion, that is, only the data generated after the previous day is exported.

After this parameter is set, CDM exports only the data generated on the previous day. In addition, you can set the job to be executed at 00:00:00 every day, so that 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, MONTH, YEAR, and DAY.

    NOTE

    Pay attention to the following special scenarios of **MONTH** and **YEAR**:

    - If the date does not exist after the offset, the latest date of the month in the calendar is used.
    - These two offset types cannot be used for the start time and end time in the **Time Filter** parameter of the source and destination jobs.

  For example, if the current date is **2023-03-01 09:00:00**, then:

  - **dateformat(yyyy-MM-dd HH:mm:ss, -1, YEAR)** indicates the year before the current time, that is, **2022-03-01 09:00:00**.
  - **dateformat(yyyy-MM-dd HH:mm:ss, -3, MONTH)** indicates three months before the current time, that is, **2022-12-01 09:00:00**.
  - **dateformat(yyyy-MM-dd HH:mm:ss, -1, DAY)** indicates the day before the current time, that is, **2023-02-28 09:00:00**.
  - **dateformat(yyyy-MM-dd HH:mm:ss, -1, HOUR)** indicates one hour before the current time, that is, **2023-03-01 08:00:00**.
  - **dateformat(yyyy-MM-dd HH:mm:ss, -1, MINUTE)** indicates one minute before the current time, that is, **2023-03-01 08:59:00**.
  - **dateformat(yyyy-MM-dd HH:mm:ss, -1, SECOND)** indicates one second before the current time, that is, **2023-03-01 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
${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

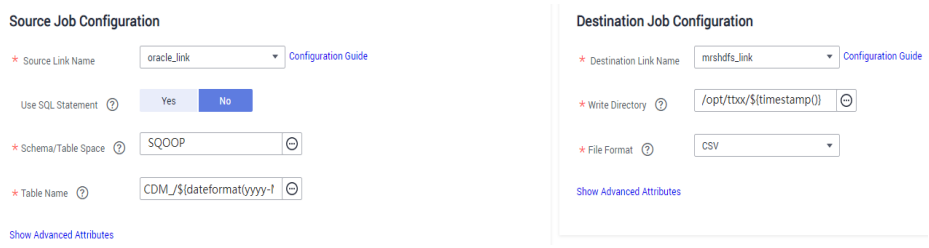
Time and Date Macro Variables of Paths and Table Names

Figure 1-7 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 1-7 Setting **Table Name** and **Write Directory** to a time and date macro variable



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 1-8 uses table **SQOOP.CDM_20171016** as an example. The table contains column **DS**, which indicates the time.

Figure 1-8 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.

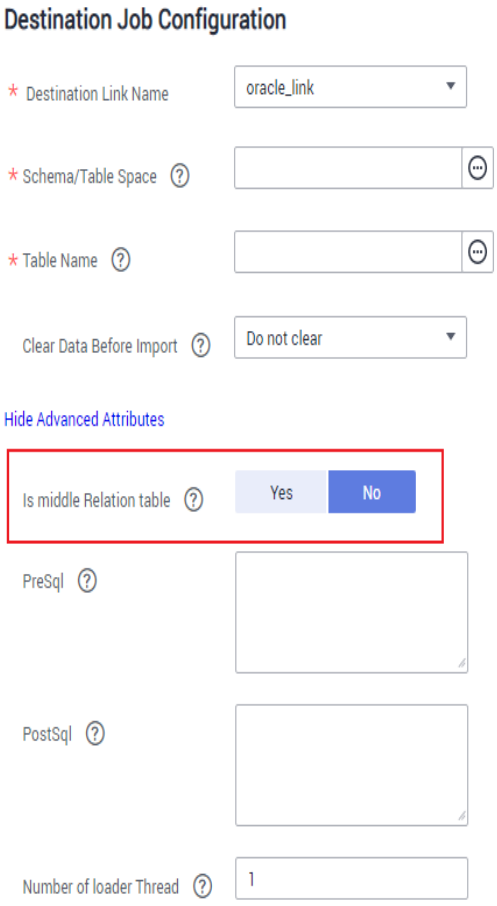
Configuration principles of other application scenarios are the same.

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 1-9 Migration in transaction mode



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: HDFS (supported in the binary format)
- Data sources supported by the migration destination: 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 HDFS and encrypt the files to be imported to HDFS.

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

  When you use CDM to create a job for exporting files from HDFS, set the migration source to HDFS and file format to binary, 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 HDFS, the files written to the migration destination are decrypted plaintext files.

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

  When you create a CDM job to import files to HDFS, set the migration destination to HDFS and file format to binary, 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 HDFS, the files in the destination HDFS 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.

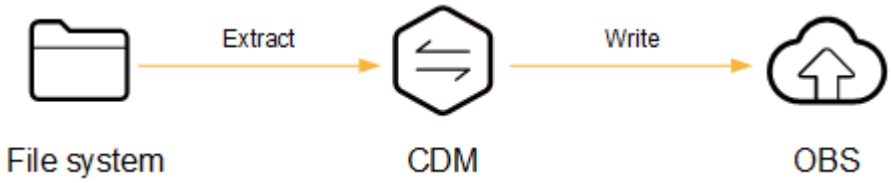
NOTE

- 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 1-10** shows the migration mode when files are migrated to OBS.

Figure 1-10 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.

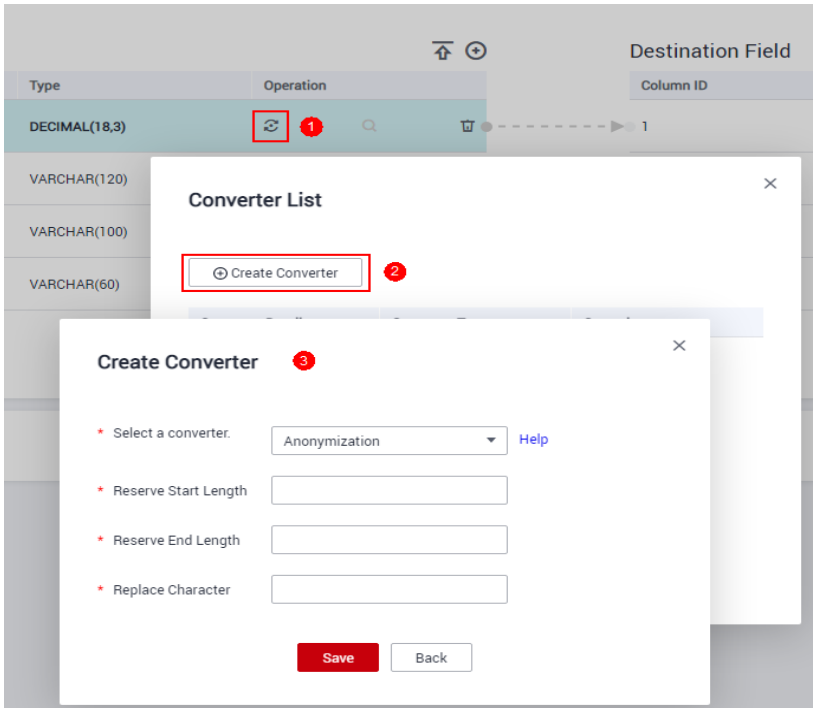
1.6 Configuring Field Converters

Scenario

- After the job parameters are configured, field mapping needs to be configured. You can click in the **Operation** column to create a field converter.
- If files are migrated between FTP, SFTP, OBS, and HDFS and the migration source's **File Format** is set to **Binary**, files will be directly transferred, free from field mapping.

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

Figure 1-11 Creating a field converter



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

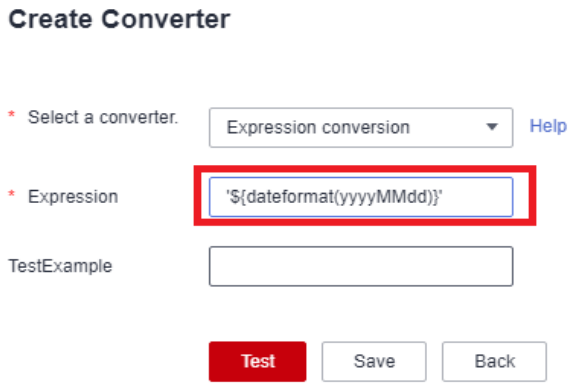
- **Anonymization**
- **Trim**
- **Reverse String**

- **Replace String**
- **Remove line break**
- **Expression Conversion**

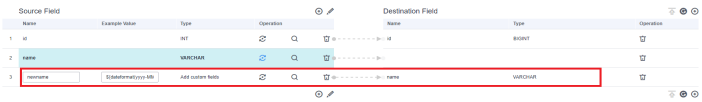
Constraints

- If **Use SQL Statement** is set to **Yes** in the source job configuration, converters cannot be created.
- On the **Map Field** tab page, if CDM fails to obtain all columns by obtaining sample values (for example, when data is exported from HBase, CloudTable, or MongoDB, there is a high probability that CDM failed to obtain all columns), you can click  and select **Add a new field** to add new fields to ensure that the data imported to the migration destination is complete.
- When a relational database, Hive, DLI, or MRS Hudi is used as the migration source, sample values cannot be obtained.
- When SQLServer is the destination, fields of the timestamp type cannot be written. You must change their type (for example, to datetime) so that they can be written.
- Column names are displayed when the source of the migration job is OBS, CSV files are to be migrated, and parameter **Extract first row as columns** is set to **Yes**.
- Field converters configuration is not involved when the binary format is used to migrate files to files.
- In the automatic table creation scenario, you need to manually add fields to the destination table in advance and then add fields to the field mapping.
- 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.
- If the field mapping is incorrect, you can adjust the field mapping by dragging fields or clicking  to map fields in batches.
- An expression processes the data of a field. When creating an expression converter, you are not advised to use a time macro. If you need to use a time macro, use either of the following methods (if the source is of the file type, only **Method 1** is supported):
  - Method 1: When creating an expression converter, use two single quotation marks ('') to enclose the expression.

    For example, if expression **${dateformat(yyyy-MM-dd)}** is not enclosed in quotation marks, the hyphen (-) in the value **2017-10-16** parsed from the expression will be recognized as a minus sign, and further calculation will be performed to generate result **1991**, which is incorrect. If you enclose the expression in quotation marks, that is, **'${dateformat(yyyy-MM-dd)}'**, you will obtain **'2017-10-16'**, which is correct.



  - Method 2: Add a custom source field, enter a macro variable of date and time for **Example Value**, and map the field to a destination field again.



- If the data is imported to GaussDB(DWS), you need to select the distribution columns in the destination fields. You are advised to select the distribution columns according to the following rules:
  a. Use the primary key as the distribution column.
  b. If multiple data segments are combined as primary keys, specify all primary keys as the distribution column.
  c. In the scenario where no primary key is available, if no distribution column is selected, DWS uses the first column as the distribution column by default. As a result, data skew risks exist.

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 *****.

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. Convert a timestamp to a date string in *yyyy-MM-dd hh:mm:ss* format, for example, convert **1701312046588** to **2023-11-30 10:40:46**.

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

  e. Convert a date string in the yyyy-MM-dd hh:mm:ss format to a timestamp.

     Expression: DateUtils.getTime(DateUtils.parseDate(value,"yyyy-MM-dd hh:mm:ss"))

  f. 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,"-")

  g. 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

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

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

  i. 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

j. 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")

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

Expression: CommonUtils.randomUUID()

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

Expression: StringUtils.capitalize(value)

m. 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)

n. 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*)

o. 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)

p. 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*")

q. 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*")

r. 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*")

s. 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*")

t. 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*)

u. 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*)

v. 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*")

w. 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*")

x. 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*")

y. 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*)

z. 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*")

aa. 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)

ab. 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)

ac. 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)

ad. 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)

ae. 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)

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

   Expression: StringUtils.isEmpty(value)

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

   Expression: StringUtils.isNumeric(value)

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

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

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

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

aj. 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*")

ak. 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*")

al. 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)

am. 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*")

an. 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*")

ao. 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.*")

ap. 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*")

aq. 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*")

ar. 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*")

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

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

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

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

au. 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*)

av. 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*)

aw. 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)

ax. 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*")

ay. 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*")

az. 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*")

ba. 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*")

bb. 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*")

bc. 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)

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

Expression: NumberUtils.toByte(value)

be. 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*)

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

Expression: NumberUtils.toDouble(value)

bg. 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*)

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

Expression: NumberUtils.toFloat(value)

bi. 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*)

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

Expression: NumberUtils.toInt(value)

bk. 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*)

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

Expression: NumberUtils.toLong(value)

bm. 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*)

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

Expression: NumberUtils.toShort(value)

bo. 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*)

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

Expression: CommonUtils.ipToLong(value)

bq. 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*")

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

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

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

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

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

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

bu. 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*)

bv. If the value is empty or null, "aaa" is returned. Otherwise, **value** is returned.

Expression: StringUtils.defaultIfEmpty(value, *"aaa"*)

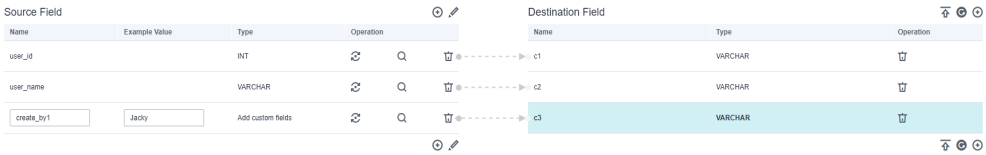
1.7 Adding Fields

Scenario

- After job parameters are configured, field mapping needs to be configured. You can customize new fields by clicking on the **Map Field** page.
- If files are migrated between FTP, SFTP, OBS, and HDFS and the migration source's **File Format** is set to **Binary**, files will be directly transferred, free from field mapping.
- In other scenarios, CDM automatically maps fields of the source table and the destination table. You need to check whether the mapping and time format are correct. For example, check whether the source field type can be converted into the destination field type.

You can click on the **Map Field** page and select **Add** to customize a new field. This field is usually used to mark the database source to ensure the integrity of the data imported to the migration destination.

Figure 1-12 Field mapping



Currently, the following field types are supported:

- **Constant Parameter**

  Constant parameters are fixed parameters and do not need to be reconfigured. For example, **lable** = **friends** is used to identify a constant value.

- **Variables**

  You can use variables such as time macros, table name macros, and version macros to mark database source information. The variable syntax is ${variable}, where **variable** indicates a variable. For example, **input_time** = **${timestamp()}** indicates the timestamp of the current time.

- Expression

  You can use the expression language to dynamically generate parameter values based on the running environment. The expression syntax is #{expr}, where **expr** indicates an expression. For example, **time** = **#{DateUtil.now()}** is used to identify the current date string.

Constraints

- On the **Map Field** tab page, if CDM fails to obtain all columns by obtaining sample values (for example, when data is exported from HBase, CloudTable, or MongoDB, there is a high probability that CDM failed to obtain all columns), you can click  and select **Add a new field** to add new fields to ensure that the data imported to the migration destination is complete.
- When a relational database, Hive, DLI, or MRS Hudi is used as the migration source, sample values cannot be obtained.
- When SQLServer is the destination, fields of the timestamp type cannot be written. You must change their type (for example, to datetime) so that they can be written.
- Column names are displayed when the source of the migration job is OBS, CSV files are to be migrated, and parameter **Extract first row as columns** is set to **Yes**.
- Field mapping is not involved when the binary format is used to migrate files to files.
- In the automatic table creation scenario, you need to manually add fields to the destination table in advance and then add fields to the field mapping.
- 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.
- If the field mapping is incorrect, you can adjust the field mapping by dragging fields or clicking  to map fields in batches.
- If the data is imported to DWS, you need to select the distribution columns in the destination fields. You are advised to select the distribution columns according to the following principles:
  a. Use the primary key as the distribution column.
  b. If multiple data segments are combined as primary keys, specify all primary keys as the distribution column.
  c. In the scenario where no primary key is available, if no distribution column is selected, DWS uses the first column as the distribution column by default. As a result, data skew risks exist.

- If a source field type is not supported, convert the field type to a type supported by CDM by referring to **Converting Unsupported Data Types**.

1.8 Migrating Files with Specified Names

You can migrate files (a maximum of 50) with specified names from FTP, OBS, or SFTP 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, OBS, or SFTP, **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.

1.9 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**.

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
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

1.10 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.

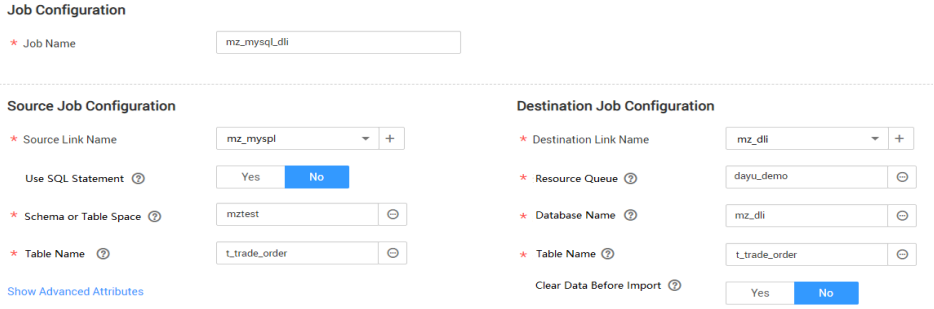
Prerequisites

- A link has been created, and the source end of the connector is a relational database.
- The destination data table contains a date and time field or timestamp field. In the automatic table creation scenario, you need to manually create the date and time field or timestamp field in the destination table in advance.

Creating a Table/File Migration Job

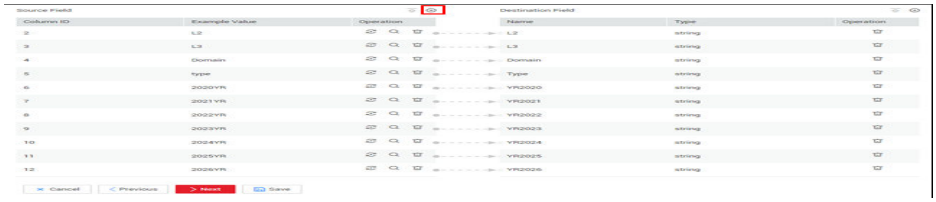
Step 1 Create a table/file migration job, and select the created source connector and destination connector.

Figure 1-13 Configuring the job



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

Figure 1-14 Configuring field mapping



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

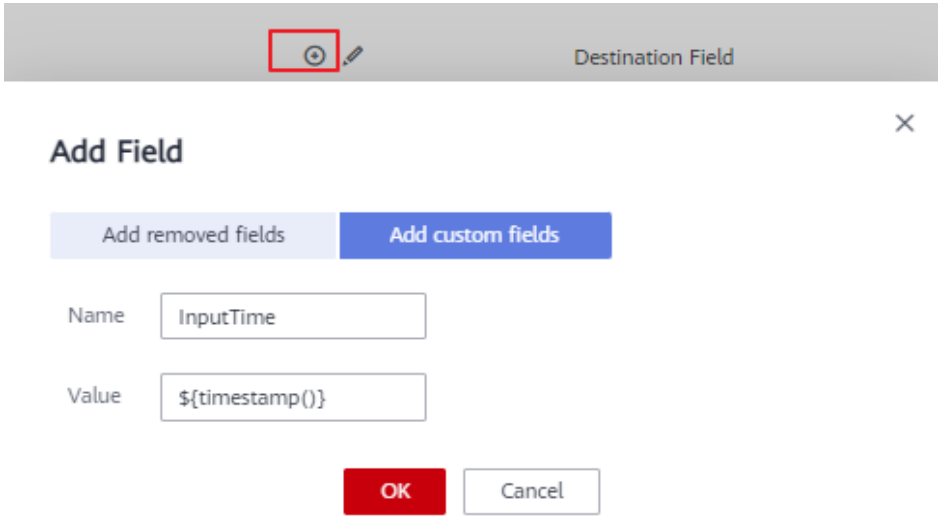


Table 1-7 Macro variable definition of time and date

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

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

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.
- After adding the fields, ensure that the customized import time field matches the field type of the destination table.

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

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 logs.

Step 7 Go to the destination data source to check the time when the data is imported to the database.

----End

1.11 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**

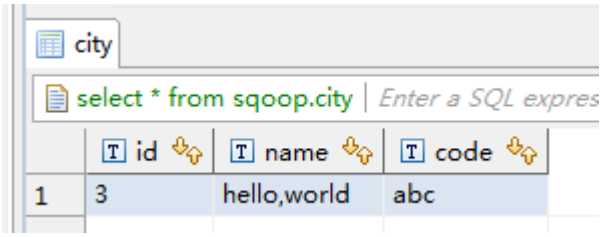
   Encoding type of a CSV file. The default value is **UTF-8**. Some Chinese characters are encoded by GBK.

   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

     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"]
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:

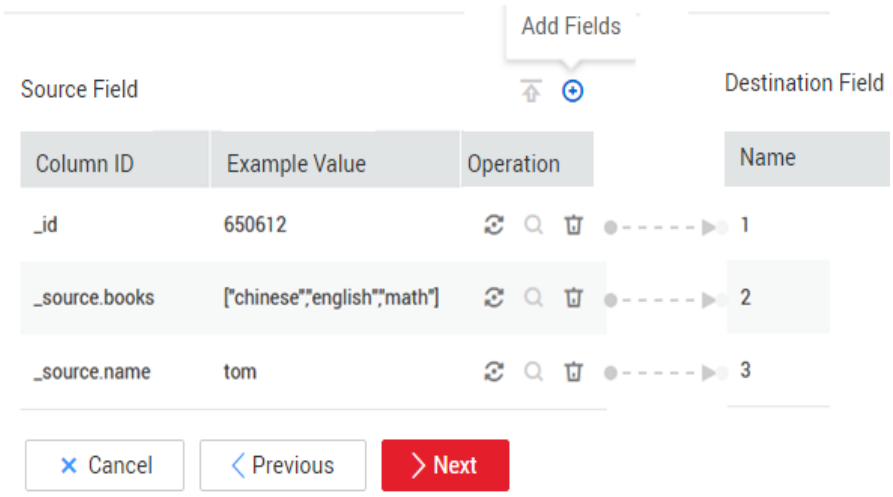
ID	SourceName	SourceBooks	MaxScore
650612	tom	["book1","book2","book3"]	1.0

ID	SourceName	SourceBooks	MaxScore
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.

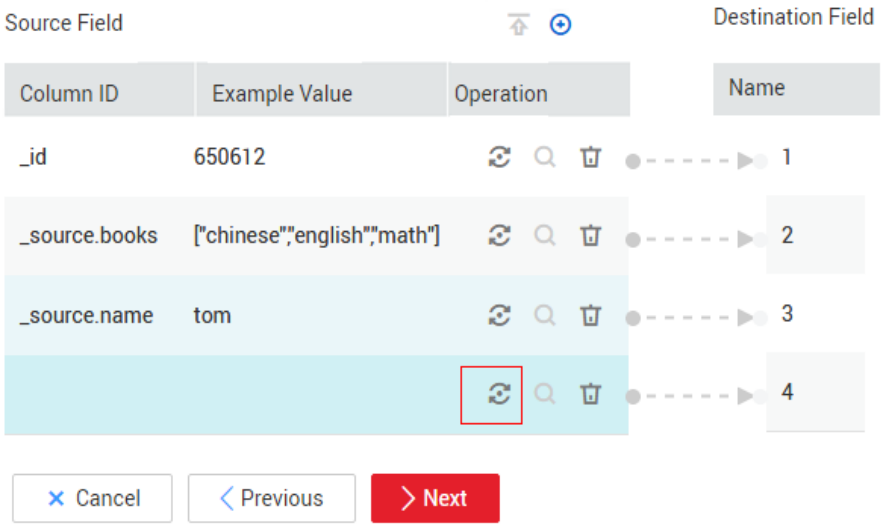
i. Click ⊕ to add a field.

Figure 1-17 Adding a field



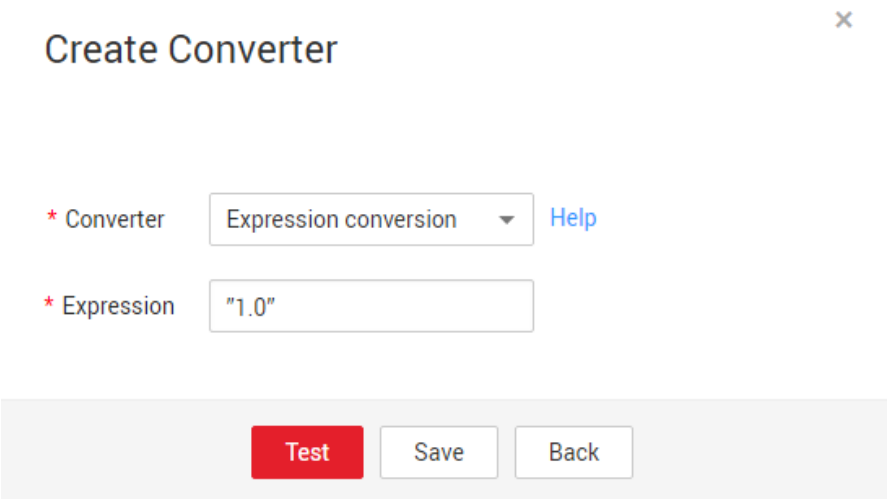
ii. Click ⟳ to create a converter for the new field.

Figure 1-18 Creating a field converter



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

Figure 1-19 Configuring a field converter



Binary

If you want to copy files between file systems, you can select the binary format. Files can be transferred in binary format at a high speed and stable performance. In addition, field mapping is not required in the second step of the job.

- **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

- **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.

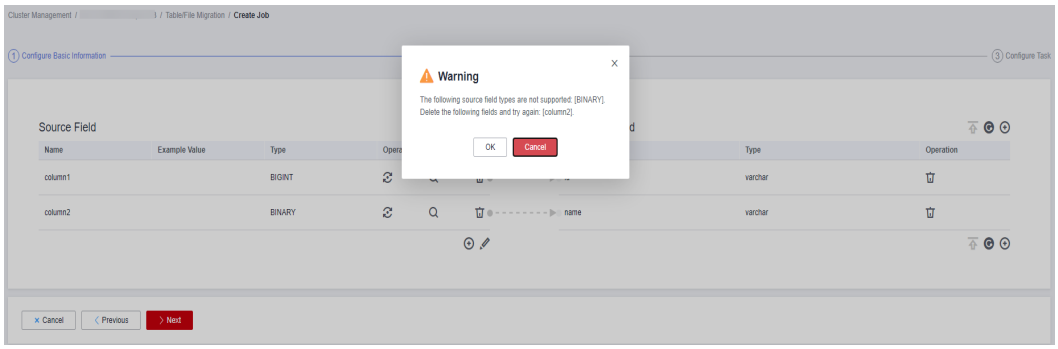
   Solution: Specify a line separator.

   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.

1.12 Converting Unsupported Data Types

Scenario

When field mapping is configured on CDM, a message is displayed indicating that the data type of the field is not supported and the field needs to be deleted. If you need to use this field, you can use SQL statements to convert the field type in the source job configuration to the type supported by CDM for data migration.



Procedure

Step 1 Modify the CDM migration job and enable **Use SQL Statement**.

NOTE

The SQL statement format is as follows: **select id,cast**(*Original field name* **as INT) as** *New field name, which can be the same as the original field name* **from schemaName.tableName;**

For example, select `id`, `name`, cast(`sex` AS char(255)) AS `sex` from `test_1117869`.`test_no_support_type`;

Step 2 Wait for the fields to be converted to the data types supported by CDM.

----End

2 Advanced Data Development Guidance

2.1 Dependency Policies for Periodic Scheduling

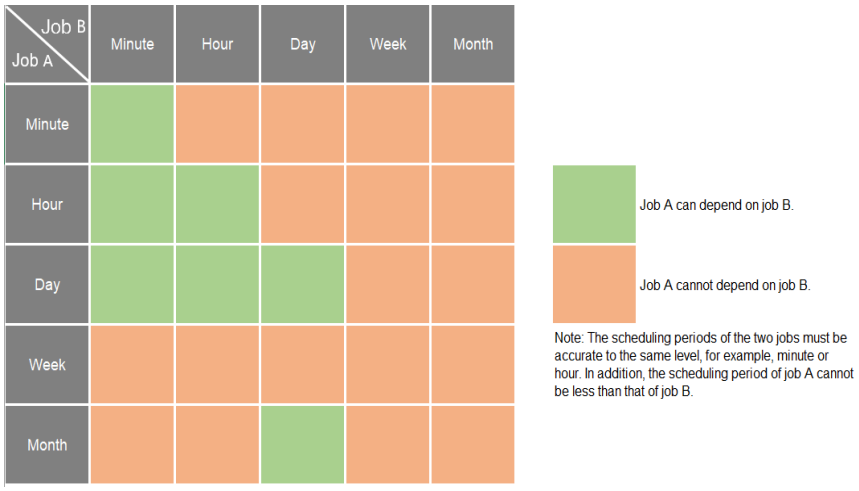
2.1.1 Comparison Between Traditional Periodic Scheduling Dependency and Natural Periodic Scheduling Dependency

Currently, DataArts Factory supports two types of job dependency policies, that is, dependency between jobs whose scheduling periods are traditional periods and dependency between jobs whose scheduling periods are natural periods.

In the traditional periodic scheduling dependency mode, jobs with the same same period can depend on each other, or jobs with a longer period can depend on those with a shorter period. Jobs with a shorter period cannot depend on those with a longer period. The details are as follows:

- Same-period dependency: The dependency time range is one period earlier than the current batch time.
- Cross-period dependency: The dependency time range is within the previous period.

Figure 2-1 Dependency between jobs with traditional periods



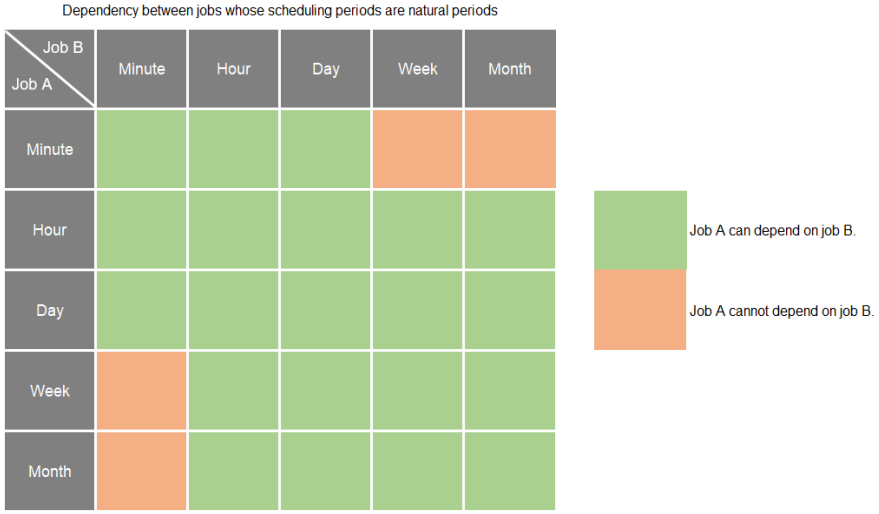
In the natural period scheduling dependency mode, jobs with the same period or different periods can depend on each other (jobs with a longer period can depend on those with a shorter period and vice versa). This mode is flexible and can meet complex scenarios of users. The detailed dependency inference rules are as follows:

- Rule 1: Infer the dependency between daily and hourly jobs by calendar day and hour.
- Rule 2: Infer the dependency between weekly and monthly jobs based on the calendar day of the current day.
- Rule 3: Jobs with a longer period depend only on the last job with a shorter period. For example, daily jobs depend on hourly jobs.

  Calendar day: [00:00:00–23:59:59]

  Calendar hour: [00:00–59:59]

Figure 2-2 Dependency between jobs with natural periods

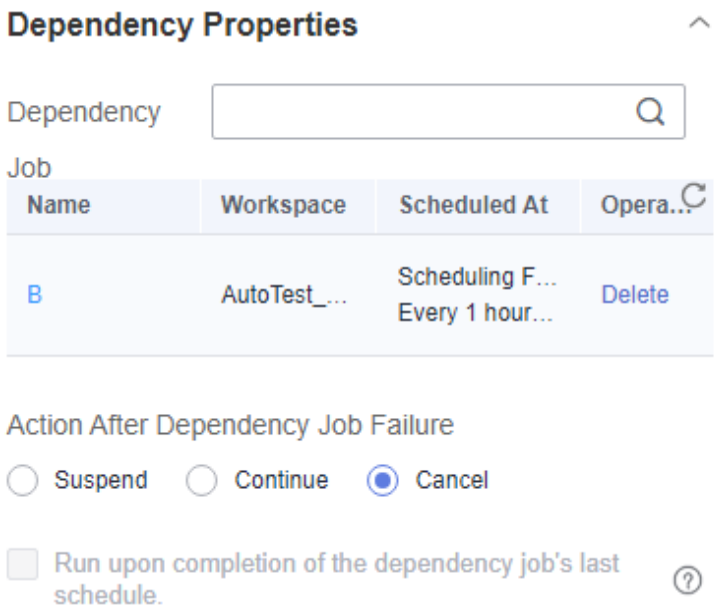


How Do I Determine Whether the Current Periodic Scheduling Dependency Is a Traditional Periodic Scheduling Dependency or a Natural Periodic Scheduling Dependency?

Job A is scheduled by hour, and job B is scheduled by day.

- When job A is being associated with a dependency job, if job B can be selected, the natural periodic scheduling dependency mode is used. (Jobs with a shorter period can depend only those with a longer period.)
- When job A is being associated with a dependency job, if job B cannot be selected, the traditional periodic scheduling dependency mode is used. (Jobs with a shorter period cannot depend only those with a longer period.)

Figure 2-3 Job dependency



2.1.2 Traditional Periodic Scheduling

Description

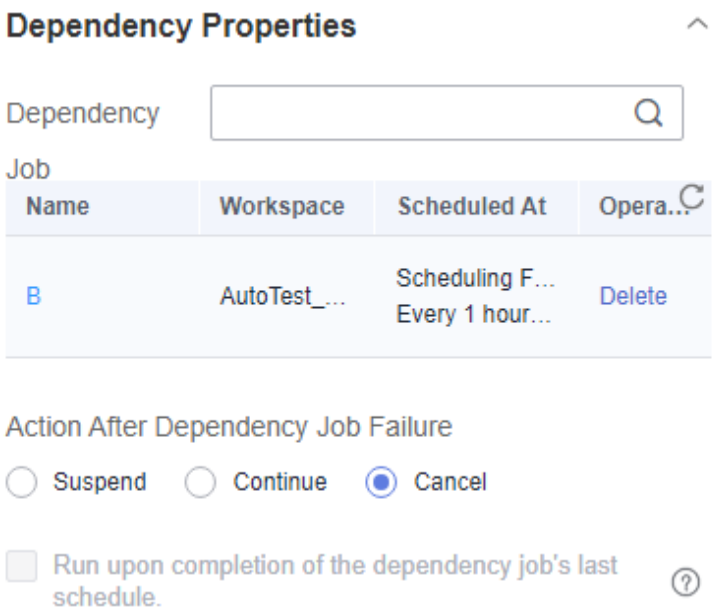
You can set a job that meets the scheduling period conditions as the dependency jobs for a job that is scheduled periodically. For details about how to set a dependency job, see **Setting Up Scheduling for a Job Using the Batch Processing Mode**.

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-period 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-period 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 2-4 Job dependency attributes



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**.

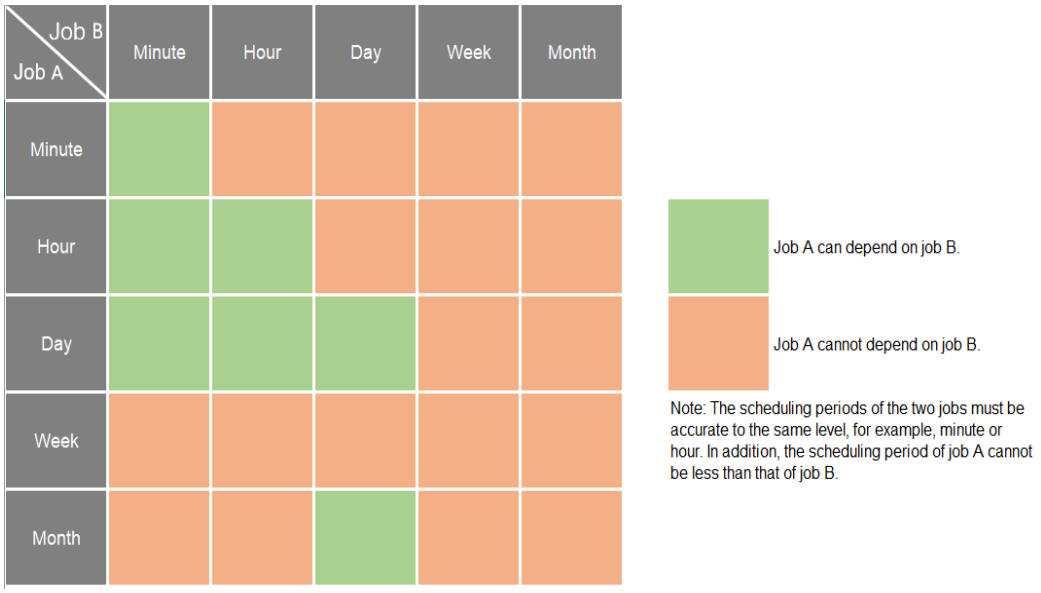
Setting Conditions of a Dependency Job

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 2-5 shows the requirements of the recurrences of the jobs that can function as the dependency jobs of other jobs

Figure 2-5 Job dependency



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-period 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 after 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

Time Point	Job B (Starting at 10:00 and Scheduled Every 10 Minutes)	Job A (Starting at 10:00 and Scheduled Every 20 Minutes)
10:30	Executed	-
...	...	...

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	-
...	...	...

NOTE

A downstream daily job can depend on an upstream daily job only if the upstream job is scheduled earlier than the downstream job.

- **Cross-period 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
...	...	...

2.1.3 Natural Periodic Scheduling

Description

DataArts Studio supports scheduling based on natural periods. Each node in the business process can be executed in an orderly manner based on the scheduling dependencies configured between the nodes, ensuring effective and timely output of business data.

Scheduling dependencies refer to the dependencies between upstream and downstream nodes. In DataArts Studio, a downstream node starts to run only after its upstream node is successfully executed.

After scheduling dependencies are configured, scheduling tasks can obtain correct data during running. (When the upstream node on which the current node depends is successfully executed, DataArts Studio determines that the latest data in the upstream table has been generated based on the status of the upstream node, and then the downstream node obtains the data.) This ensures that the upstream table data has been properly generated when the downstream node attempts to acquire data, preventing a data acquisition failure.

You can configure both same-period dependencies and dependencies on the previous period.

For details about the same-period dependency, see **Natural Periodic Scheduling: Same-Period Dependency**.

For details about the dependency on the previous period, see **Natural Periodic Scheduling: Dependency on the Previous Period**.

NOTE

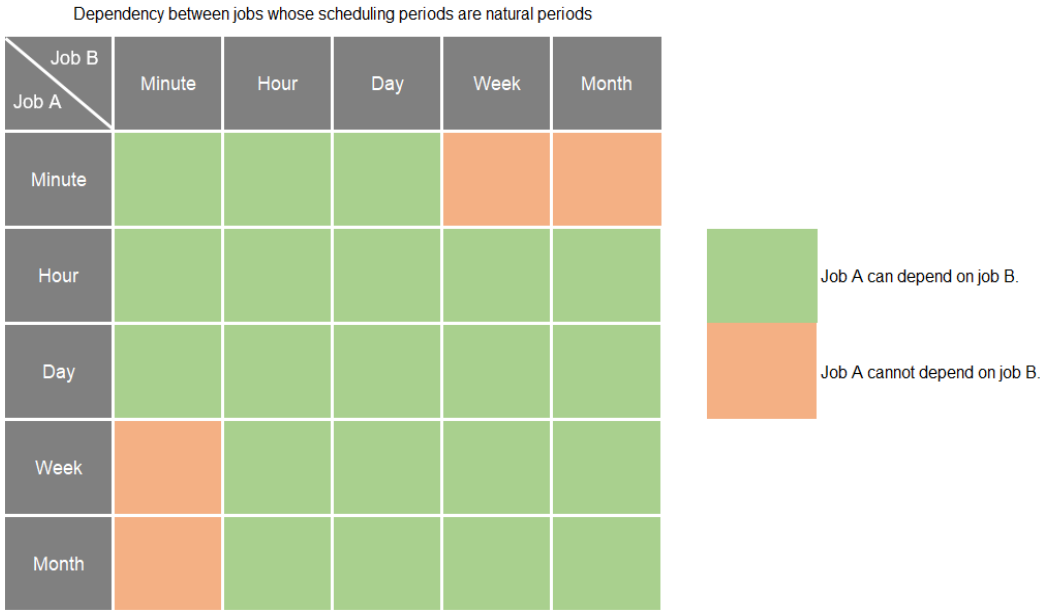
If the minute-based scheduling period cannot be exactly divided by the hour-based scheduling period, periodic scheduling is not performed strictly based on the interval. Instead, periodic scheduling is performed based on the cron expression rule. That is, periodic scheduling is triggered at the top of the hour.

2.1.4 Natural Periodic Scheduling: Same-Period Dependency

Introduction

Job A depends on Job B which has the same scheduling period as job A. The period unit can be minute, hour, day, week, or month. **Figure 2-6** lists the scheduling periods that can be configured for the dependency jobs of jobs with different scheduling periods.

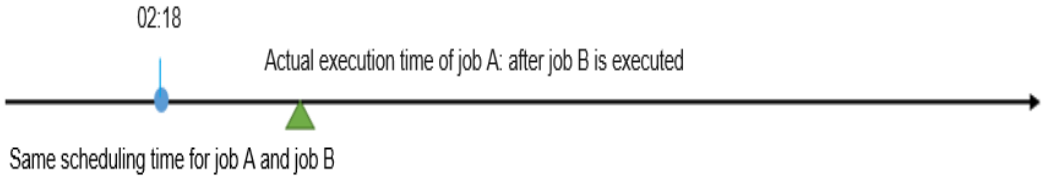
Figure 2-6 Dependency between jobs with the same period



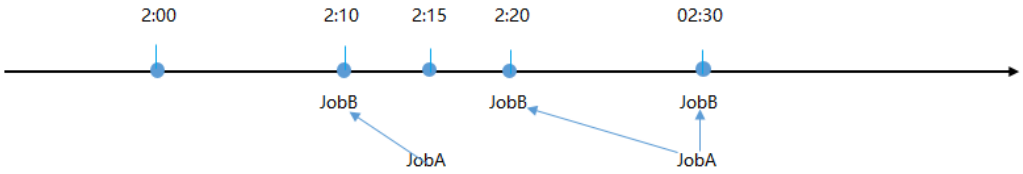
A Minute-Level Job Depends on Another Minute-Level Job

Rule: The minute is the minimum scheduling granularity, and there is no natural minute period. The dependency policy is to find the dependency instance in the previous scheduling period.

Example 1: Job A and job B have the same scheduling period (minutes), and A depends on B. At the same time point, A is executed after B is executed.



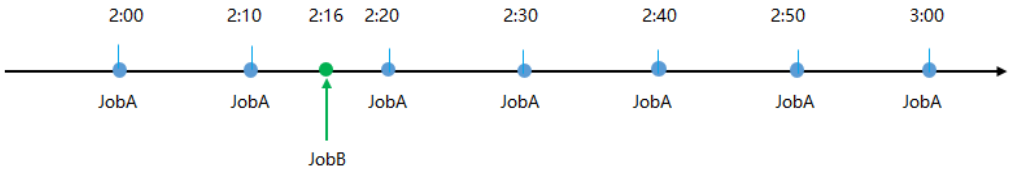
Example 2: Job A depends on job B. The scheduling period of job A is 15 minutes, and that of job B is 10 minutes. Job A depends on the job B instances within 15 minutes before job A's execution (including the current hour). The job A executed at 02:15 depends on a job B instance (executed at 02:10), and the job A executed at 02:30 depends on two job B instances (executed at 02:20 and 02:30, respectively). The boundary range is (0 minutes:15 minutes].



A Minute-Level Job Depends on an Hourly Job

Rule: A minute-level job is executed after the previous job which is executed each calendar hour is complete.

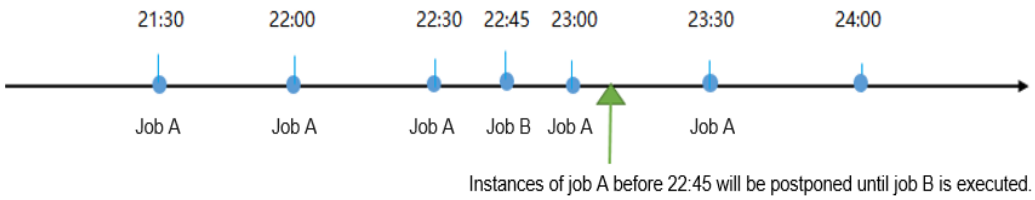
Example: Minute-level job A depends on hour-level job B. Job A is triggered every 10 minutes, and job B is executed at the 16th minute of each hour. In this case, job A is executed after job B is executed in the previous period.



A Minute-Level Job Depends on a Daily Job

Rule: A minute-level job which depends on a daily job is executed only after the daily job is executed.

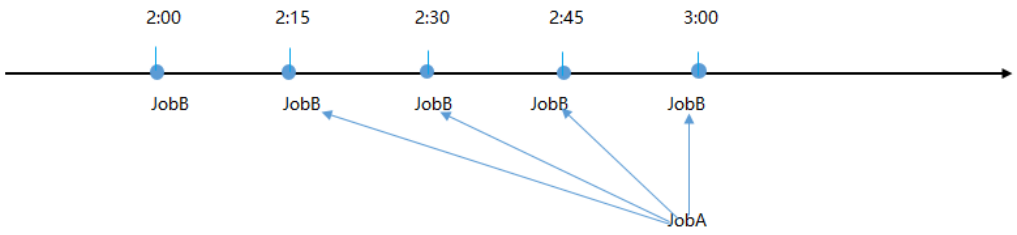
Example: Job A depends on job B. Job A is executed every 30 minutes, and job B is executed at 22:45. All the instances of job A before 22:45 will be executed after job B is executed.



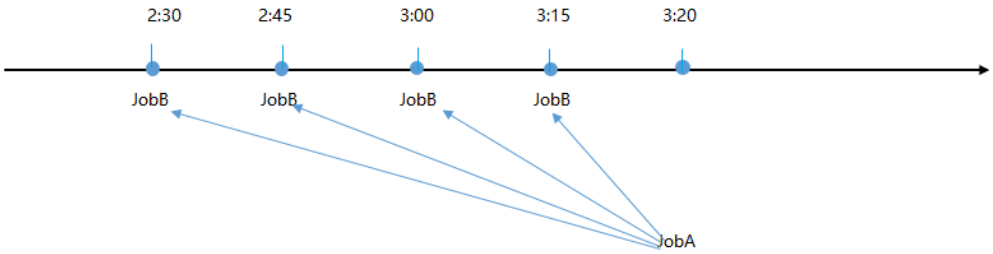
An Hourly Job Depends on a Minute-Level Job

Rule: An hourly job depends on a minute-level job. All the minute-level job instances within the last natural hour are executed (excluding the start time of the previous hour and including the start time of the current hour).

Example 1: Job A depends on job B. Job A is an hourly job and is executed at the top of each hour. Job B is executed every 15 minutes. After job B is executed, job A is executed.



Example 2: Job A depends on job B. Job A is an hourly job starting at 03:20. Job B is executed every 15 minutes. All the instances of job B within one hour before 03:20 are executed.



If you select **Recent**, the hourly job depends only on the latest running instance of the selected job. For example, if job A starts at 03:20, it depends on the instance of job B which is executed at 03:00.

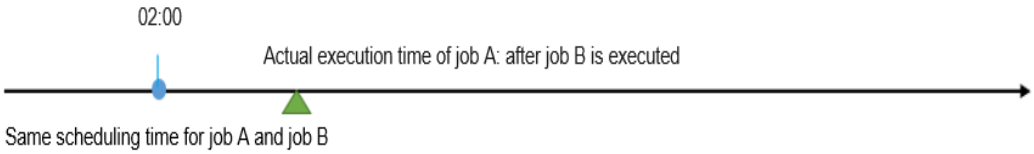
NOTE

If job A is scheduled at 00:00, it can depend on job B which is a minute-level job of yesterday.

An Hourly Job Depends on Another Hourly Job

Rule: Instances in each calendar hour depend on each other. The range boundary is the calendar hour [00:00, 00:59].

Example 1: Job A depends on job B. In the same calendar hour, job A is always executed after job B regardless of when they are executed.



Example 2: Job A depends on job B. Job A is executed at the fifth minute of each hour, and job B is executed at the 12th minute of each hour. A is executed after B is executed.

NOTE

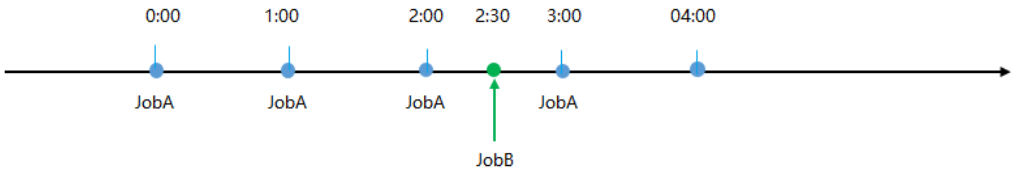
A job scheduled at a specified time point (discrete hour) depends on another job of this type:

- On a calendar day, the number of upstream and downstream tasks in the dependency relationship is the same, and the number of upstream and downstream periods is also the same.
- The number of upstream tasks is inconsistent with that of downstream tasks on a calendar day. A periodic instance generated on the day when a downstream task runs depends on the upstream instance that is closest to the scheduling time of the periodic instance. The periodic instance may depend on the upstream instances in an entire time range before the index or the nearest instance after the index.

An Hourly Job Depends on a Daily Job

Rule: An hourly job which depends on a daily job is executed only after the daily job is executed.

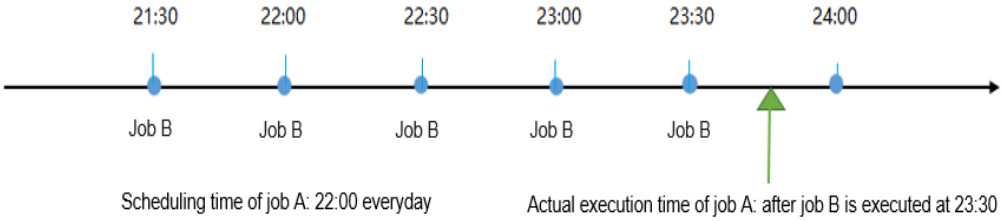
Example: Job A depends on job B. Job A is executed at the top of each hour, and job B is executed at 02:30. All the instances of job A before 02:30 will be executed after job B is executed.



A Daily Job Depends on a Minute-Level Job

Rule: Instances of a daily job depend on the instances of all minute-level jobs within a day.

Example: Job A is a daily job and is scheduled at 22:00 every day. It depends on job B which is scheduled every 30 minutes. Job A depends on all instances of job B on a calendar day. Job A is executed after the last instance of job B is executed.

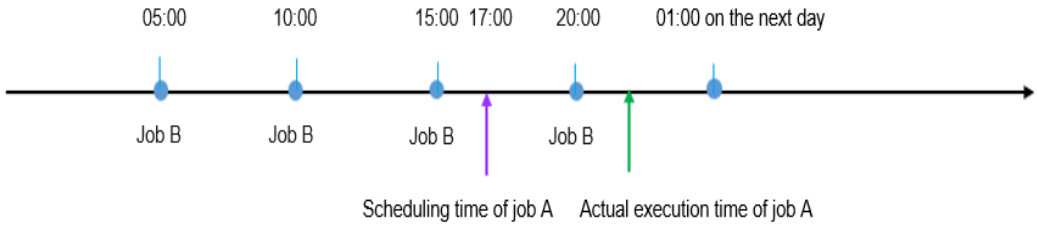


If you select **Recent**, the daily job depends only on the latest running instance of the selected job. For example, if job A is scheduled at 22:00 every day, it depends on the instance of job B which is executed at 21:30.

A Daily Job Depends on an Hourly Job

Rule: Instances of a daily job depend on the instances of all hourly jobs within a day. Job A is a daily job and depends on job B. Job A depends on all instances of job B on a calendar day. Job A is executed after the last instance of job B is executed.

Example: Job A depends on job B. If job A is scheduled at 17:00 every day and job B is executed every five hours starting from 00:00, job A is executed after the instance of job B is executed at 20:00.

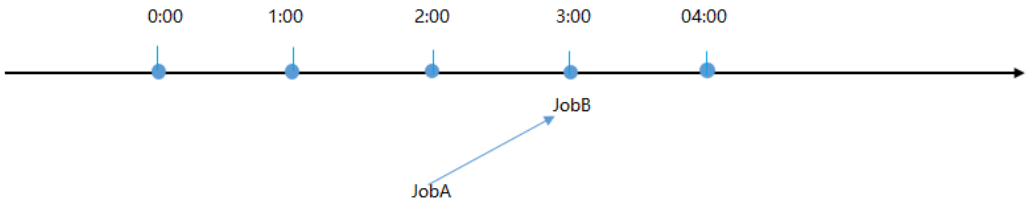


If you select **Recent**, the daily job depends only on the latest running instance of the selected job. For example, if job A is scheduled at 17:00 every day, it depends on the instance of job B which is executed at 15:00.

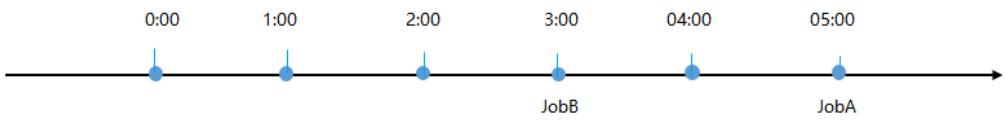
A Daily Job Depends on Another Daily Job

Rule: Jobs depend only on instances on the same calendar day. If job A depends on job B within the same calendar day, job A is always executed after job B regardless of their execution time. The day range is [00:00:00, 23:59:59].

Example 1: Job A depends on job B. Job A is executed at 02:00, and job B is executed at 03:00. Job A is executed after job B is executed at 03:00.



Example 2: Job A depends on job B. Job A is executed at 05:00, and job B is executed at 03:00. Job A is executed at 05:00 after job B is executed.



A Daily Job Depends on a Weekly Job

Rule: dependence on calendar days.

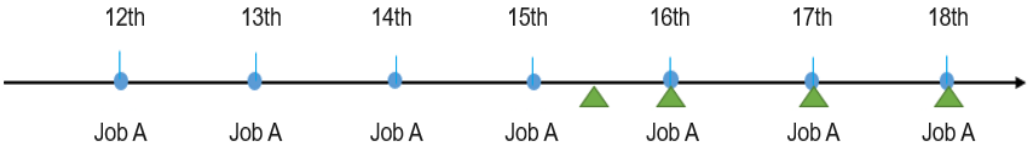
Example 1: Job A depends on job B. Job A is executed every day, and job B is executed every Wednesday. If the day when job A is scheduled to be executed is not Wednesday, job B is not executed, and job A is directly executed.

Example 2: Job A depends on job B. Job A is executed every day, and job B is executed every Wednesday. If the day when job A is scheduled to be executed is Wednesday, job B is executed, and job A is executed after job B is executed.

A Daily Job Depends on a Monthly Job

Rule: A daily job which depends on a monthly job is executed only after the monthly job is executed.

Example: Job A depends on job B. Job A is executed once a day, and job B is executed once on the 15th day of each month. On the 15th day of each month, job A is executed after job B is executed.



A Weekly Job Depends on an Hourly Job

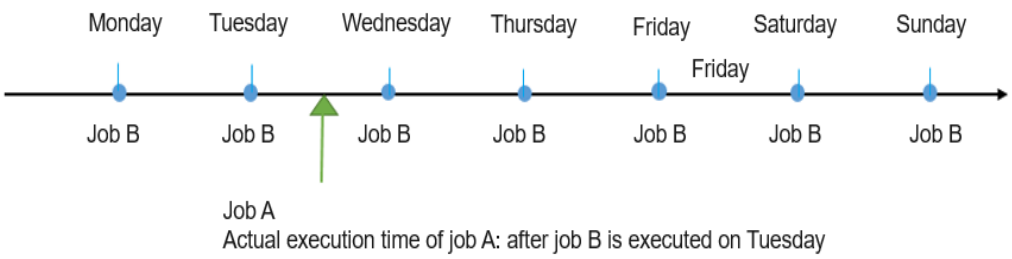
Rule: On a calendar day from 00:00 of the previous day (included) to 00:00 of the current day (excluded), the weekly job A is executed after all hourly tasks of job B are executed.

Example: Job A depends on job B. Job A is scheduled every Monday, and job B is executed at the 50th minute of each hour. Job A is executed after the last task of job B is executed at 23:50 on Monday.

A Weekly Job Depends on a Daily Job

Rule: Weekly jobs depend only on jobs scheduled and executed on the same day.

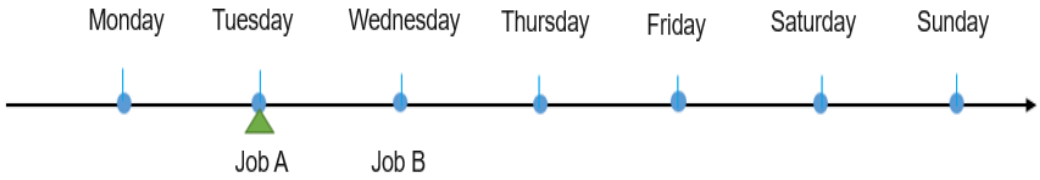
Example: Job A depends on job B. Job A is scheduled to be executed on Tuesday, and job B is executed every day. On Tuesday, Job A is executed after job B is executed.



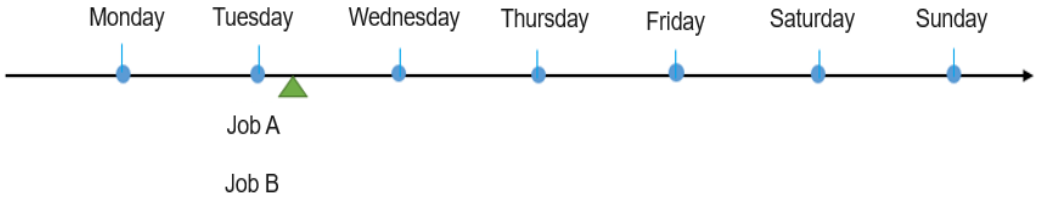
A Weekly Job Depends on Another Weekly Job

Rule: Weekly jobs depend only on job instances scheduled and executed on the same day.

Example 1: Job A depends on job B. Job A is scheduled to be executed on Tuesday, and job B is scheduled to be executed on Wednesday. Job A is executed on Tuesday rather than on Wednesday after job B is executed.



Example 2: Job A and job B are both executed on Tuesday, and Job A depends on job B. Job A is executed after job B is executed.



A Weekly Job Depends on a Monthly Job

Rule: dependence on calendar days.

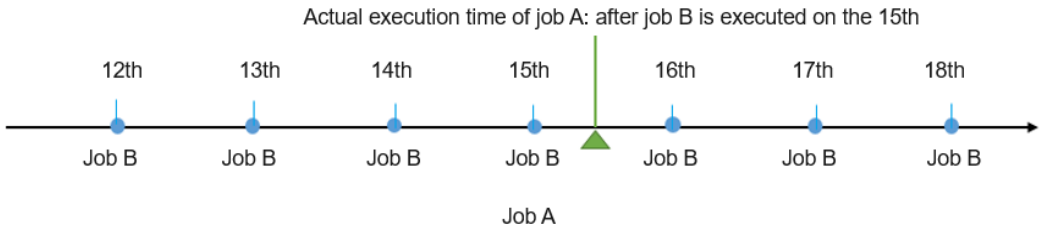
Example 1: Job A depends on job B. Job A is executed every Wednesday, and job B is executed on the 10th day of each month. If job A is scheduled to be executed on the 10th day of a month, job A will be executed after job B is executed.

Example 2: Job A depends on job B. Job A is executed every Wednesday, and job B is executed on the 10th day of each month. If job A is not scheduled to be executed on the 10th day of a month, job A will be directly executed.

A Monthly Job Depends on a Daily Job

Rule: A monthly job can be executed as long as the current daily job is complete.

Example: Job A depends on job B. Job A is a monthly job and depends on job B which is a daily job. Job A is executed after job B is executed on the day when job A is scheduled to be executed.



A Monthly Job Depends on a Weekly Job

Rule: dependence on calendar days.

Example 1: Job A depends on job B. Job A is executed on the 10th day of each month, and job B is executed every Wednesday. If the day when job A is scheduled to be executed is not Wednesday, job B is not executed, and job A is directly executed.

Example 2: Job A depends on job B. Job A is executed on the 10th day of each month, and job B is executed every Wednesday. If the day when job A is scheduled to be executed is Wednesday, job A is executed after job B is executed.

A Monthly Job Depends on Another Monthly Job

Rule: dependence on calendar days.

Example 1: Job A depends on job B. Job A is executed on the first day of each month, and job B is executed on the second day of each month. On the first day of a month, job A is executed normally without being blocked by job B.

Example 2: Job A depends on job B. Job A and job B are both executed on the second day of each month. Job A is executed after job B is executed.

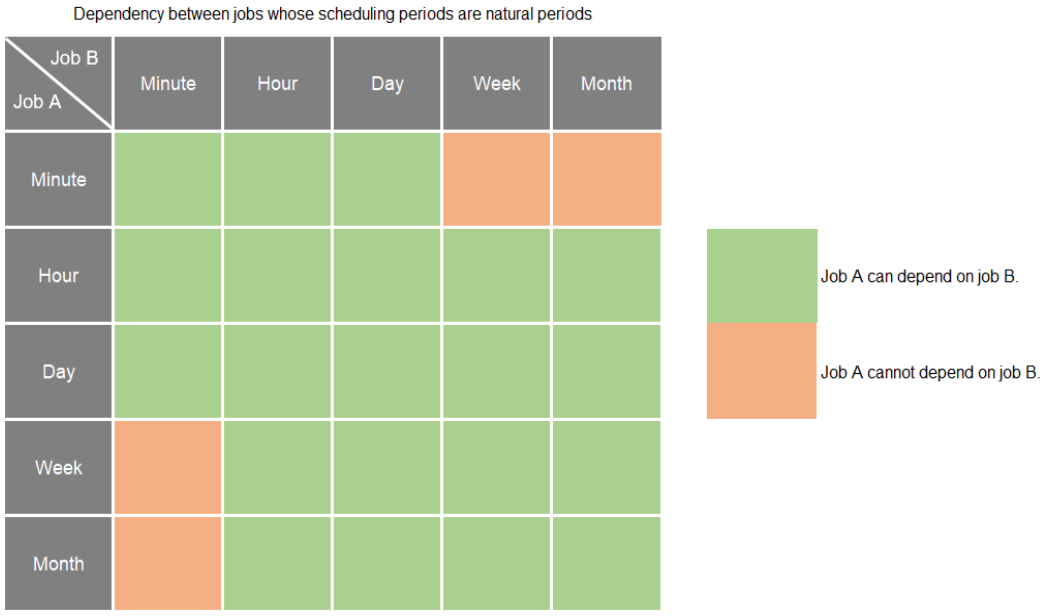
Example 3: Job A is executed on the third day of each month, and job B is executed on the second day of each month. Job A depends on job B.

2.1.5 Natural Periodic Scheduling: Dependency on the Previous Period

Introduction

The natural periods for job scheduling can be classified into the following types: minute, hour, day, week, or month. **Figure 2-7** lists the scheduling periods that can be configured for the dependency jobs of jobs with different scheduling periods.

Figure 2-7 Dependency between a job and that of the previous period



The scheduling of job A depends on the previous scheduling period of job B. The following scenarios are involved.

A Minute-Level Job Depends on Another Minute-Level Job

Rule: The minute is the minimum scheduling granularity, and there is no natural minute period. The dependency policy is to find the dependency instance of the job with a longer scheduling period in the previous scheduling period.

Example: Job A depends on job B. Job A is executed every 10 minutes, and job B is executed every 15 minutes, both starting from the 0th minute of each hour. Job A is executed after job B which starts from the 45th minute of the previous hour is complete.

A Minute-Level Job Depends on an Hourly Job

Rule: A minute-level job is executed after the previous job which is executed each calendar hour is complete.

Example: Minute-level job A depends on hour-level job B. Job A is triggered every 10 minutes, and job B is executed at the 18th minute of each hour. In this case, job A is executed at the 10th minute of the period.

A Minute-Level Job Depends on a Daily Job

Rule: A minute-level job depends on the previous period of the daily job and can be executed only after the previous daily job is executed.

Example: Job A depends on job B. Job A is executed every five minutes, and job B is executed at 02:30 every day. Job A depends on the instance of job B after 02:30 on the previous day.

An Hourly Job Depends on a Minute-Level Job

Rule: An hourly job depends on a minute-level job. All the minute-level job instances within the last natural hour are executed (excluding the start time of the previous hour and including the start time of the current hour).

Example 1: Job A is an hourly job and is executed at the 0th minute every hour. Job B is executed every 15 minutes. Job A depends on the instance of job B which is executed at the 45th minute in the previous hour.

Example 2: Job A is an hourly job starting at 03:20. Job B is executed every 15 minutes. Job A depends on the instance of job B generated at 03:15.

An Hourly Job Depends on Another Hourly Job

Rule: Instances in each calendar hour depend on each other. The range boundary is [00:00, 00:59]. The dependency policy is to find dependency instances from the previous scheduling period of a job with a long scheduling period.

Job A depends on job B. In the same calendar hour, job A is always executed after job B is executed in the previous period.

Example: Job A is executed at the fifth minute of each hour and job B is executed at the 12th minute. Job A depends on the instance generated in the previous hour at the fifth minute of each hour.

NOTE

A job scheduled at a specified time point (discrete hour) depends on another job of this type:

- On a calendar day, the number of upstream and downstream tasks in the dependency relationship is the same, and the number of upstream and downstream periods is also the same.
- The number of upstream tasks is inconsistent with that of downstream tasks on a calendar day. A periodic instance generated on the day when a downstream task runs depends on the upstream instance that is closest to the scheduling time of the periodic instance. The periodic instance may depend on the upstream instances in an entire time range before the index or the nearest instance after the index.

An Hourly Job Depends on a Daily Job

Rule: An hourly job which depends on a daily job is executed only after the daily job is completed in the previous hour.

Example: Job A depends on job B. If job A is executed at the top of each hour, and job B is executed at 02:30, job A depends on the instance of job B at 02:30 on the previous day.

A Daily Job Depends on a Minute-Level Job

Rule: Instances of a daily job depend on the instances of all minute-level jobs on the previous day.

Example: Job A is a daily job and depends on job B (minute-level). Job A depends on the last instance of job B on the current day. Job A is executed after the last instance of job B is executed.

A Daily Job Depends on an Hourly Job

Rule: Instances of a daily job depend on the instances of an hourly job on the previous day.

Example: Job A is a daily job and depends on hourly job B. Job A depends on the instances of an hourly job in the last period of the previous day.

A Daily Job Depends on Another Daily Job

Rule: A job depends on the instance of the previous period of a job which is scheduled every calendar day. If job A depends on job B within the same calendar day, job A always depends on the instance of job B in the previous period regardless of the execution time configured for job A and job B. The day range is [00:00:00, 23:59:59].

Example: If job A is executed at 02:00 and job B is executed at 03:00, job A depends on the instance of job B executed at 03:00 in the previous period and is executed at 02:00 in the current period.

A Daily Job Depends on a Weekly Job

Rule: dependence on calendar days.

Example 1: Job A depends on job B. Job A is executed every day, and job B is executed every Wednesday. If the day before the day when job A is scheduled to be executed is not Wednesday, job B is not executed, and job A is directly executed.

Example 2: Job A depends on job B. Job A is executed every day, and job B is executed every Wednesday. If the day before the day when job A is scheduled to be executed is Wednesday, job B is executed, and job A is executed after job B is executed.

A Daily Job Depends on a Monthly Job

Rule: A daily job which depends on a monthly job is executed only after the monthly job is executed in the previous period .

Example: Job A depends on job B. Job A is executed once a day, and job B is executed once on the 15th day of each month. The execution of job A depends on the running instance of job B on the 15th day of the previous month.

A Weekly Job Depends on an Hourly Job

Rule: On a calendar day from 00:00 of the previous day (included) to 00:00 of the current day (excluded), the weekly job A is executed after all hourly tasks of job B are executed.

Example: Job A depends on job B. Job A is scheduled every Monday, and job B is executed at the 50th minute of each hour. Job A is executed after the last task of job B is executed at 23:50 on the last Sunday.

A Weekly Job Depends on a Daily Job

Rule: A weekly job can be executed only after the daily job of the previous day is complete.

Example: Job A depends on job B. Job A is executed once a day, and job B is executed once on every Monday of each month. The execution of job B depends on the running instances of job A on the previous day.

A Weekly Job Depends on Another Weekly Job

Rule: A weekly job which depends on a weekly job of the previous day can be executed only after the weekly job of the previous day is complete. If there is no instance for the previous day, no dependency is required.

Example: Job A depends on job B. Job A is executed once on every Monday, and job B is executed once on every Tuesday of each month. The execution of job B depends on the running instances of job A on Monday.

A Weekly Job Depends on a Monthly Job

Rule: dependence on calendar days.

Example 1: Job A depends on job B. Job A is executed every Wednesday, and job B is executed on the 10th day of each month. If the day before the day when job A is scheduled to be executed is the 10th day of a month, job A will be executed after job B is executed.

Example 2: Job A depends on job B. Job A is executed every Wednesday, and job B is executed on the 10th day of each month. If the previous day of the day when job A is scheduled to be executed is not the 10th day of a month, job A will be directly executed.

A Monthly Job Depends on a Daily Job

Rule: A monthly job can be executed only after the daily job of the previous day is complete.

Example: Job A depends on job B. Job B is executed once every day, and job A is executed once every month. The execution of job A depends on the running instances of job B on the previous day.

A Monthly Job Depends on a Weekly Job

Rule: dependence on calendar days.

Example 1: Job A depends on job B. Job A is executed on the 10th day of each month, and job B is executed every Wednesday. If the day before the day when job A is scheduled to be executed is not Wednesday, job B is not executed, and job A is directly executed.

Example 2: Job A depends on job B. Job A is executed on the 10th day of each month, and job B is executed every Wednesday. If the day before the day when job A is scheduled to be executed is Wednesday, job B is executed, and job A is executed after job B is executed.

A Monthly Job Depends on Another Monthly Job

Rule: dependence on calendar days.

Example 1: Job A depends on job B. Job A is executed on the first day of each month, and job B is executed on the second day of each month. On the first day of a month, job A is executed normally without being blocked by job B.

Example 2: Job A depends on job B. Job A is executed on the third day of each month, and job B is executed on the second day of each month. Job A can be executed after job B is executed on the second day of each month.

Example 3: Job A is executed on the third day of each month, and job B is executed on the second day of each month. Job A depends on job B.

2.2 Using PatchData

Scenario

In the migration of a project, if you want to supplement historical business data in a previous period and view details of the historical data, PatchData can meet your requirements.

A job executes a scheduling task to generate a series of instances in a certain period of time. This series of instances are called PatchData. PatchData can be used to fix the job instances that have data errors in the historical records or to build job records for debugging programs.

NOTE

- In addition to SQL scripts, PatchData supports other nodes.
- If the content of a SQL script changes, the PatchData job runs the latest script.
- When you use PatchData, if the variable in the SQL statement is **DATE**, enter **${DATE}** in the script. The script parameter **DATE** is then automatically added to the job parameters, and its value can be an EL expression. If the variable is a time variable, enter the expression of the **DateUtil** embedded object. The platform automatically converts the expression into a historical date. For details about how to use EL expressions, see **EL Expressions**.
- PatchData jobs support script parameters and global environment variables as well as job parameters.

Constraints

- PatchData is available only when periodic scheduling is configured for the data development job.

Example

Scenario

Among the product data tables of a company, there is a source data table A that records the product sales amount. To import the historical product sales amount to the destination table B, you can create a PatchData job.

Table 1 lists the source and destination tables.

Table 2-5 Source and destination tables

Source Table	Destination Table
A	B

Procedure

1. Prepare the source and destination tables. To facilitate subsequent job execution and verification, you need to create a source DWS table and a destination DWS table and insert data into the tables.
   a. Create a DWS table. You can create a DWS SQL script on the DataArts Factory console of DataArts Studio and run the following SQL statements:
   ```
   /* Create tables. */
   CREATE TABLE A (PRODUCT_ID INT, SALES INT, DATE DATE);
   CREATE TABLE B (PRODUCT_ID INT, SALES INT, DATE DATE);
   ```
   b. Insert sample data into the source data table. You can create a DWS SQL script on the DataArts Factory console of DataArts Studio and run the following SQL statements:
   ```
   /* Insert sample historical data into the source table. */
   INSERT INTO A VALUES ('1','60', '2022-03-01');
   ```



```
INSERT INTO A VALUES ('2','80', '2022-03-01');
INSERT INTO A VALUES ('1','50', '2022-02-28');
INSERT INTO A VALUES ('2','55', '2022-02-28');
INSERT INTO A VALUES ('1','60', '2022-02-27');
INSERT INTO A VALUES ('2','45', '2022-02-27');
```

2. Develop a PatchData script. Ensure that the script expression contains a time variable. (For example, if the variable in the SQL statement is **DATE**, enter **${DATE}** in the script.) You can set the expression for script parameter **DATE** in job parameter settings in **3**.

   On the **Develop Script** page, enter following statement in the editor:

   ```
   INSERT INTO B (SELECT * FROM A WHERE DATE = ${DATE})
   ```

   Figure 2-8 Developing a script

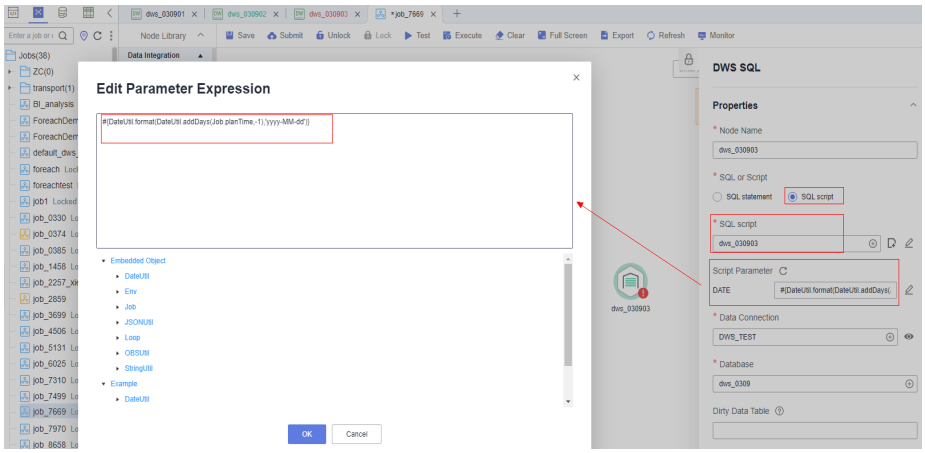
   ```
   -- DWS sql
   -- ********************************************************************** --
   -- author: [redacted]
   -- create time: 2023/05/23 17:03:02 GMT+08:00
   -- ********************************************************************** --
   INSERT INTO B (SELECT * FROM A WHERE DATE = ${DATE})
   ```

   After compiling the script, save it and submit the latest version.

3. Develop a PatchData batch processing job. When developing the job, you need to configure the node attributes and scheduling period.

   In the left navigation pane of the DataArts Factory console, choose **Data Development** > **Develop Job**.

   Figure 2-9 Node parameters



NOTE

- If the job-associated SQL script uses a parameter, the parameter name (such as **DATE**) is displayed. Set the parameter value in the text box next to the parameter name. The parameter value can be an EL expression. For details about EL expressions, see **Expression Overview**.

  If the parameter is time, view the example expression of the DateUtil embedded object. The platform automatically replaces the parameter with the historical date of the patch data (determined by the service date of the patch data).

  You can also directly enter a SQL expression.

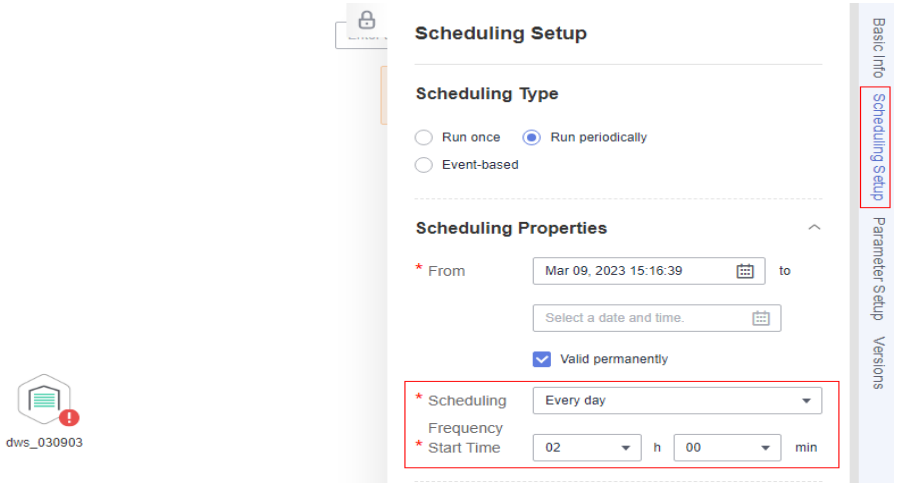
- If the parameters of the associated SQL script change, you can click to synchronize the change or click to edit the parameters.
- The following is an example of script parameters:

  Example: #{DateUtil.format(DateUtil.addDays(Job.planTime,-1),'yyyy-MM-dd')}

  - **Job.planTime** indicates the planned job time, and *yyyy-MM-dd* indicates the time format.
  - If the planned job time is March 2, the previous day is March 1. The planned job time will be replaced by the configured patch data service date.
  - The **Job.planTime** is converted into a time in the *yyyy-MM-dd* format using an expression.

Configure the scheduling period of the PatchData job. Click **Scheduling Setup** and set **Scheduling Frequency** to **Every day**.

Figure 2-10 Configuring the scheduling period



NOTE

- If **Scheduling Frequency** is set to **Every day**, the job is scheduled every day, and a PatchData instance is generated. You can view the statuses of PatchData instances on the **Monitor Instance** page. On the **Monitor Instance** page, view the instance information about the job and perform more operations on instances as required.
- The job scheduling time takes effect from March 9, 2023, and the job is scheduled at 02:00 every day.
- Run the following SQL statement to check whether destination table B contains data of source table A:

  SELECT * FROM B

After configuring the parameters, save and submit the latest version of the job and test the job.

Click **Execute** to run the job.

4. Create a PatchData task.

   After creating a periodic job, you need to configure PatchData for the job.

   a. In the left navigation pane of DataArts Factory, choose **Monitoring** > **Monitor Job**.

   b. Click the **Batch Job Monitoring** tab. In the **Operation** column of the job, choose **More** > **Configure PatchData**. The **Configure PatchData** page is displayed.

      If you want to supplement historical data from February 27, 2023 to March 1, 2023, set **Date** to **Feb 28, 2023 00:00:00 – Mar 02, 2023 23:59:59**. The system automatically transfers the configured date to the planned job time. In the expression of the script time variable **DATE**, the defined time is the planned job time minus one day. That is, the time of the day before the planned job time is the time range (**Feb 27, 2023 to Mar 1, 2023**) for PatchData.

Figure 2-11 Configuring PatchData

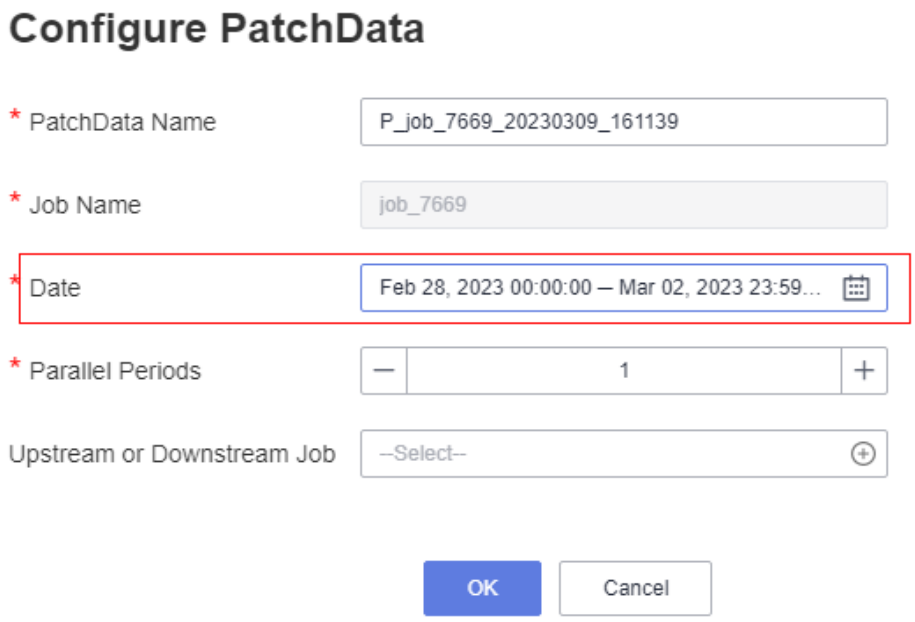


Table 2-6 Description

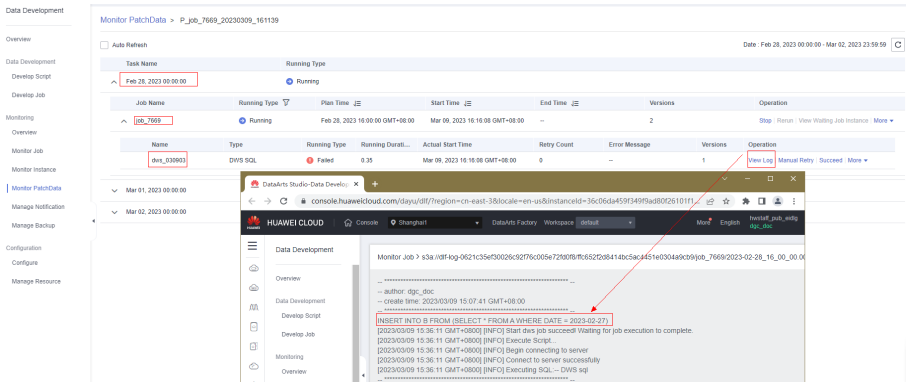
Parameter	Description
PatchData Name	Name of the automatically generated PatchData task. The value can be modified.
Job Name	Name of the job that requires PatchData, which is automatically displayed

Parameter	Description
Date	Period of time when PatchData is required. This date is transferred to the planned job time. When the job is executed, the planned job time is replaced by the time in the PatchData. **NOTE** PatchData can be configured for a job multiple times. However, avoid configuring PatchData multiple times on the same date to prevent data duplication or disorder. If you select **Patch data in reverse order of date**, the patch data of each day is in positive sequence. **NOTE** • This function is applicable when the data of each day is not coupled with each other. • The PatchData job will ignore the dependencies between the job instances created before this date.
Parallel Periods	Number of instances to be executed at the same time. A maximum of five instances can be executed at the same time. **NOTE** Set this parameter based on the site requirements. For example, if a CDM job instance is used, data cannot be supplemented at the same time. The value of this parameter can only be set to **1**.
Upstream or Downstream Job	This parameter is optional. Select the downstream jobs (jobs that depend on the current job) that require PatchData. You can select multiple jobs.

c. Click **OK**. The system starts to run the PatchData task based on the configured scheduling period.

d. On the **Monitor PatchData** page, you can view the PatchData task status, date, number of parallel periods, PatchData job name, and stopped tasks. You can also view details about the PatchData task.

Figure 2-12 Querying PatchData details



e. Run the following SQL statement to check whether destination table B contains historical data of source table A:

```
SELECT * FROM B
```

2.3 Setting the Job Scheduling Time to the Last Day of Each Month

Scenario

When configuring job scheduling, you can set the scheduling time to the last day of each month using either of the two methods provided in the following table.

Table 2-7 Setting the scheduling time to the last day of each month

Method	Advantage	Procedure
Set the scheduling frequency to every day and use a condition expression to determine whether a day is the last day of each month.	This method applies to multiple scenarios. You can compile condition expressions to flexibly schedule jobs, for example, on the last day or 7th of each month.	**Method 1**
Set the scheduling frequency to every month and select the last day of each month.	You can set a specific job scheduling time instead of compiling any statements.	**Method 2**

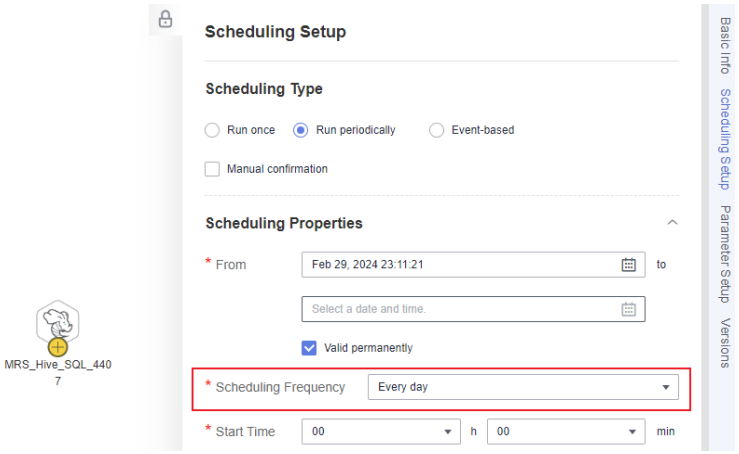
Method 1

In DataArts Studio, create a job that is scheduled every day and add an empty Dummy node (which does not process data) to the job. You can set a condition expression on the connection line between the Dummy node and its subsequent node to check whether the current day is the last day of the current month. If it is

the last day, the subsequent nodes are executed. Otherwise, the subsequent nodes are skipped.

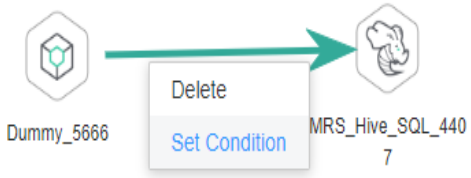
1. In the left navigation pane of the DataArts Factory console, choose **Data Development** > **Develop Job**.
2. Set **Scheduling Frequency** to **Every day**.

   Figure 2-13 Setting Scheduling Frequency to Every day



3. Right-click the connection line between the Dummy node and its subsequent node and select **Set Condition** to configure a condition expression that is used to determine whether to execute the subsequent node.

   Figure 2-14 Configuring a condition expression

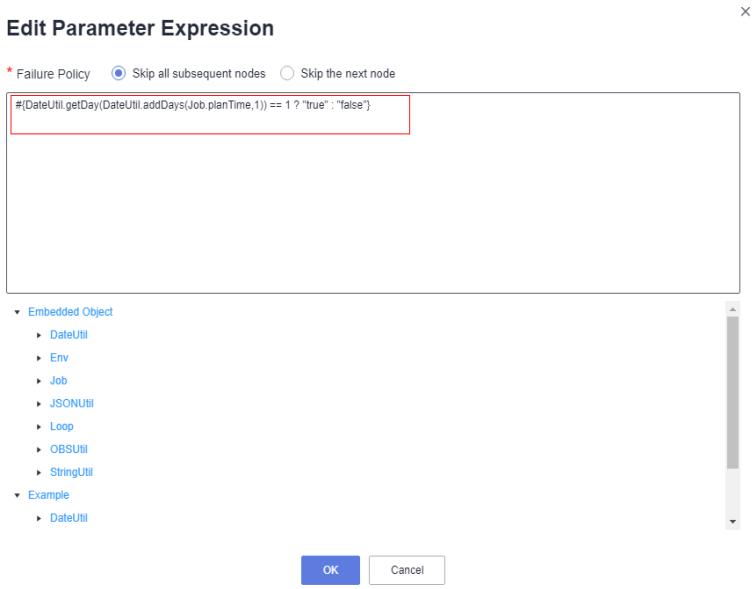


4. Configure the expression as follows:

```
#{DateUtil.getDay(DateUtil.addDays(Job.planTime,1)) == 1 ? "true" : "false"}
```

   The expression is used to obtain the current time and check whether the next day is 1st of a month. If yes, the current day is the last day of the current month, and the subsequent node will be executed; if no, the subsequent node will be skipped.

Figure 2-15 Condition expression



For example, if you want a job to be executed on the last day and seventh day of each month, perform the following operation:

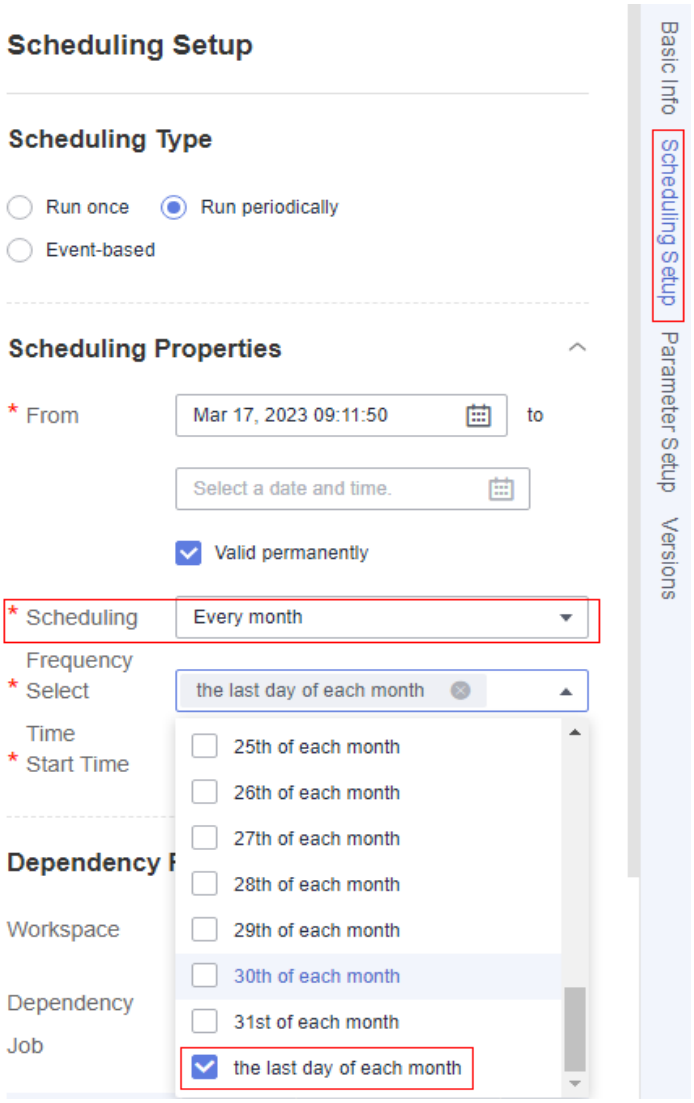
Configure the following expression to check whether the current day is 7th:

```
#{DateUtil.getDay(Job.planTime) == 7 ? "true" : "false"}
```

Method 2

1. In the left navigation pane of DataArts Factory, choose **Development** > **Develop Job**.
2. Click **Scheduling Setup** on the right of the job canvas.
3. Set **Scheduling Type** to **Run periodically**, **Scheduling Frequency** to **Every month**, and **Select Time** to **the last day of each month**.

Figure 2-16 Setting the scheduling time to the last day of each month



After the scheduling time is configured, the job will be automatically executed on the last day of each month.

2.4 Obtaining the Output of an SQL Node

This section describes how to obtain the output of an SQL node and apply the output to subsequent nodes or judgment in job development.

Scenario

When you use EL expression **#{Job.getNodeOutput("Name of the previous node")}** to obtain the output of the previous node, the output is a two-dimensional array, for example, **[["Dean",...,"08"],...,["Smith",...,"53"]]**. To obtain the values in the array, use either of the methods provided in **Table 2-8**.

Table 2-8 Methods for obtaining output values

Method	Key Configuration	Application Scenario Requirements
Obtaining Output Value Using StringUtil	If the output of the SQL node contains only one field, for example **[["11"]]**, you can use the StringUtil EL expression with an embedded object to split the two-dimensional array and obtain the field value in the output of the previous node. #{StringUtil.split(StringUtil.split(StringUtil.split(Job.getNodeOutput("***Name of the previous node***"),"]")[0],"[")[0],"\\"")[0]}	This method is easy to use but has the following requirements on application scenarios: • The output of the previous SQL node contains only one field, for example, **[["11"]]**. • The output value is a string. The application scenario must support the string data type. For example, if the IF condition needs to be used to judge the size of the output value, the string type is not supported. In this case, this method cannot be used.
Obtaining Output Values Using the For Each Node	Use the For Each node to cyclically obtain the values in the two-dimensional array in the dataset. • For Each node dataset: #{Job.getNodeOutput('***Name of the previous node***')} • Subjob parameters of the For Each node: #{Loop.current[***Index***]}	This method is applicable to more scenarios, though jobs need to be split into main jobs and subjobs.

Obtaining Output Value Using StringUtil

Scenario

The StringUtil EL expression with an embedded object is used to split the two-dimensional array result and obtain the output field value of the previous node, which is a string.

In this example, the MRS Hive SQL node returns a two-dimensional array that contains a single field. The data sent by the Kafka Client node is defined as the StringUtil EL expression with an embedded object. You can use this expression to split the two-dimensional array and obtain the output field value of the MRS Hive SQL node.

NOTE

To make it easy to view the obtained value, this example uses the Kafka Client node. In practice, you can select a subsequent node type as needed. By using a StringUtil EL expression with an embedded object on the node, you can obtain the data value returned by the previous node.

Figure 2-17 Example job

The key configuration of the Kafka Client node is the **Sent Content** parameter. Set it as follows:

```
#{StringUtil.split(StringUtil.split(StringUtil.split(Job.getNodeOutput("count95"),"]")[0],"[")[0],"\\"")[0]}
```

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**.

Step 3 Create table **student_score**. Create a temporary Hive SQL script, select a Hive connection and database, paste the following SQL statement, and run the script. After the script is successfully executed, delete it.

```
CREATE TABLE `student_score` (`name` String COMMENT '', `score` INT COMMENT '');
INSERT INTO
    student_score
VALUES
    ('ZHAO', '90'),
    ('QIAN', '88'),
    ('SUN', '93'),
    ('LI', '94'),
    ('ZHOU', '85'),
    ('WU', '79'),
    ('ZHENG', '87'),
    ('WANG', '97'),
    ('FENG', '83'),
    ('CEHN', '99');
```

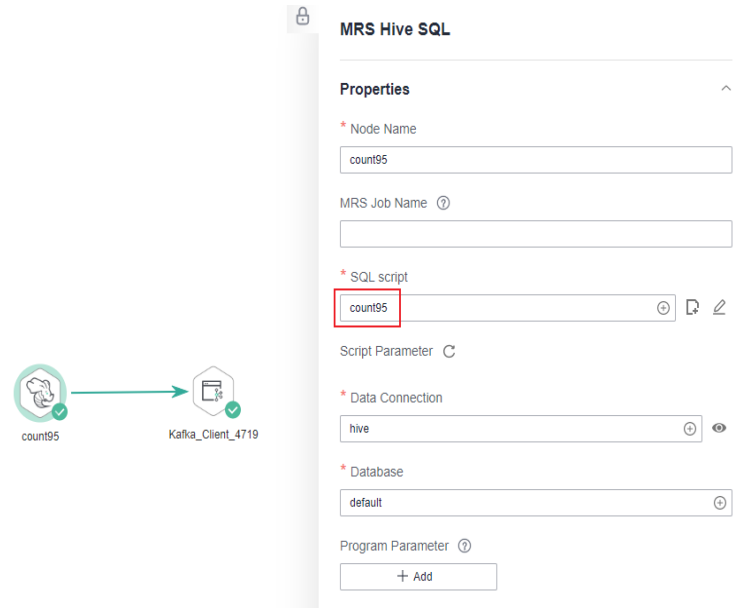
Step 4 Create the Hive SQL script to be invoked by the MRS Hive SQL node. Create a Hive SQL script named **count95**, select a Hive connection and database, paste the following SQL statement, and submit a version.

```
--Obtain the number of students whose scores are higher than 95 from the student_score table.--
SELECT count(*) FROM student_score WHERE score> "95" ;
```

Step 5 On the **Develop Job** page, create a data development job. Drag an MRS Hive SQL node and a Kafka Client node and drop them on the canvas. Click and hold to connect the nodes, as shown in **Figure 2-17**.

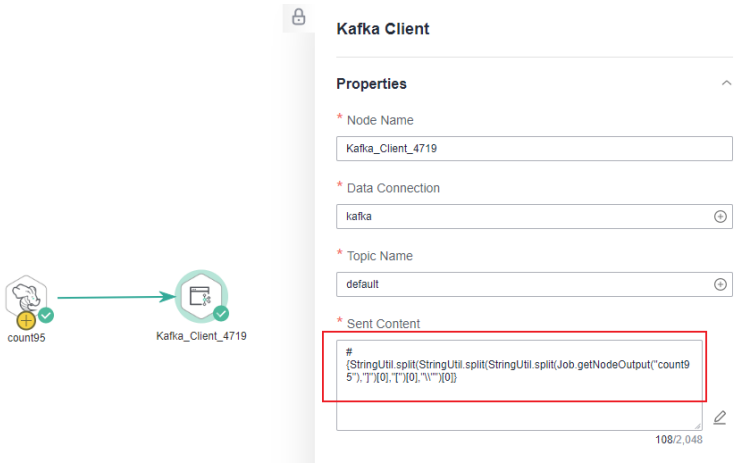
Step 6 Configuring parameters for an MRS Hive SQL node Select the **count95** script submitted in **Step 4** for **SQL script** and select a Hive connection and database.

Figure 2-18 Configuring parameters for an MRS Hive SQL node



Step 7 Configure parameters for the Kafka Client node. Set **Sent Content** to **#{StringUtil.split(StringUtil.split(StringUtil.split(Job.getNodeOutput("count95"),"]")[0],"[")[0],"\\"")[0]}** and select a Kafka connection and a topic name.

Figure 2-19 Configuring parameters for the Kafka Client node



Step 8 After the node configuration is complete, click **Test**. After the job test is successful, right-click the Kafka Client node to view its log. You can find that the two-dimensional array **[["2"]]** returned by the MRS Hive SQL node has been converted to **2**.

NOTE

You can set **Sent Content** of the Kafka Client node to **#{Job.getNodeOutput("count95")}** and run the job. Then you can view the log of the Kafka Client node to verify that the result returned by the MRS Hive SQL node is two-dimensional array **[["2"]]**.

Figure 2-20 Check the Kafka Client node logs.

----End

Obtaining Output Values Using the For Each Node

Scenario

You can use the For Each node and the EL expression #{Loop.current[0]} with a Loop embedded object to cyclically obtain the output values of the previous node.

In this example, the MRS Hive SQL node returns a two-dimensional array that contains multiple fields. You can use the For Each node which cyclically invokes the subjobs of the Kafka Client node and set **Sent Content** of the Kafka Client node to **#{Loop.current[]}** to obtain the output values of the MRS Hive SQL node.

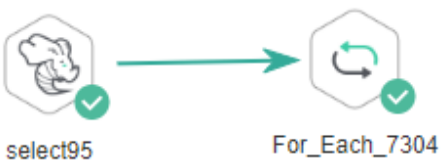
NOTE

To make it easy to view the obtained values, this example uses the Kafka Client node as the subjob node of the For Each node. In practice, you can select a subjob node type as needed. By using an EL expression with an embedded Loop object on the node, you can obtain the values returned by the previous node of the For Each node.

Orchestrate the main job shown in Figure 2-21. Key configurations of the For Each node are as follows:

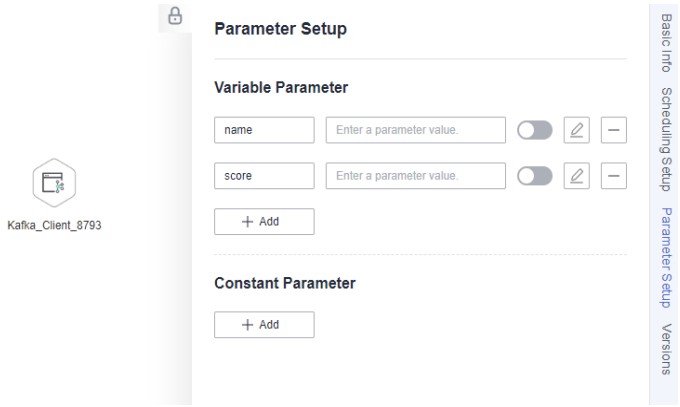
- **Dataset**: Enter the execution result of the select statement on the Hive SQL node. Use the **#{Job.getNodeOutput("select95")}** expression, where **select95** is the name of the previous node.
- **Subjob Parameter Name**: Enter the parameter name defined in the subjob. Transfer the parameter value defined in the main job to the subjob. Set the subjob parameter names to **name** and **score**, whose values are those in the first and second columns in the dataset, respectively. EL expressions **#{Loop.current[0]}** and **#{Loop.current[1]}** are used.

Figure 2-21 Example main job



For the subjobs selected for the For Each node, you must set their parameter names so that the main job can identify the parameter definitions.

Figure 2-22 Example subjob



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**.

Step 3 On the **Develop Job** page, create a data development subjob named **EL_test_slave**. Select a Kafka Client node, configure job parameters, and orchestrate the job shown in **Figure 2-22**.

Set the parameter name to **name** and **score**. This parameter is only used by the For Each node in the main job to identify subjob parameters. You do not need to set the parameter value.

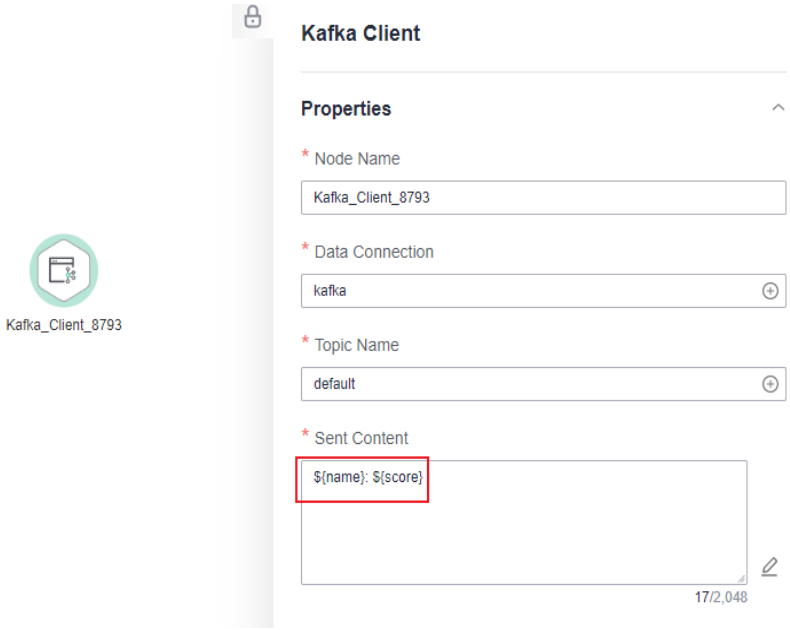
Step 4 Configure parameters for the Kafka Client node. Set **Sent Content** to **${name}: ${score}** and select a Kafka connection and a topic name.

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 2-23 Configuring parameters for the Kafka Client node



Step 5 Submit the subjob after the configuration is complete.

----End

Developing a Main Job

Step 1 Go to the **Develop Script** page.

Step 2 Create table **student_score**. Create a temporary Hive SQL script, select a Hive connection and database, paste the following SQL statement, and run the script. After the script is successfully executed, delete it.

```
CREATE TABLE `student_score` (`name` String COMMENT '', `score` INT COMMENT '');
INSERT INTO
   student_score
VALUES
   ('ZHAO', '90'),
   ('QIAN', '88'),
   ('SUN', '93'),
   ('LI', '94'),
   ('ZHOU', '85'),
   ('WU', '79'),
   ('ZHENG', '87'),
   ('WANG', '97'),
   ('FENG', '83'),
   ('CEHN', '99');
```

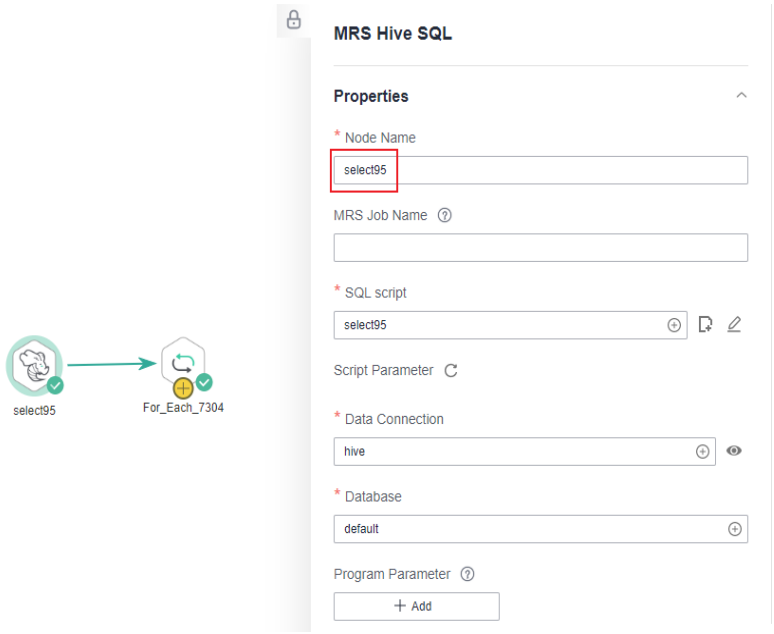
Step 3 Create the Hive SQL script to be invoked by the MRS Hive SQL node. Create a Hive SQL script named **select95**, select a Hive connection and database, paste the following SQL statement, and submit a version.

```
--Display the names and scores of students whose scores are higher than 95 in the student_score table.--
SELECT * FROM student_score WHERE score> "95" ;
```

Step 4 On the **Develop Job** page, create a data development job named **EL_test_master**. 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 2-21**.

Step 5 Configure parameters for the MRS Hive SQL node. Select the **select95** script submitted in **Step 3** for **SQL script** and select a Hive connection and database.

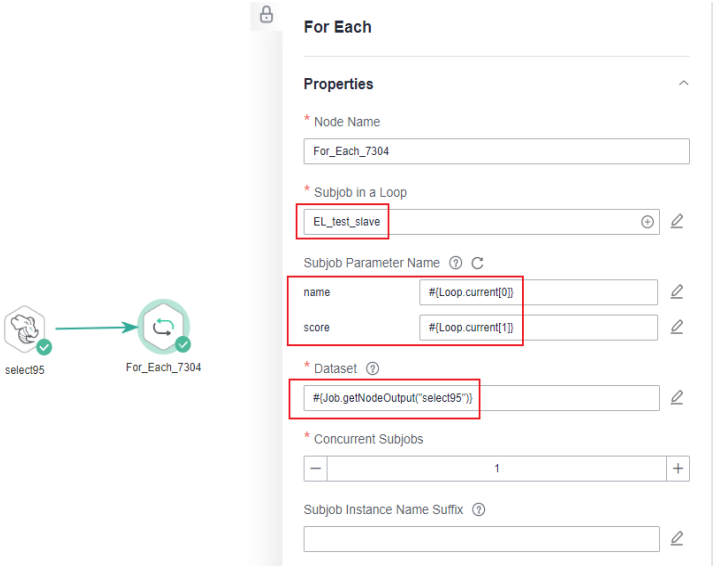
Figure 2-24 Configuring parameters for an MRS Hive SQL node



Step 6 Configure properties for the For Each node.

- **Subjob in a Loop**: Select **EL_test_slave**, the subjob that has been developed.
- **Dataset**: Enter the execution result of the select statement on the Hive SQL node. Use the **#{Job.getNodeOutput("select95")}** expression, where **select95** is the name of the previous node.
- **Subjob Parameter Name**: Enter the parameter name defined in the subjob. Transfer the parameter value defined in the main job to the subjob. Set the subjob parameter names to **name** and **score**, whose values are those in the first and second columns in the dataset, respectively. EL expressions **#{Loop.current[0]}** and **#{Loop.current[1]}** are used.

Figure 2-25 Configuring properties for the For Each node



Step 7 Save the job.

----End

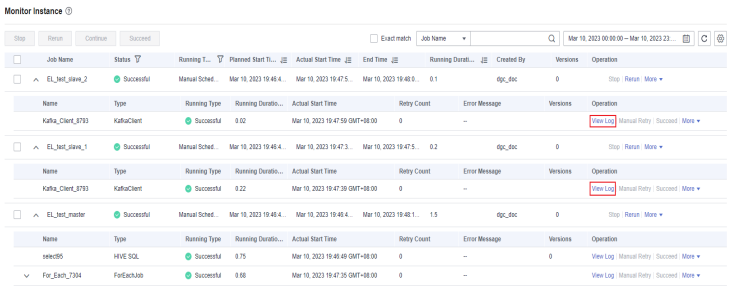
Testing the Main Job

Step 1 Click **Test** above the main job **EL_test_master** canvas to test the job. After the main job is executed, the subjob **EL_test_slave** is cyclically 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.

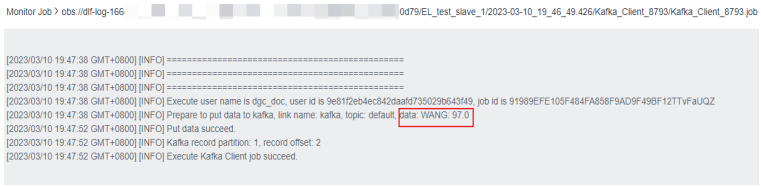
Step 3 After the job is executed, view the cyclic execution result of the subjob **EL_test_slave** on the **Monitor Instance** page.

Figure 2-26 Execution result of the subjob



Step 4 View the log of the cyclic execution of subjob **EL_test_slave**. The log shows that the output values of the previous node of the For Each node was obtained through the For Each node and the EL expression with a Loop embedded object.

Figure 2-27 Viewing the log



----End

2.5 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.

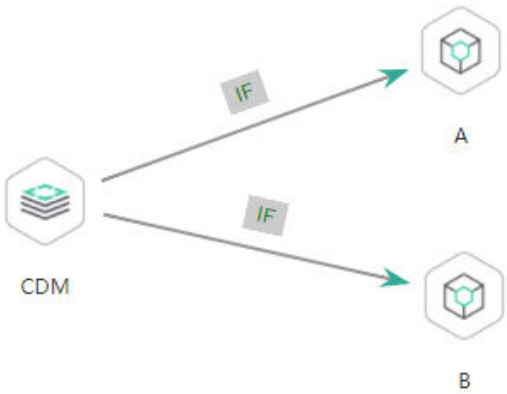
For details about how to use EL expressions, see **EL Expressions**.

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 2-28**.

Figure 2-28 Example job



Configuration Method

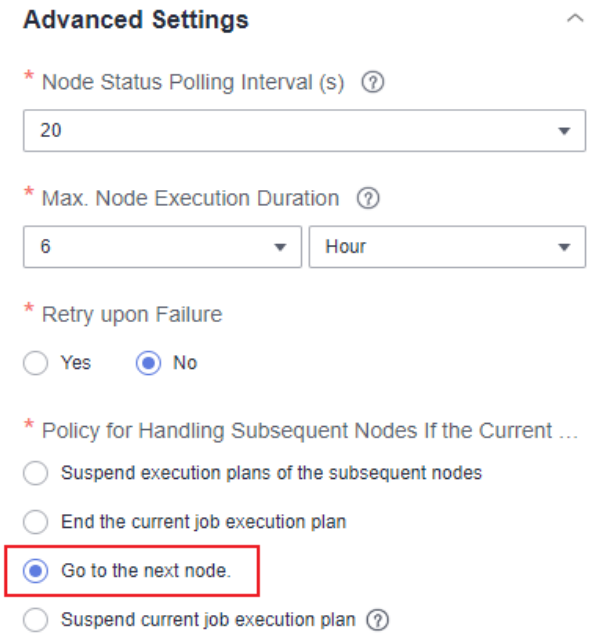
Step 1 Log in to the DataArts Studio console, locate the target DataArts Studio instance, and click **Access** on the instance card.

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 2-28**.

Set the **Failure Policy** for the CDM node to **Go to the next node**.

Figure 2-29 Configuring the failure policy for the CDM 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.

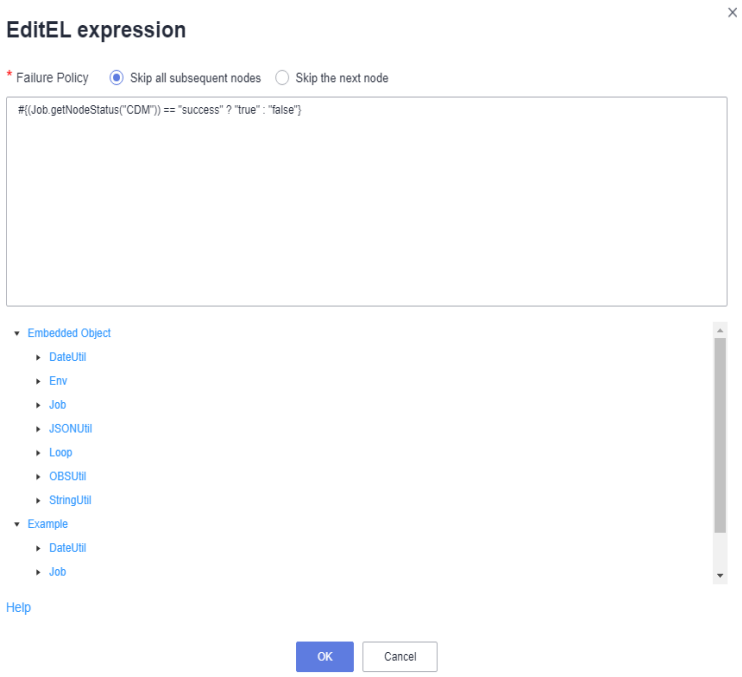
Each statement branch requires an IF statement. The IF statement is a ternary expression based on the EL expression syntax. If the result of the ternary expression is **true**, subsequent nodes will be connected. Otherwise, subsequent nodes will be skipped.

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.

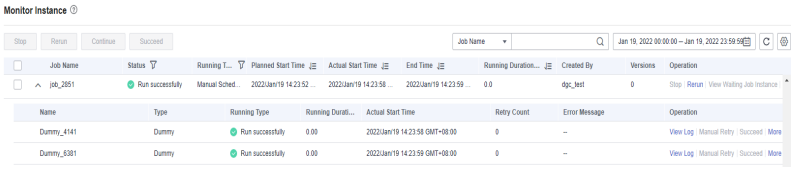
Figure 2-30 Configuring a failure policy



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 2-31 Job execution result



----End

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

Scenario Description

Scenario: Use the Hive SQLnode to collect statistics on the number of people whose score is higher than 85, transfer the execution result as a parameter to the next node, compare the result with the number of people who have passed the test, and determine the IF condition branch to be executed.

Analysis: The execution result of the select statement on the Hive SQL node is a two-dimensional array which contains a single field. Therefore, EL expression

#{Loop.dataArray[][]} or **#{Loop.current[]}** can be used to obtain the value in the two-dimensional array. Currently, only the For Each node supports loop expressions, so the Hive SQL node needs to be connected to a For Each node.

NOTE

In this scenario, the loop expression cannot be replaced by the StringUtil expression **#{StringUtil.split(StringUtil.split(StringUtil.split(Job.getNodeOutput("*Name of the previous node*"),"]")[0],"[")[0],"\\"")[0]}** because the StringUtil expression returns a string which cannot be compared with the standard data of the int type.

Figure 2-32 shows the job orchestration.

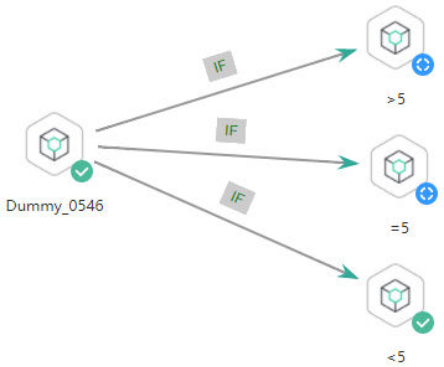
Figure 2-32 Example job

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]}** or **#{Loop.current[]}** is used. This example uses **{Loop.dataArray[0][0]}** as an example.

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 2-33 shows the job orchestration.

Figure 2-33 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 For Each. Drag four Dummy nodes and drop them on the canvas, click and hold to connect them, as shown in **Figure 2-33**.

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.

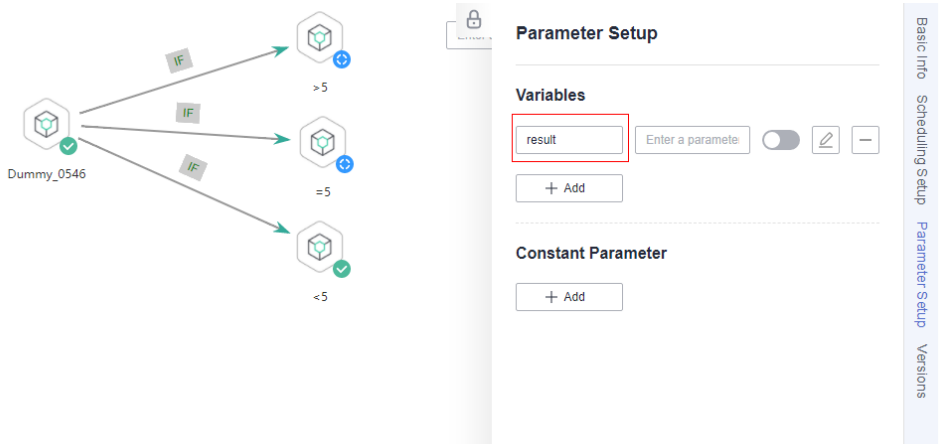
- For the **>5** branch, the IF statement expression is **#{${result} > 5 ? "true" : "false"}**.
- For the **=5** branch, the IF statement expression is **#{${result} == 5 ? "true" : "false"}**.
- For the **<5** branch, the IF statement expression is **#{${result} < 5 ? "true" : "false"}**.

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

> **NOTE**
>
> If an expression contains multiple conditions, you can use || to combine them conditions. The following is an example:
>
> #{(${result} >= 19 || ${result} <=9) ? "true" : "false"}

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 2-34 Configuring job parameters



Step 6 Save the job.

----End

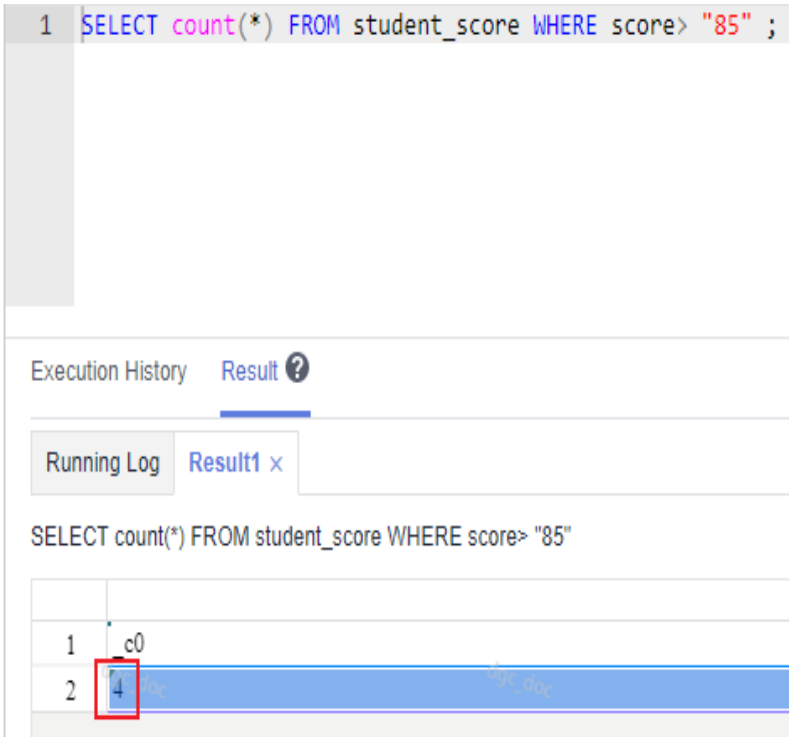
Developing a Job

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 2-32**.

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

```
--Obtain the number of people whose scores are higher than 85 from the student_score table.
SELECT count(*) FROM student_score WHERE score> "85" ;
```

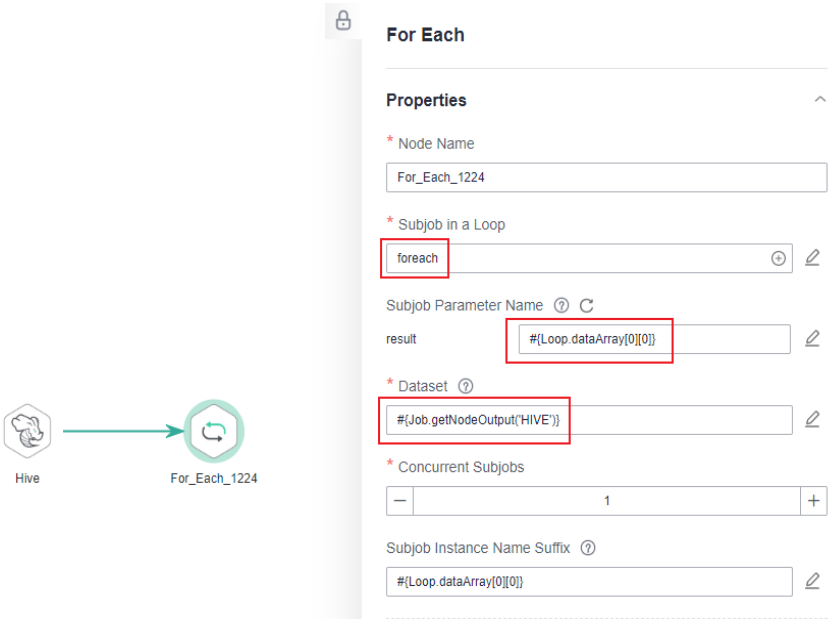
Figure 2-35 HIVE SQL script execution result



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 2-36 Properties of the For Each node



Step 4 Save the job.

----End

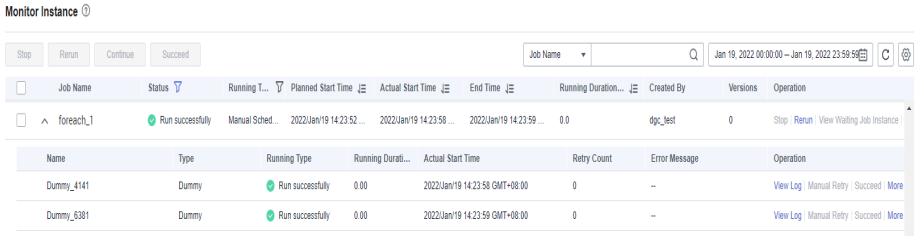
Testing the Main Job

Step 1 Click **Test** above the canvas to test the main 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 **4**. Therefore, the **>5** and **=5** branches are skipped, and the **<5** branch is successfully executed.

Figure 2-37 Execution result of the subjob



----End

Configuring the Policy for Executing a Node with Multiple IF Statements

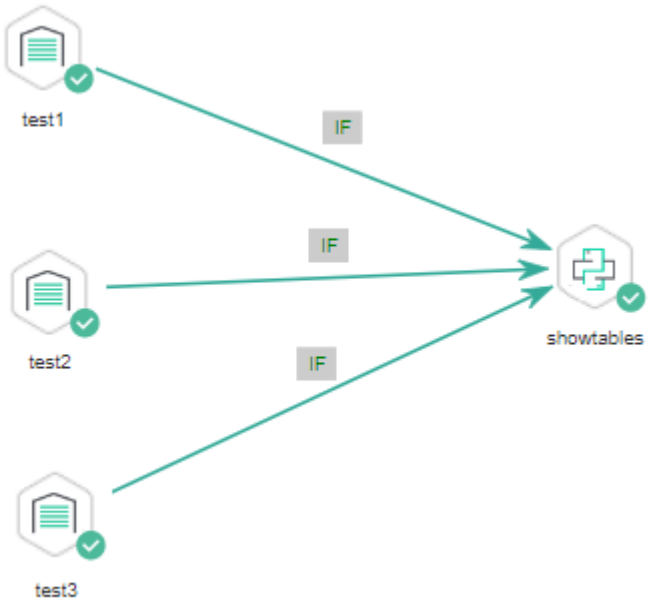
If the execution of a node depends on multiple IF statements, the policy for executing the node can be **AND** or **OR**.

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 2-38 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 2-38**.

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.

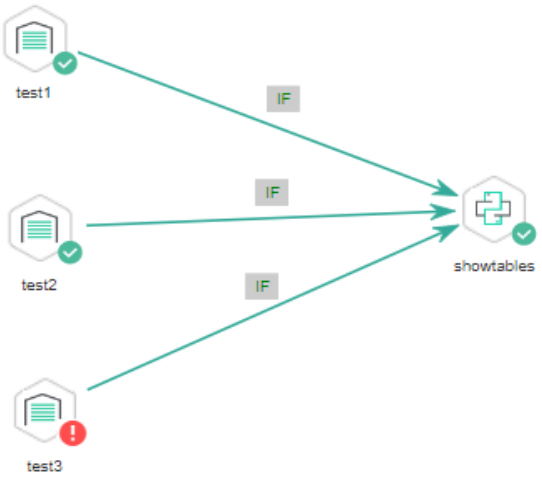
If **test1** is executed successfully, the corresponding IF statement is true.

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 2-39 How the job runs if Multi-IF Policy is OR

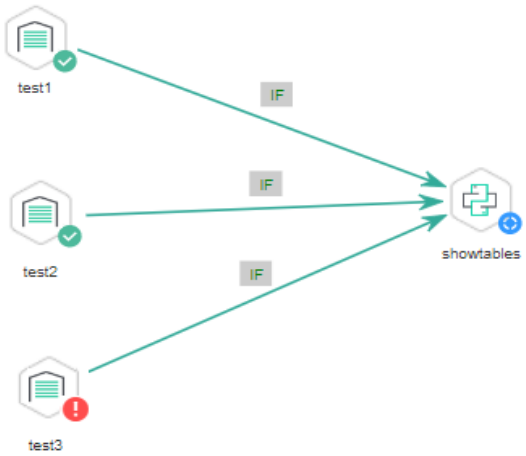


Logs

[**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 2-40 How the job runs if Multi-IF Policy is AND



Logs

[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

2.6 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 2-41**, the first Rest Client node invokes the API of MRS to query the cluster list. **Figure 2-42** 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.

  NOTE

  When setting **The response message body parses the transfer parameter**, ensure that the transferred parameter name (for example, **clusterId**) is unique among all node parameters of the job.

Figure 2-41 Rest Client job example 1

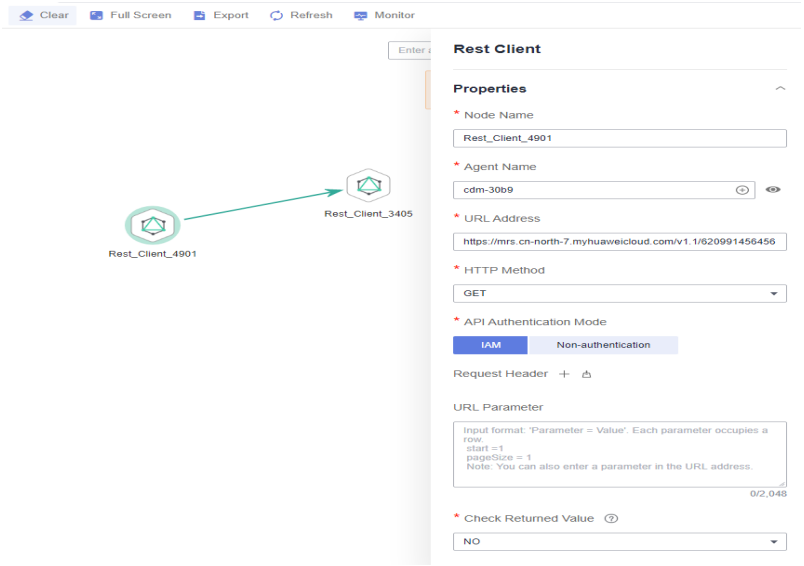


Figure 2-42 JSON message body


```
{
    "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":"cerxth-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
                },
```

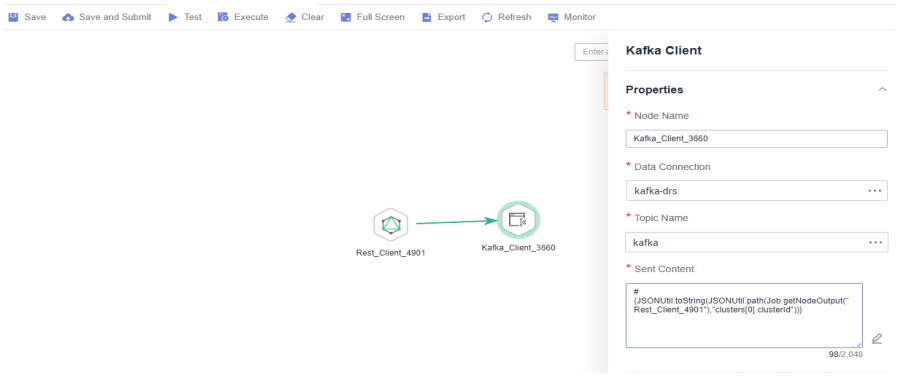

Obtaining the Return Value Using an EL Expression

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 2-43**, 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 2-43 Rest Client job example 2



2.7 Using For Each Nodes

Scenario

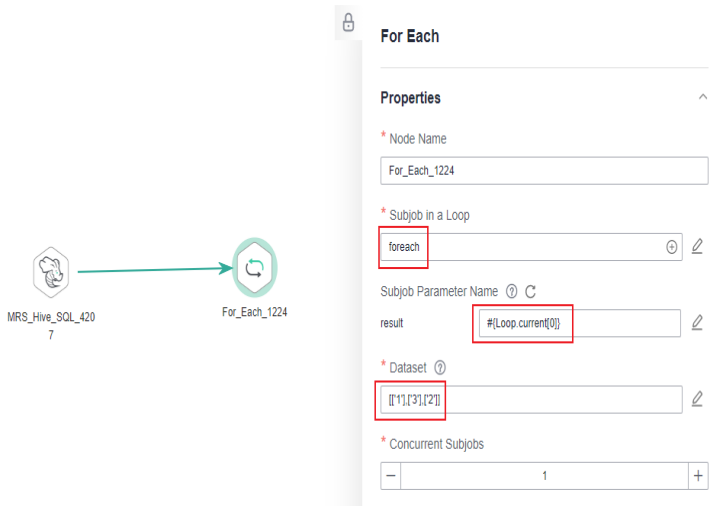
During job development, if some jobs have different parameters but the same processing logic, you can use For Each nodes to avoid repeated job development.

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 row of data in the dataset is traversed and transferred to the subjob.

Figure 2-44 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 2-44 For Each node



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 2-9 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 node 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**.

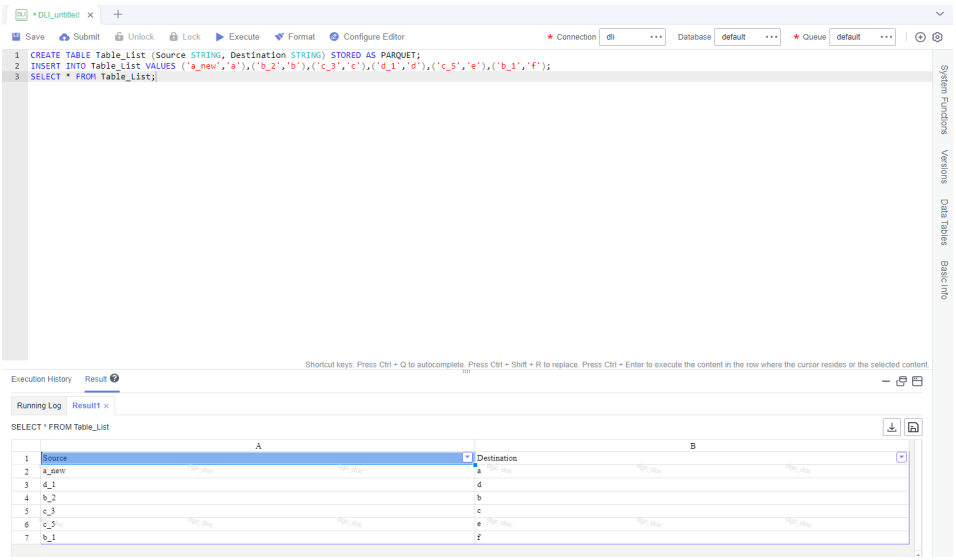
3. You can save the data in **Table 1** to a CSV file in the HDFS. Then use a Hive SQL statement to create a Hive foreign table, associate it with the CSV file, and use the query result of the Hive foreign table as the dataset. For details about how to create an MRS foreign table, see **Creating a Table**.

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 2-45 Data in the Table_List table



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 2-46**.

   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:

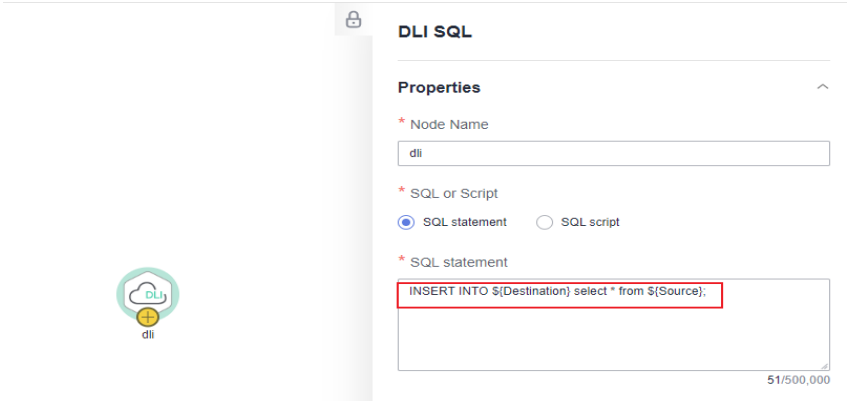
   ```
   INSERT INTO ${Destination} select * from ${Source};
   ```


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.

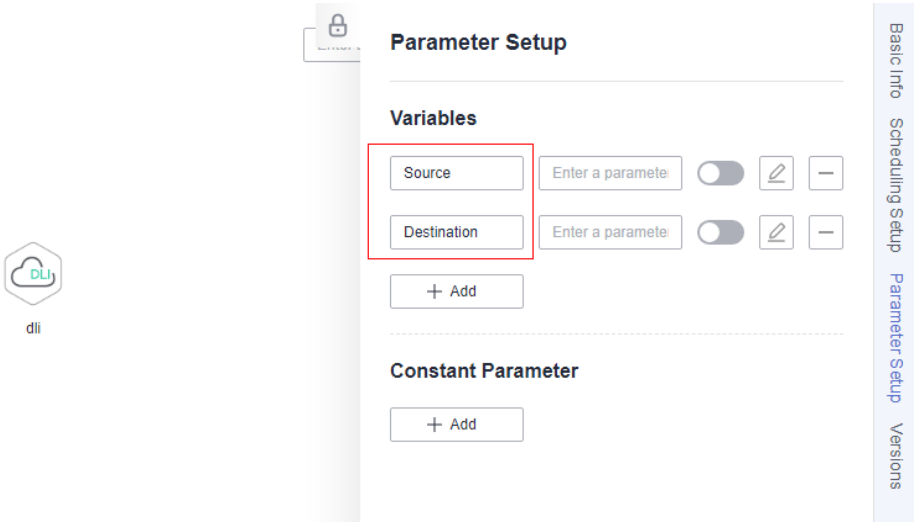
To obtain the parameter values passed from the parent job and the global variables configured for the workspace, you are advised to use the **${job_param_name}** expression.

Figure 2-46 Cyclically executing a subjob



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 2-47 Configuring subjob parameters



3. Save the job.

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

1. Access the DataArts Studio **DataArts Studio** page and choose **Develop Job**. Create a data development master job named **ForeachDemo_master**. Select

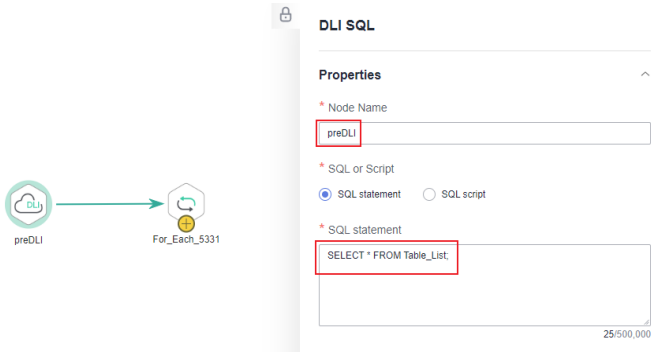
the DLI SQL and For Each nodes and click and drag to compile the job shown in **Figure 2-48**.

Figure 2-48 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.

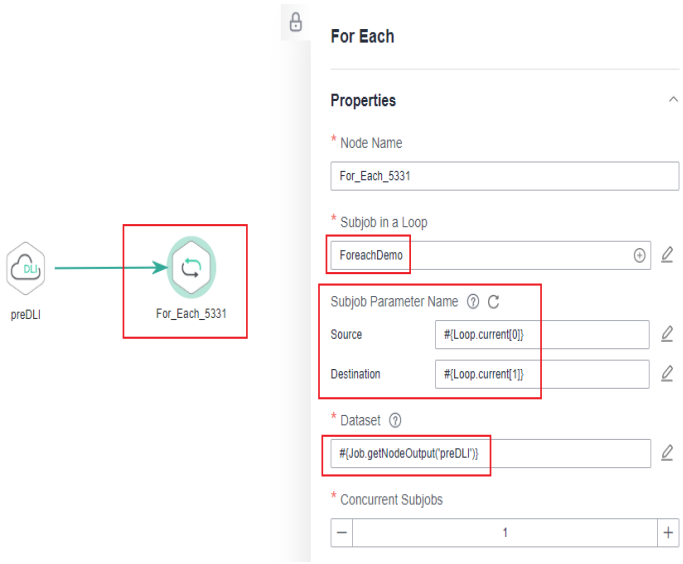
```
SELECT * FROM Table_List;
```

Figure 2-49 DLI SQL node configuration



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 2-50 Configuring properties for the For Each node

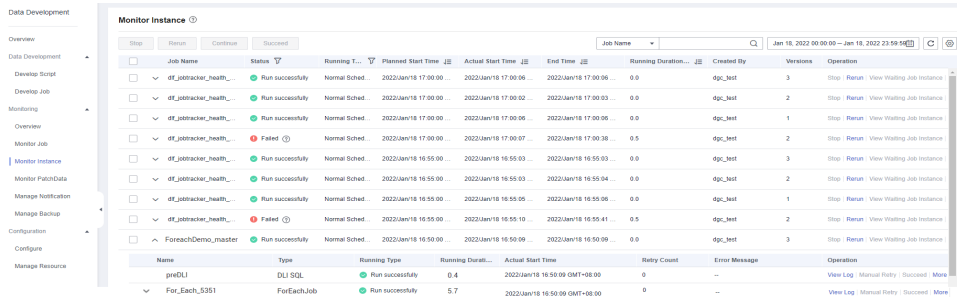


4. Save the job.

Step 5 Test the main job.

1. Click **Test** above the canvas to test the main 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 2-51 Viewing job instances

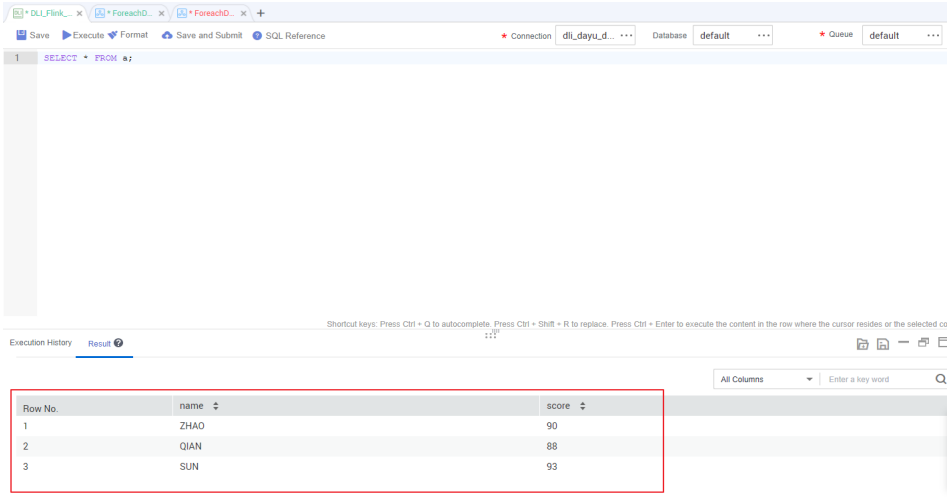


3. Check whether the data has been inserted into the six DLI destination tables. You can create a DLI SQL script on the **DataArts Factory** page and run the following commands to import data into the table. You can also run the following SQL commands in the SQL editor on the DLI console.

```
/* 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 2-52 Destination table data



----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**

2.8 Invoking DataArts Quality Operators Using DataArts Factory and Transferring Quality Parameters During Job Running

Parameters cannot be transferred to data quality jobs during the execution of SQL statements. You can invoke DataArts Quality operators through DataArts Factory to transfer parameters of data quality jobs to data quality operator jobs.

Scenario

Parameters need to be transferred to a data quality operator job, and the job can run properly.

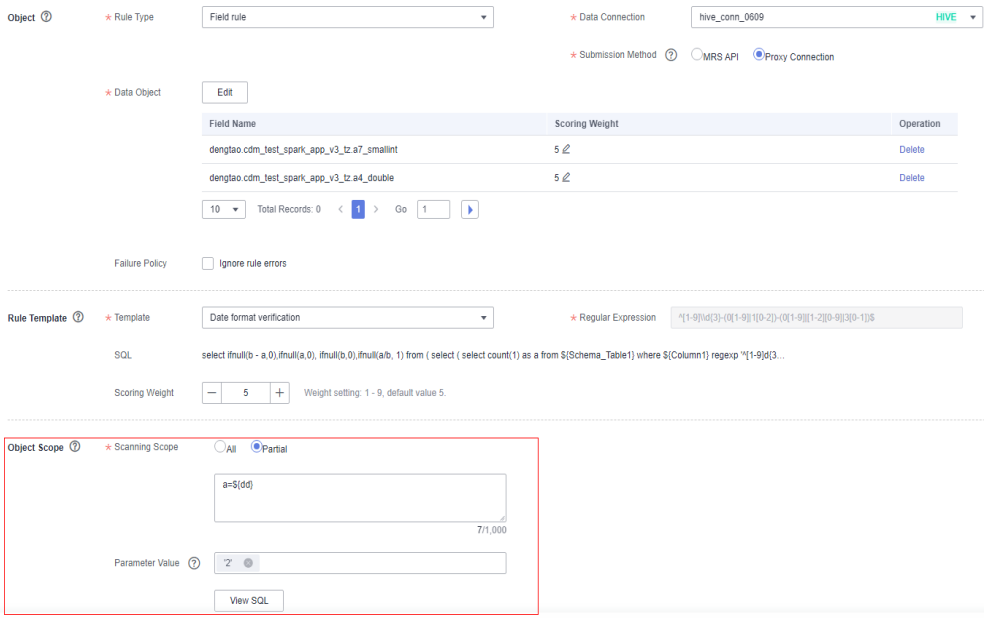
Procedure

Creating a Quality Job

1. On the DataArts Studio console, locate an instance and click **Access**. On the displayed page, locate a workspace and click **DataArts Quality**.

2. (Optional) In the left navigation pane, choose **Quality Monitoring** > **Quality Jobs** and create a directory. If a directory exists, you do not need to create one.
3. On the **Quality Jobs** page, click **Create**. On the displayed **Set Basic Info** page, configure basic information about the quality job.
4. Click **Next**. On the **Define Rule** page, configure rules for the quality job. In the **Object Scope** area, set **Scanning Scope** to **Partial** and configure parameters.

   Figure 2-53 Configuring parameters for the data quality job

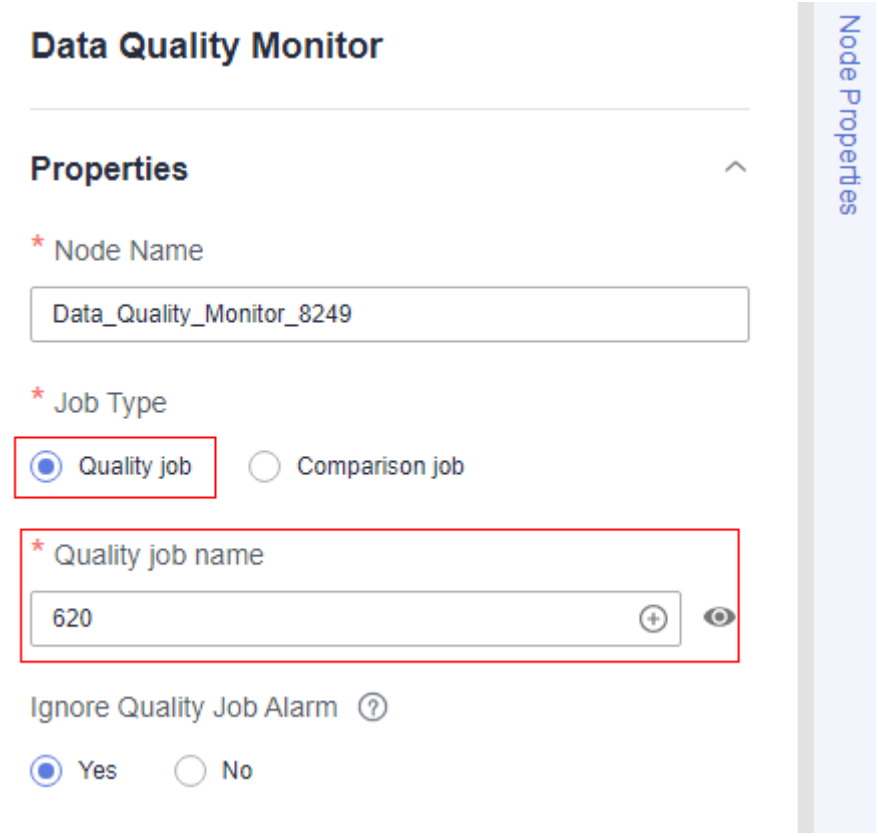


5. Click **Next** and configure alarm, subscription, and scheduling information. For details about how to configure a quality job, see **Creating Quality Jobs**.
6. Click **Submit**. The data quality job is configured successfully.

Configuring a Data Development Job

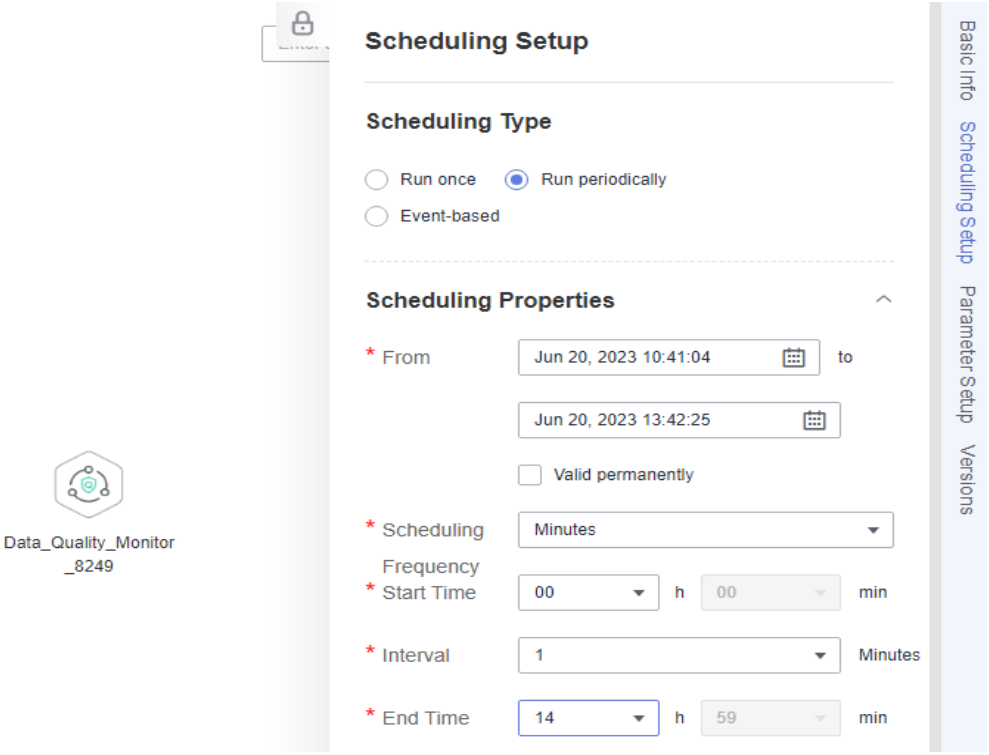
1. Log in to the DataArts Studio console. Locate an instance and click **Access**. On the displayed page, locate a workspace and click **DataArts Factory**.
2. In the left navigation pane of the DataArts Factory console, choose **Data Development** > **Develop Job**.
3. Create a batch processing pipeline job and open the job.
4. Drag a Data Quality Monitor operator to the canvas and configure node properties.

Figure 2-54 Configuring properties for the Data Quality Monitor node

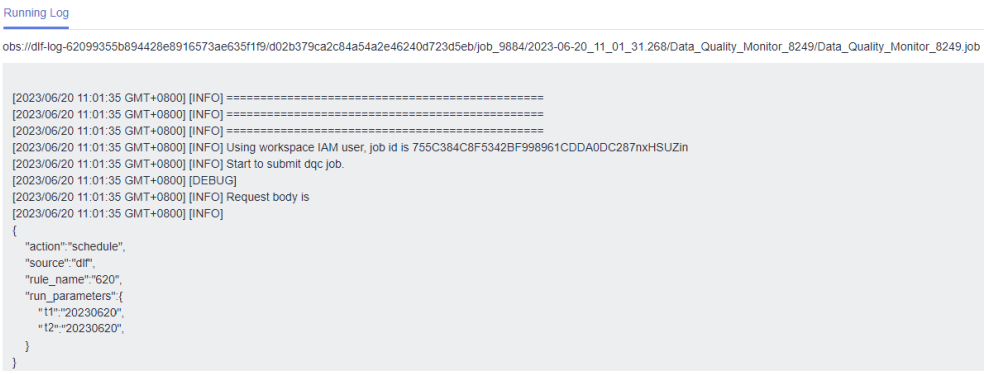


5. Configure the scheduling information.

Figure 2-55 Configuring the scheduling information



6. Submit the version and perform scheduling.
7. On the **Monitor Job** page, view the job run logs.



2.9 Scheduling Jobs Across Workspaces

Scenario

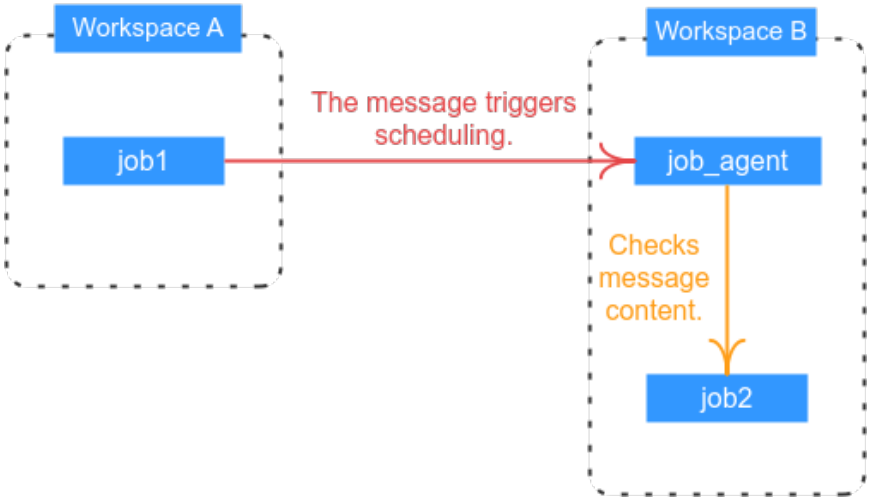
If you have assigned permissions based on workspaces, users in different workspaces can only perform operations on jobs in their own workspaces. However, if jobs in different workspaces depend on each other, you can schedule the jobs across workspaces by following the instructions in this section.

Solution

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

As shown in the following figure, after job1 in workspace A is complete, you can use DIS Client or Kafka Client to send a message to trigger job_agent for which event-driven scheduling has been configured. After job_agent is triggered by the message sent by DIS Client or Kafka Client, job_agent checks whether the message meets the expectation. If yes, job2 is triggered. If no, job2 is not triggered.

Figure 2-56 Scheduling solution



Prerequisites

Either of the following conditions must be met:

- A DIS stream is available.
- The MRS Kafka component is available, and MRS Kafka connections have been created in the Management Center of workspaces A and B.

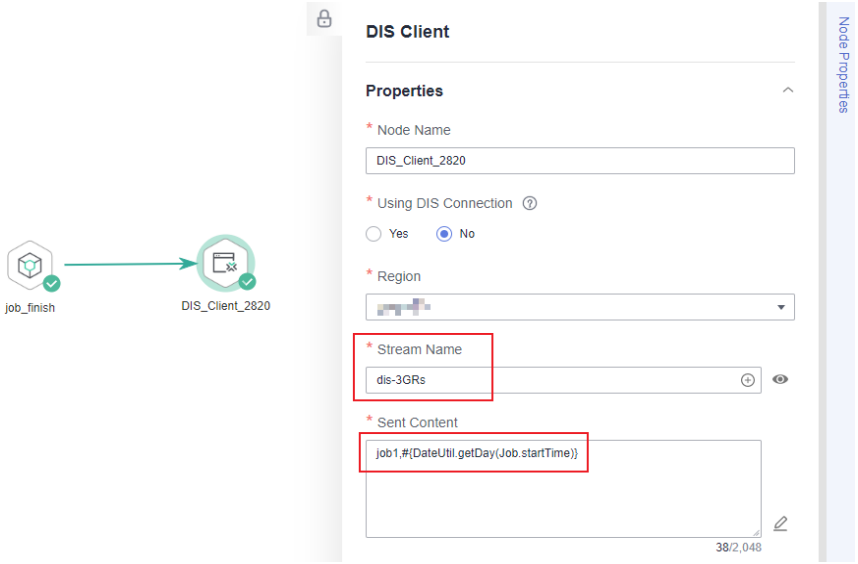
Configuration Method (DIS Client)

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 **job1**. Drag a Dummy and a DIS Client node and drop them on the canvas, and click and hold to connect the nodes, as shown in **Figure 2-57**.

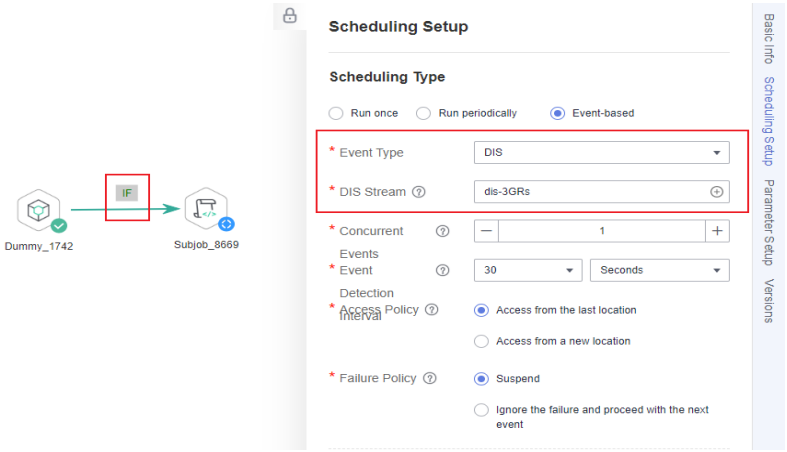
- The Dummy node does not perform any operation. In this example, the Dummy node is only used for demonstration. You can replace it with another node.
- The DIS Client node is used to send messages. You need to select the DIS region and stream, and set **Sent Content** to the EL expression **job1,#{DateUtil.getDay(Job.startTime)}**. After the job is executed, the DIS Client node sends a string message job1,***Job execution date***. For example, if job1 was executed on February 15, the message is **job1,15**.
- Retain the default values of other job parameters.

Figure 2-57 DIS Client node configuration for job1



Step 3 In workspace B, create a job named **job_agent**. Drag a Dummy node and a Subjob node and drop them on the canvas, and click and hold to connect the nodes, as shown in **Figure 2-58**.

Figure 2-58 Scheduling settings for job_agent



- The Dummy node does not perform any operation. In this example, the Dummy node is used to set the IF condition for the connection between the Dummy node and the Subjob node.
- The Subjob node is used to reference and execute job2 as a subjob. In practice, you can reference an existing job or use another job node to replace the Subjob node.
- Set **Scheduling Type** to **Event-based**, and set **DIS Stream** to the DIS stream selected for the DIS Client node of job1 in workspace A. The stream is used to trigger job execution through DIS messages.

- Set the IF condition to check whether the message sent by the DIS Client node meets the expectation. If yes, the Subjob node will be executed. Otherwise, the Subjob node will be skipped.

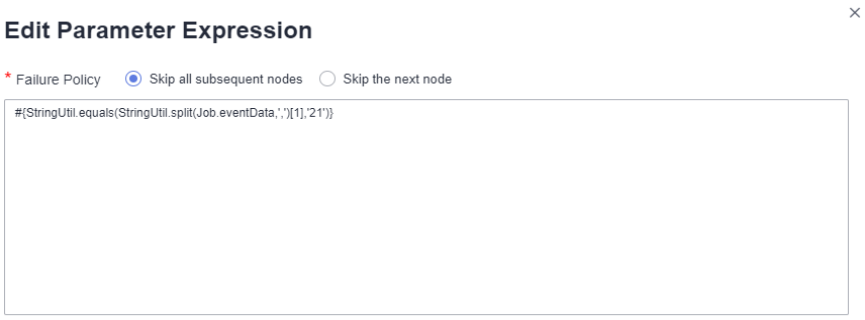
  Right-click the connection line and select **Set Condition**. In the **Edit Parameter Expression** dialog box, enter the IF condition in the text box and retain the default failure policy. The IF condition is a ternary expression based on the EL expression syntax. The node following the connection line 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],'21')}
  ```

  This IF condition indicates that subsequent job nodes are executed only if **21** (21st of each month) follows the comma in the message obtained from the DIS stream.

  NOTE

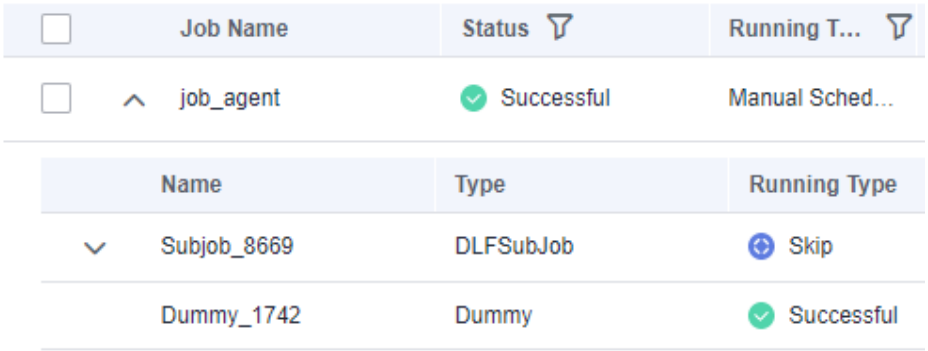
  If you want to match multiple messages, you can add multiple Dummy nodes, set the IF condition for the connection between each Dummy node and the Subjob node, and set **Multi-IF Policy** to **OR** in the configuration of DataArts Factory.



Step 4 Run the job_agent job when job1 in workspace A is not running. Then go to the **Monitor Instance** page and check whether the execution result meets the expectation.

Because job1 is not running, no message is sent, and the Subjob node in the job_agent job is skipped, indicating that the IF condition takes effect.

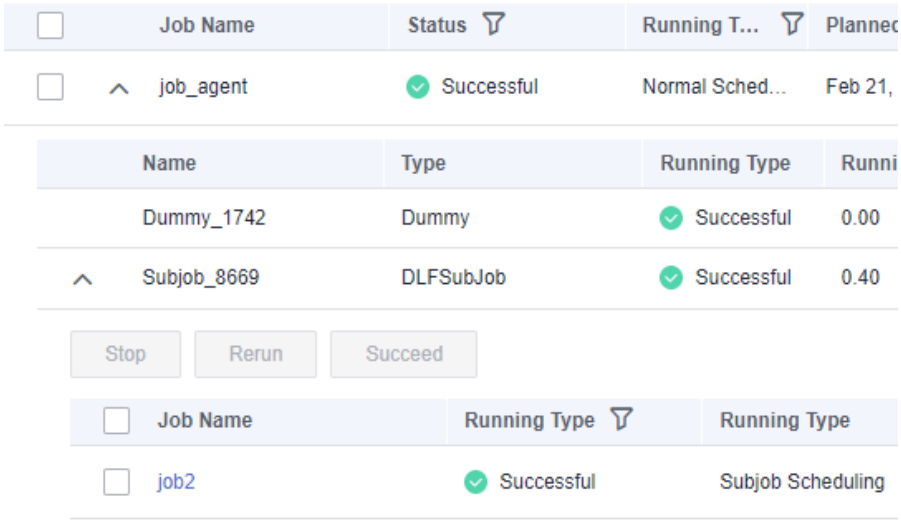
Figure 2-59 Subjob node skipped



Step 5 Start scheduling the job_agent job. Then run job1 in workspace A. After job1 is successfully executed, go to the **Monitor Instance** page of workspace B to check whether the job execution result meets the expectation.

- The job_agent job is triggered.
- If the current date matches the date in the IF condition, the Subjob node in the job_agent job is successfully executed, and job2 is also successfully executed. Otherwise, the Subjob node is skipped.

Figure 2-60 Subjob node executed successfully



----End

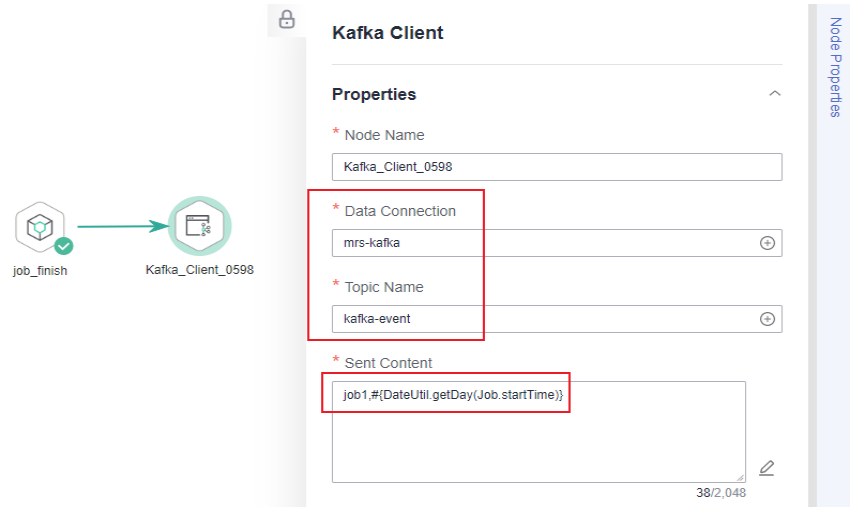
Configuration Method (Kafka Client)

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 **job1**. 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 2-61**.

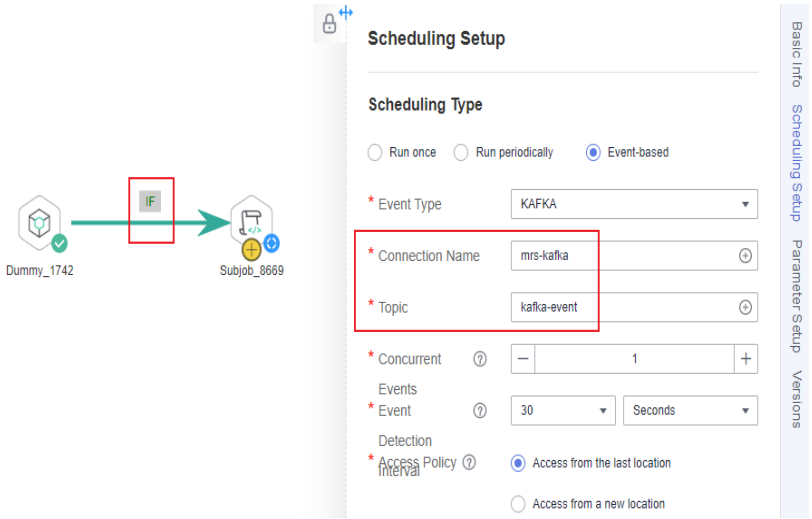
- The Dummy node does not perform any operation. In this example, the Dummy node is only used for demonstration. You can replace it with another node.
- The Kafka Client node is used to send messages. You need to select a Kafka connection and a topic name, and set **Sent Content** to the EL expression **job1,#{DateUtil.getDay(Job.startTime)}**. After the job is executed, the Kafka Client node sends a string message job1,***Job execution date***. For example, if job1 was executed on February 15, the message is **job1,15**.
- Retain the default values of other job parameters.

Figure 2-61 Kafka Client node configuration for job1



Step 3 In workspace B, create a job named **job_agent**. Drag a Dummy node and a Subjob node and drop them on the canvas, and click and hold to connect the nodes, as shown in **Figure 2-62**.

Figure 2-62 Scheduling settings for job_agent



- The Dummy node does not perform any operation. In this example, the Dummy node is used to set the IF condition for the connection between the Dummy node and the Subjob node.
- The Subjob node is used to reference and execute job2 as a subjob. In practice, you can reference an existing job or use another job node to replace the Subjob node.
- Set **Scheduling Type** to **Event-based**, and set **Connection Name** and **Topic** to the Kafka connection and topic in workspace B, which correspond to the Kafka connection and topic selected for the Kafka Client node of job1 in workspace A. The connection and topic are used to trigger job execution through Kafka messages.

- Set the IF condition to check whether the message sent by the Kafka Client node meets the expectation. If yes, the Subjob node will be executed. Otherwise, the Subjob node will be skipped.

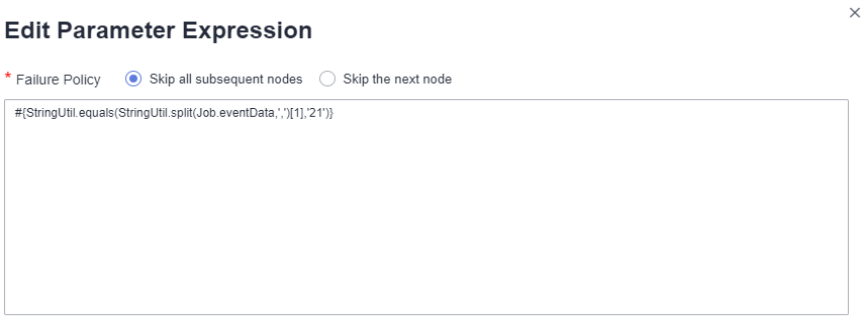
  Right-click the connection line and select **Set Condition**. In the **Edit Parameter Expression** dialog box, enter the IF condition in the text box and retain the default failure policy.. The IF condition is a ternary expression based on the EL expression syntax. The node following the connection line 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],'21')}
  ```

  This IF condition indicates that subsequent job nodes are executed only if **21** (21st of each month) follows the comma in the message obtained from the Kafka stream.

  NOTE

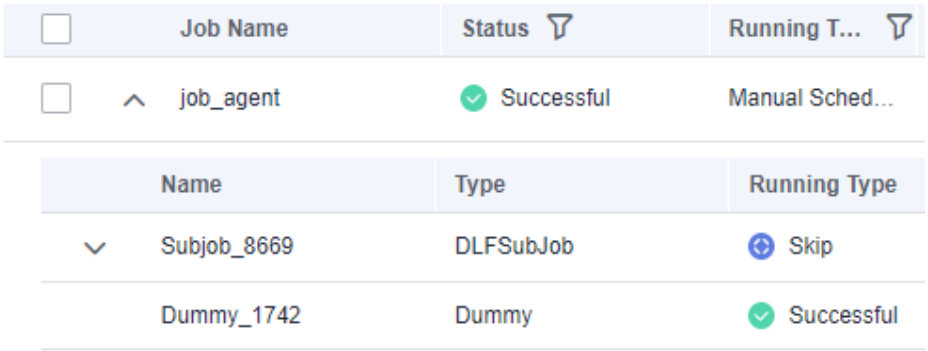
  If you want to match multiple messages, you can add multiple Dummy nodes, set the IF condition for the connection between each Dummy node and the Subjob node, and set **Multi-IF Policy** to **OR** in the configuration of DataArts Factory.



Step 4 Run the job_agent job when job1 in workspace A is not running. Then go to the **Monitor Instance** page and check whether the execution result meets the expectation.

Because job1 is not running, no message is sent, and the Subjob node in the job_agent job is skipped, indicating that the IF condition takes effect.

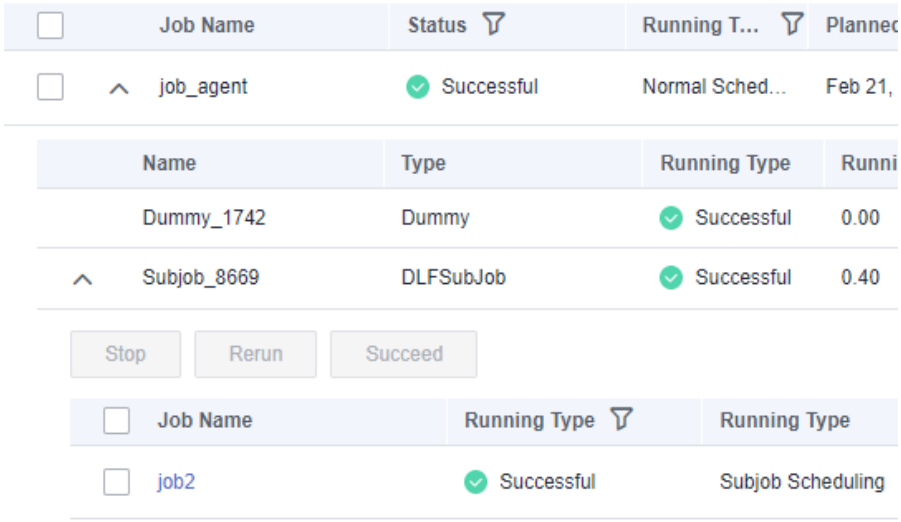
Figure 2-63 Subjob node skipped



Step 5 Start scheduling the job_agent job. Then run job1 in workspace A. After job1 is successfully executed, go to the **Monitor Instance** page of workspace B to check whether the job execution result meets the expectation.

- The job_agent job is triggered.
- If the current date matches the date in the IF condition, the Subjob node in the job_agent job is successfully executed, and job2 is also successfully executed. Otherwise, the Subjob node is skipped.

Figure 2-64 Subjob node executed successfully



----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.

Resources that can be migrated include those in DataArts DataService and DataArts Catalog as well as the data connections in Management Center.

Constraints

- Resources can be imported from OBS or a local path.
- 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.
- Only the enterprise edition supports the export of data catalogs (categories, tags, and collection tasks). The expert edition does not support this function.
- The size of the file to be imported from an OBS bucket or local path cannot exceed 10 MB.

Exporting Resources from the Old Workspace

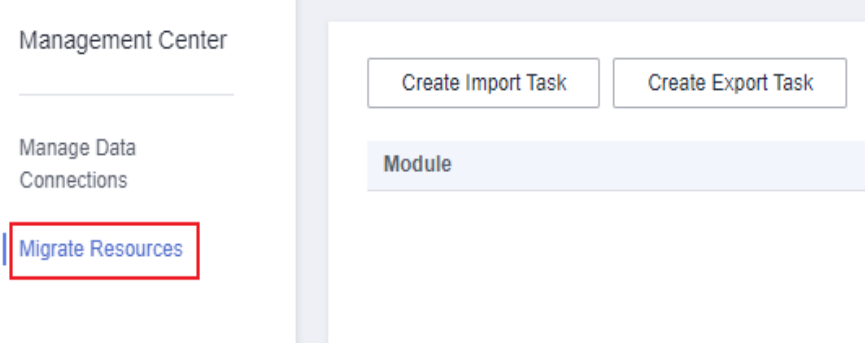
Log in to the console, access the **Management Center** module of the old workspace, and perform the following operations to export resources:

Step 1 Log in to the DataArts Studio console by following the instructions in **Accessing the DataArts Studio Instance Console**.

Step 2 On the DataArts Studio console, locate a workspace and click **Management Center**.

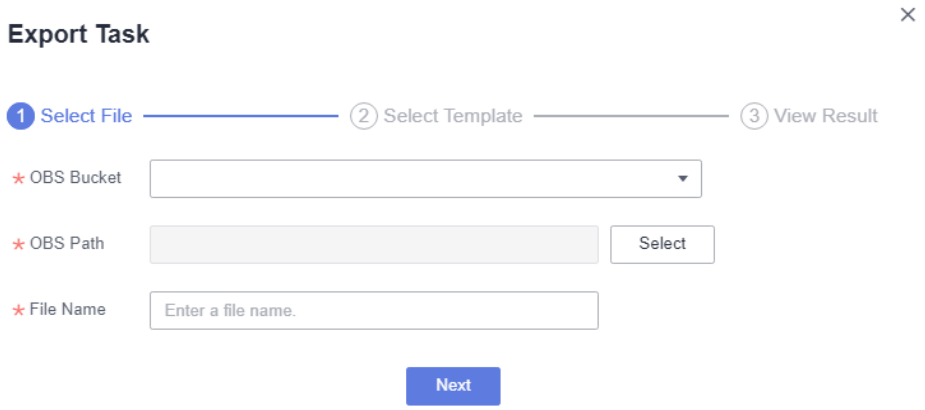
Step 3 In the navigation pane, choose **Migrate Resources**.

Figure 3-1 Migrating Resources



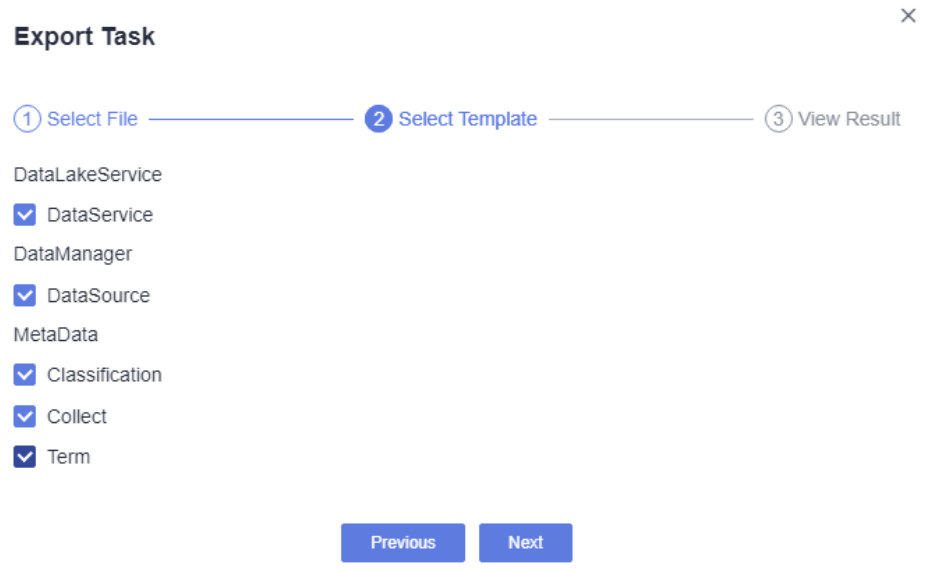
Step 4 Click **Create Export Task** to configure the file name and the OBS path for saving the file.

Figure 3-2 Export Task



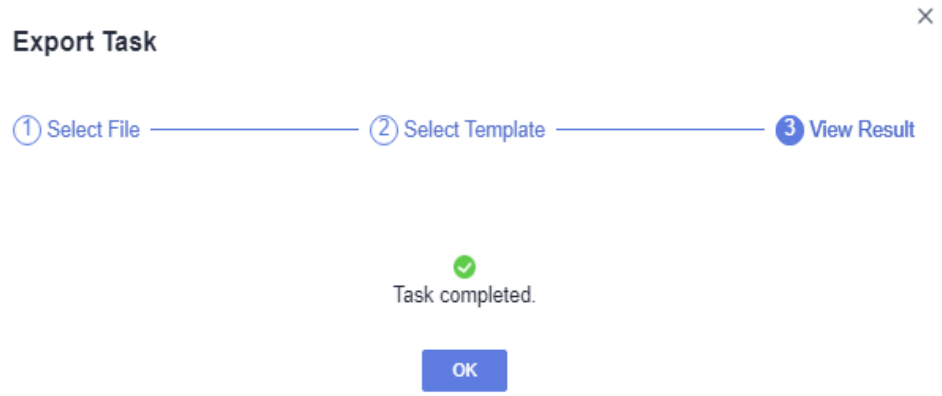
Step 5 Click **Next** and select the resources to export.

Figure 3-3 Selecting the resource to export



Step 6 Click **Next** and wait until the export is complete. The resource package is exported to the OBS path you have set.

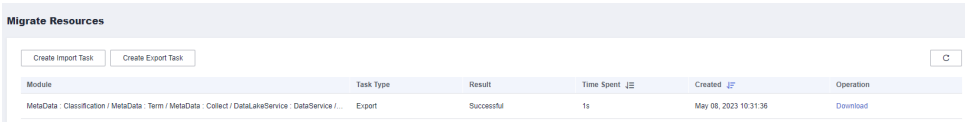
Figure 3-4 Export completed



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

Step 7 After the export is complete, you can click **Download** in the row of the corresponding migration task to download the exported resource package.

Figure 3-5 Downloading the exported result



----End

Importing Resources to Another Workspace

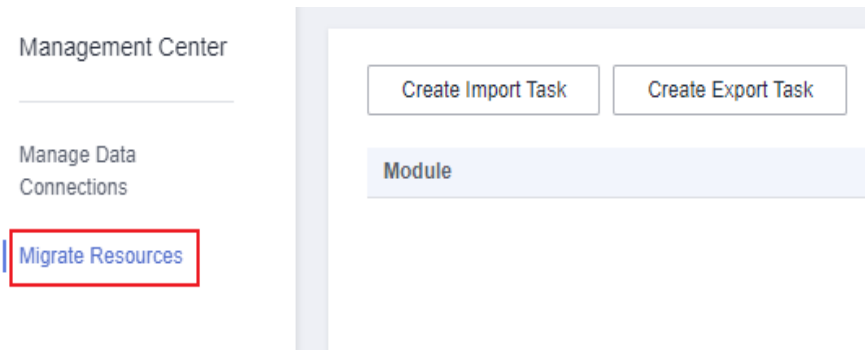
Log in to the console, access the **Management Center** module of the new workspace, and perform the following operations to import resources:

Step 1 Log in to the DataArts Studio console by following the instructions in **Accessing the DataArts Studio Instance Console**.

Step 2 On the DataArts Studio console, locate a workspace and click **Management Center**.

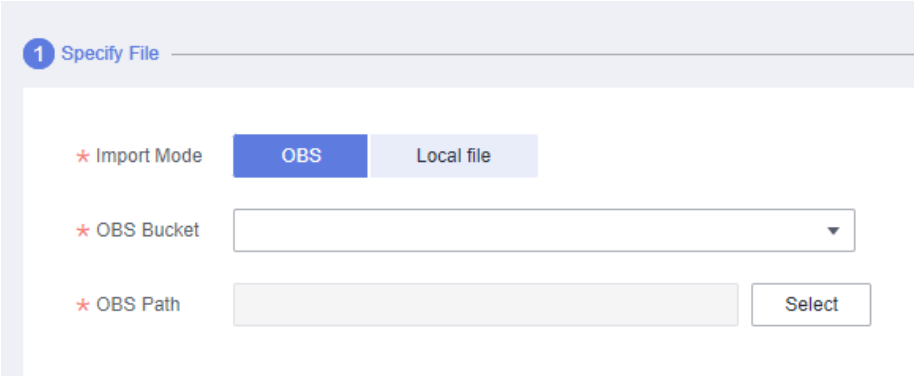
Step 3 In the navigation pane, choose **Migrate Resources**.

Figure 3-6 Migrating Resources



Step 4 Click **Create Import Task**. On the displayed page, select an import mode and set the OBS bucket and path or local path that stores resources. The resource to be imported must be a .zip file exported from the console.

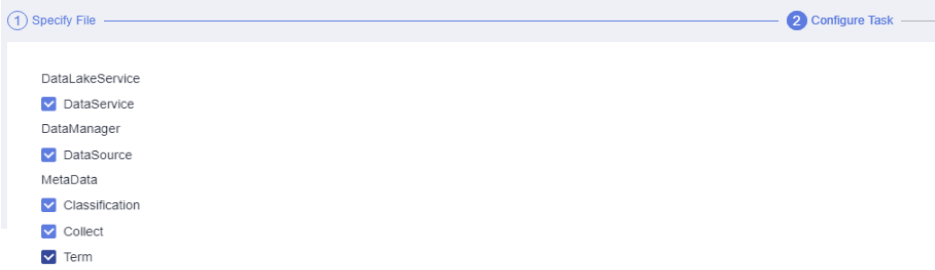
Figure 3-7 Configuring the path that stores the resources to be imported



Step 5 Click **Create Import Task** and upload a .zip resource file that you have exported.

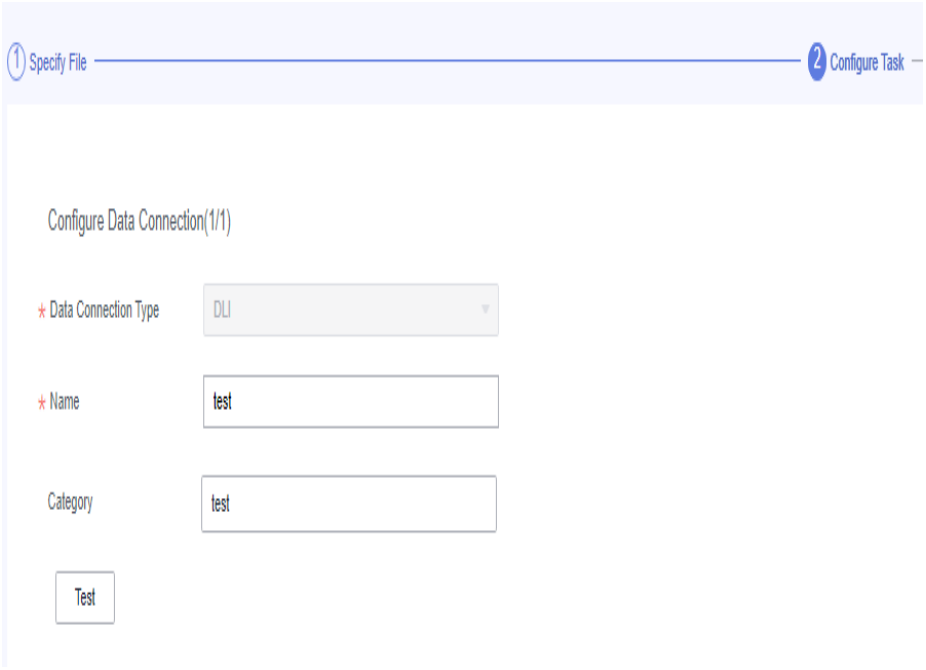
Step 6 Click **Next** and select the resources to import.

Figure 3-8 Selecting the resource to import



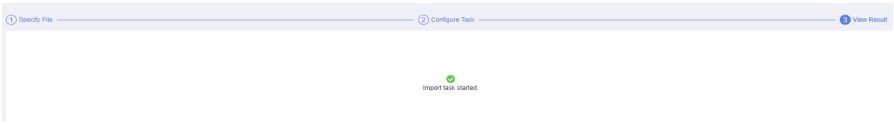
Step 7 If you select **DataSource**, click **Next** to configure a data connection.

Figure 3-9 Configuring a data connection



Step 8 Click **Next** and wait until the import task is delivered. When the import task is delivered successfully, the system displays message "Import task started."

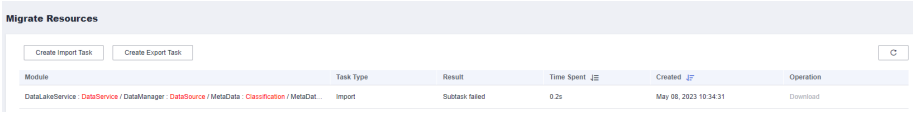
Figure 3-10 Import task started



Step 9 Click **OK**. You can view the import result in the resource migration task list.

Subtasks that fail are marked in red. You can click their names to view the failure causes.

Figure 3-11 Viewing the import result



----End

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.

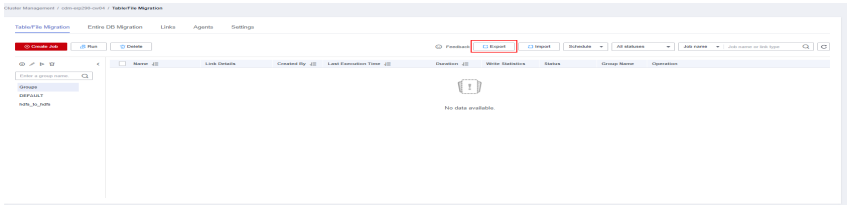
Exporting Jobs and Links from the Old Workspace

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.

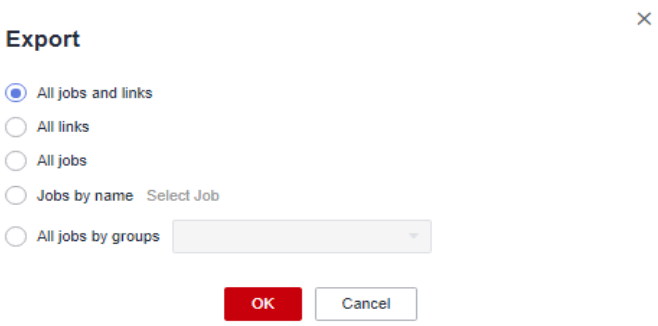
Step 2 Click **Export** above the job list.

Figure 3-12 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 3-13 Exporting all jobs and links



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

----End

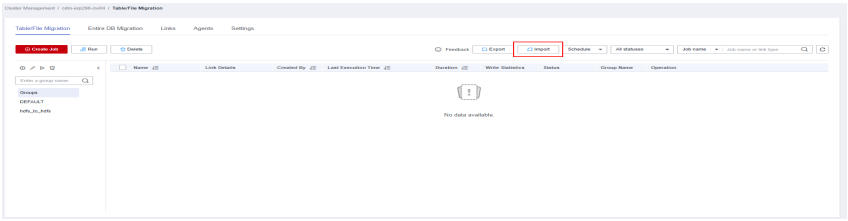
Importing Jobs and Links to the New Workspace

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.

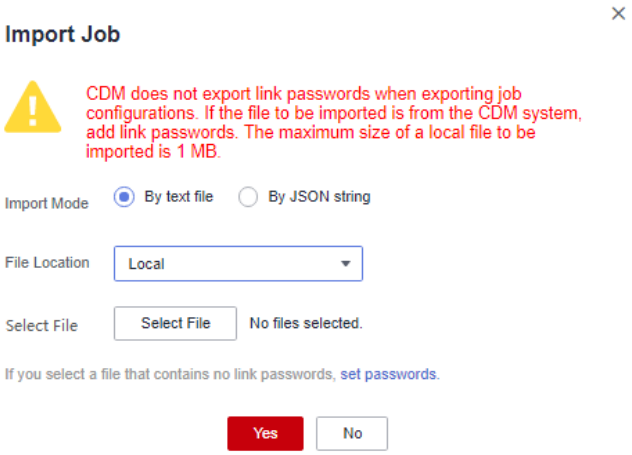
Step 2 Click **Import** above the job list to import the JSON file.

Figure 3-14 Importing jobs and links



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

Figure 3-15 Selecting a JSON file

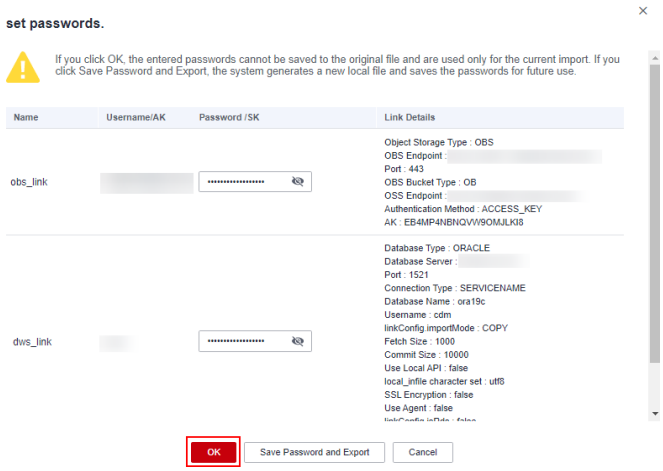


Step 4 After the JSON file is uploaded, click **set passwords** to set passwords or SKs for data links.

Figure 3-16 Setting passwords

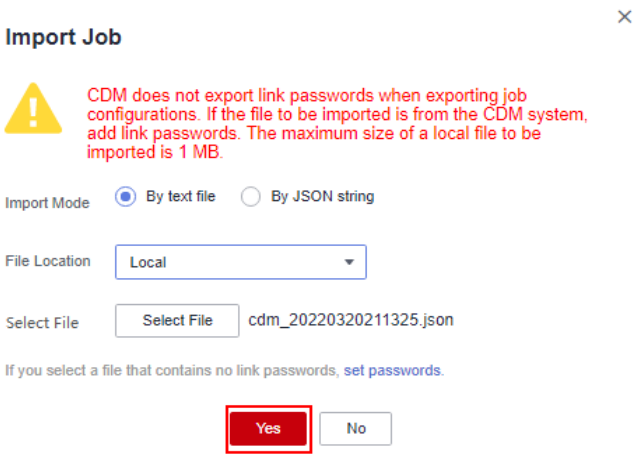
Step 5 In the displayed dialog box, enter the password or SK of each data link and click **OK** to return to the **Import Job** dialog box.

Figure 3-17 Setting passwords



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

Figure 3-18 Starting the import



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.
- The maximum size of a file to be imported is 4 MB. A maximum of 3,000 metrics can be imported. A maximum of 500 tables can be exported at a time.

Exporting Table Data from the Old Workspace

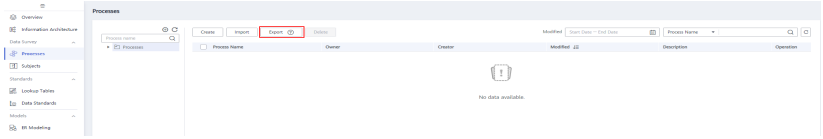
Log in to the console, access the DataArts Architecture module of the old workspace, and perform the following operations to export **processes**, **subjects**, **lookup tables**, **data standards**, **ER modeling tables/entities**, **dimensions/fact tables**, **business metrics**, **technical metrics**, 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 3-19 Exporting processes



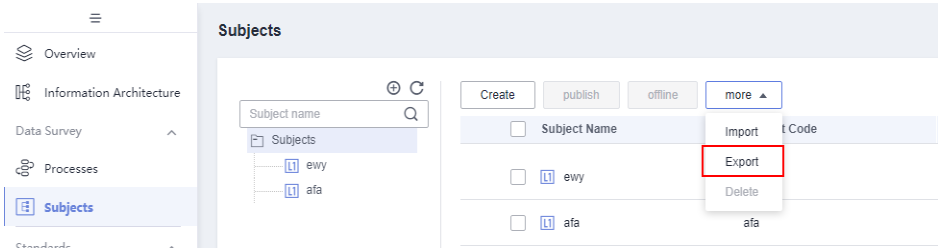
----End

Exporting subjects

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

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

Figure 3-20 Exporting subjects



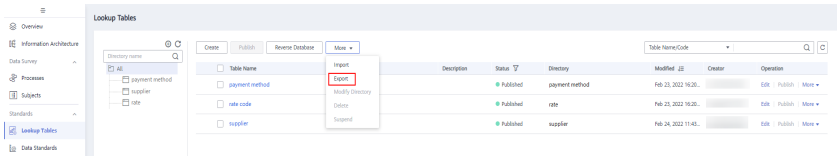
----End

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 3-21 Exporting lookup tables



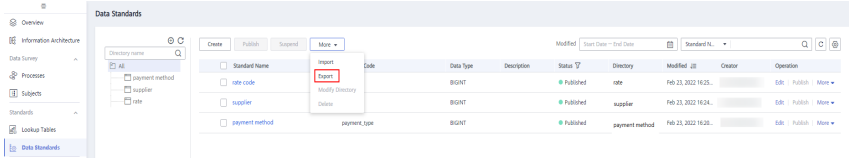
----End

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 3-22 Exporting data standards



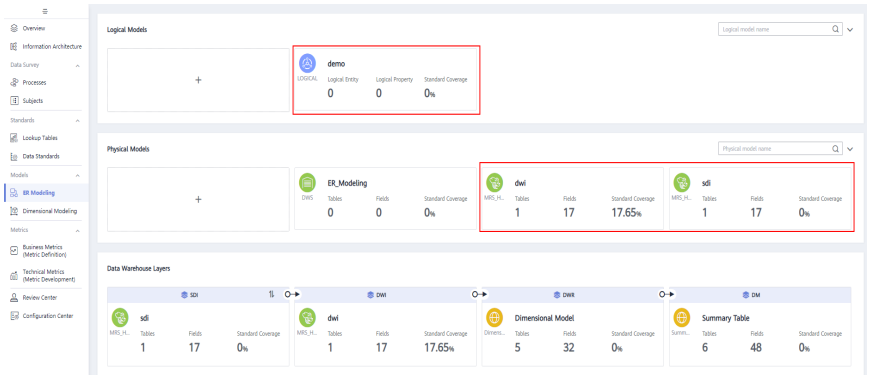
----End

Exporting ER modeling tables/entities

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

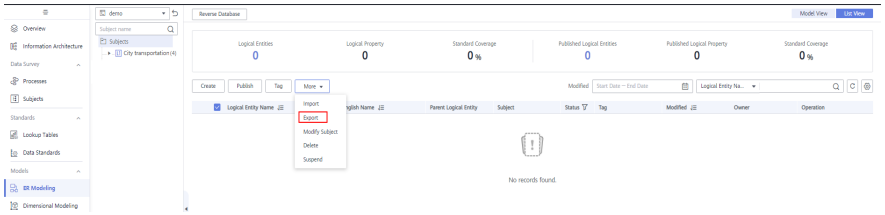
Step 2 Access a logical or physical model and export tables/entities in the model. This section uses logical model **demo** as an example.

Figure 3-23 Accessing the model



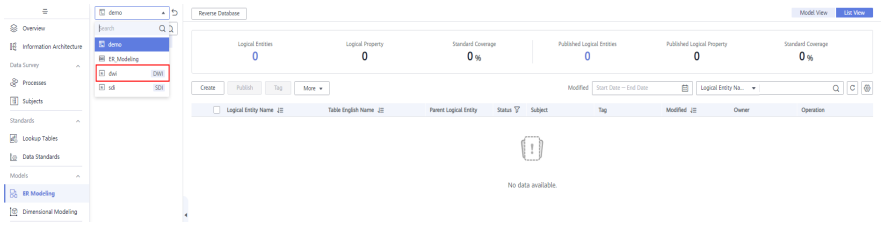
Step 3 Select the tables/entities to export, click **More** above the list, and select **Export** from the drop-down list box (you are advised to select **Table** for **Export**). After the export is successful, you can obtain the exported .xlsx file.

Figure 3-24 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 3-25 Exporting tables/entities of other models



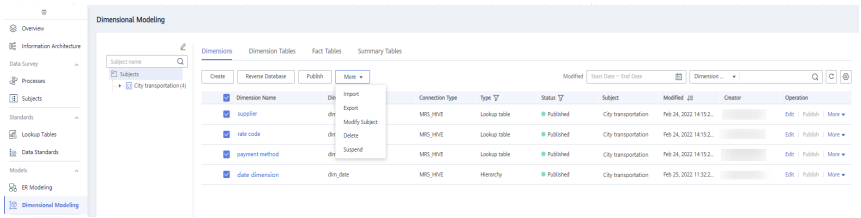
----End

Exporting dimensions/fact tables

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

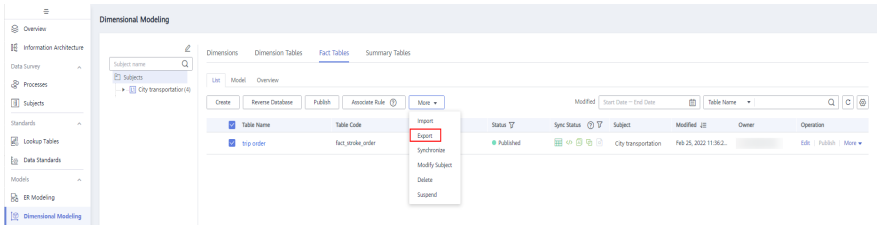
Step 2 On the displayed **Dimensions** page, select the dimensions to export, click **More** above the list, and select **Export** from the drop-down list box. After the export is successful, you can obtain the exported .xlsx file.

Figure 3-26 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 3-27 Exporting fact tables



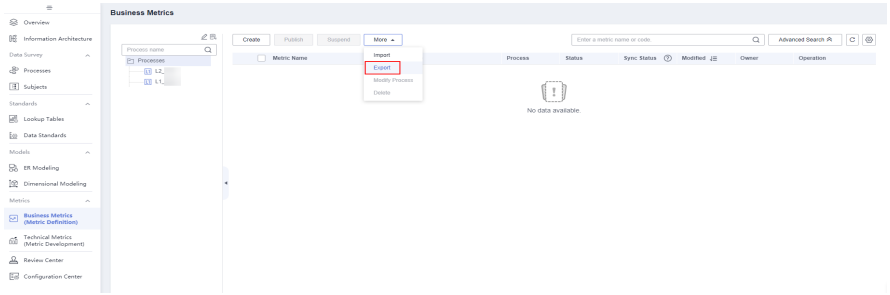
----End

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 3-28 Exporting business metrics



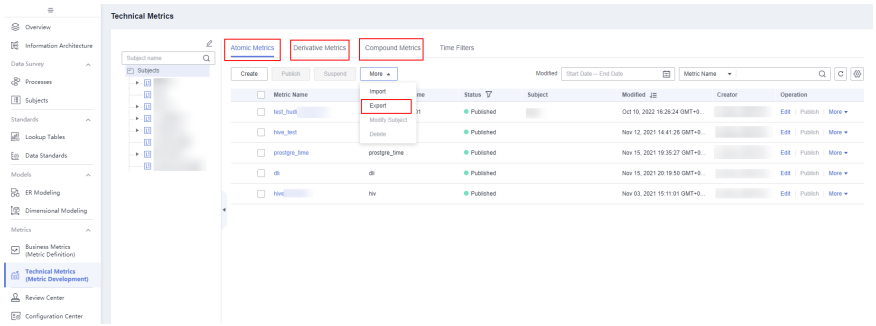
----End

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 3-29 Exporting technical metrics



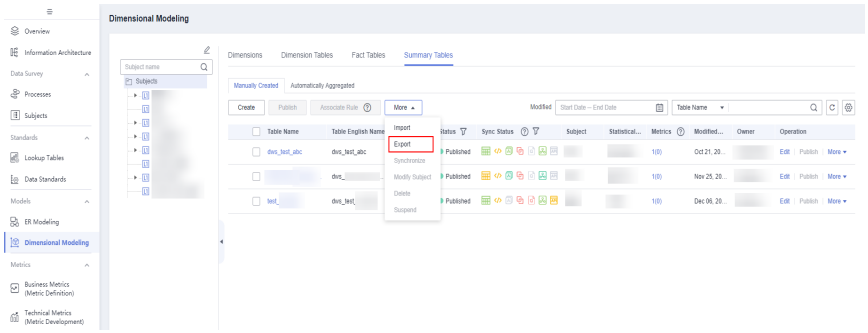
----End

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 3-30 Exporting summary tables



----End

Importing Table Data to the New Workspace

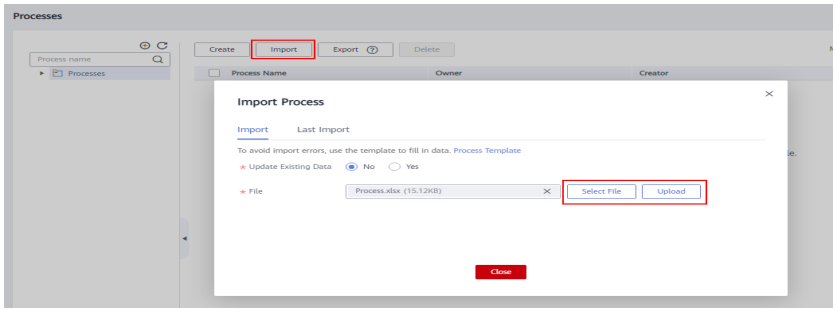
Log in to the console, access the DataArts Architecture module of the new workspace, and perform the following operations to import **processes**, **subjects**, **lookup tables**, **data standards**, **ER modeling tables/entities**, **dimensions/fact tables**, **business metrics**, **technical metrics**,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

Importing processes

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

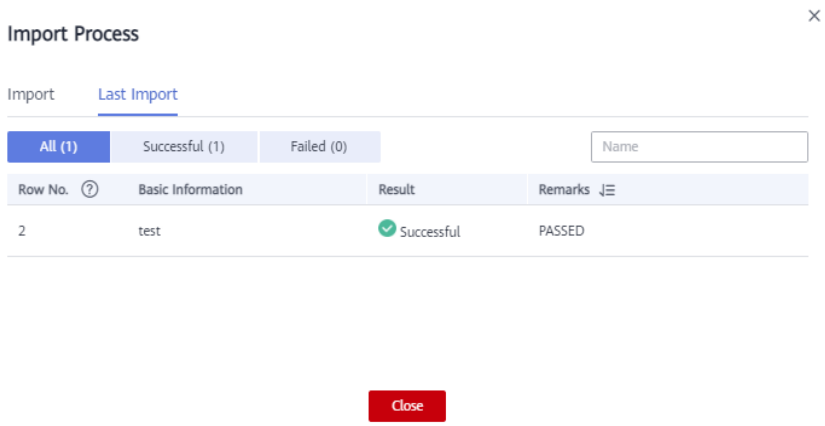
Step 2 Click **Import** above the list. In the displayed dialog box, select and upload the process file to import.

Figure 3-31 Import processes



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 process import



----End

Importing subjects

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

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

Figure 3-33 Importing subjects

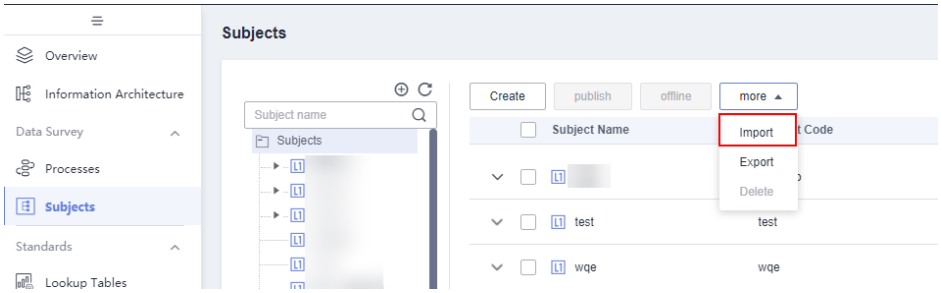
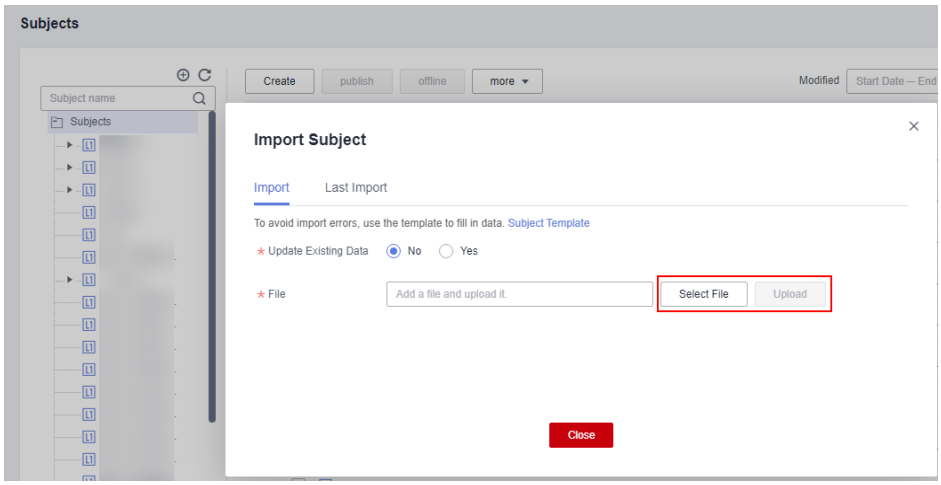
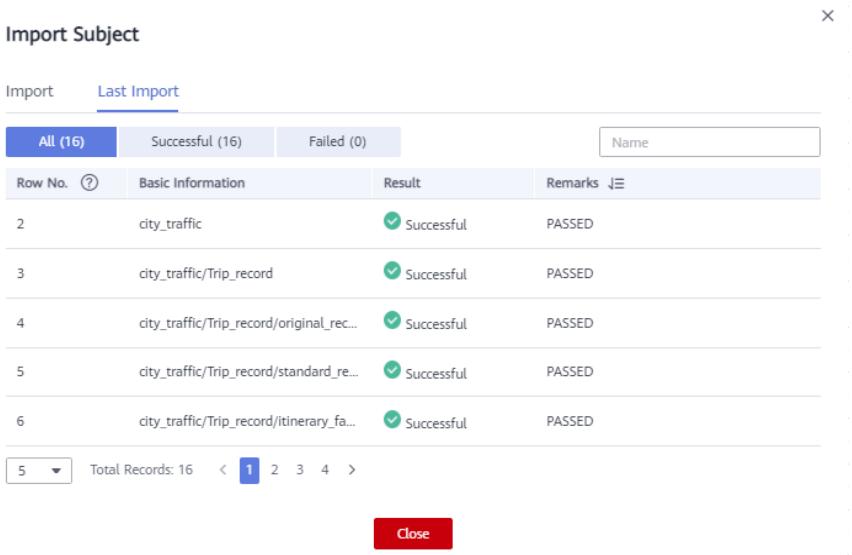


Figure 3-34 Selecting a file



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

Figure 3-35 Successful subject import



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

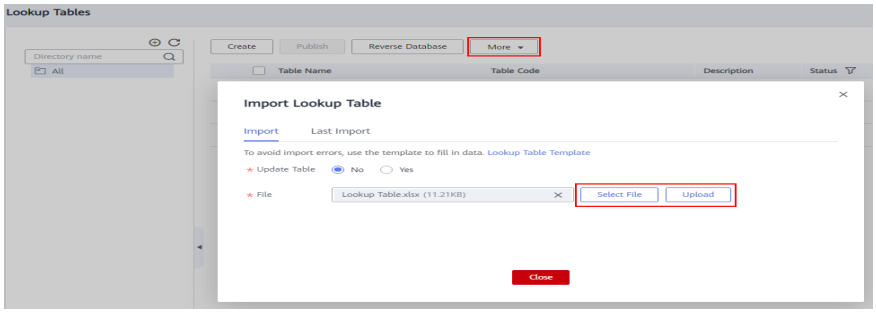
----End

Importing lookup tables

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

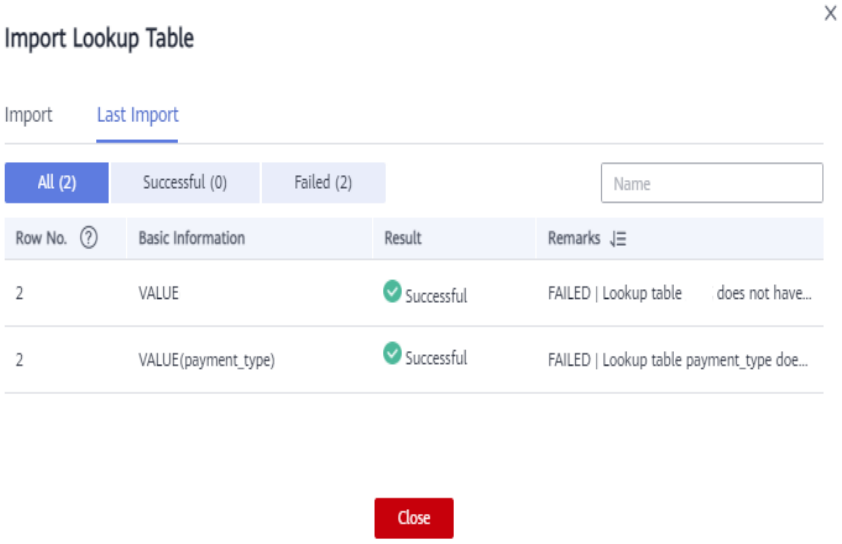
Step 2 Click **More** above the list and select **Import** from the drop-down list box. In the displayed dialog box, select and upload the lookup table file to import.

Figure 3-36 Importing lookup tables



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

Figure 3-37 Successful import of lookup tables



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

----End

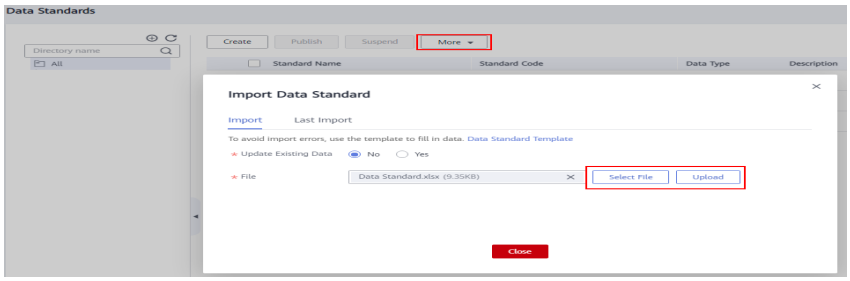
Importing data standards

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**.

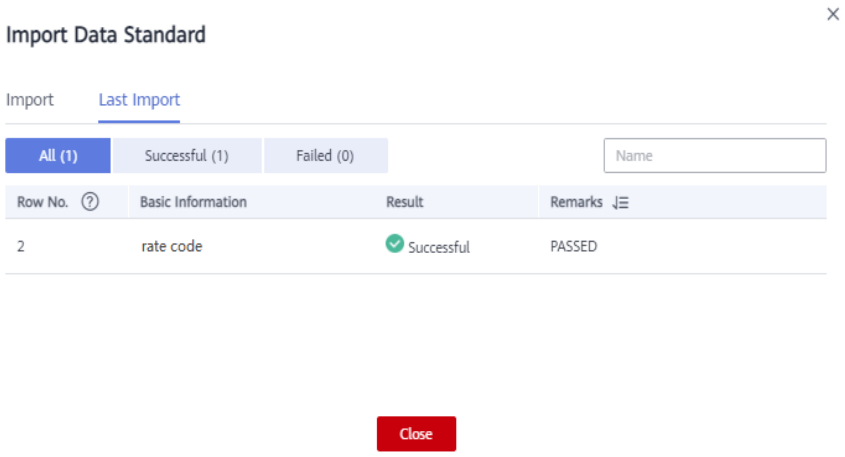
Step 3 Click **More** above the list and select **Import** from the drop-down list box. In the displayed dialog box, select and upload the data standard file to import.

Figure 3-38 Importing data standards



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

Figure 3-39 Successful import of data standards



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

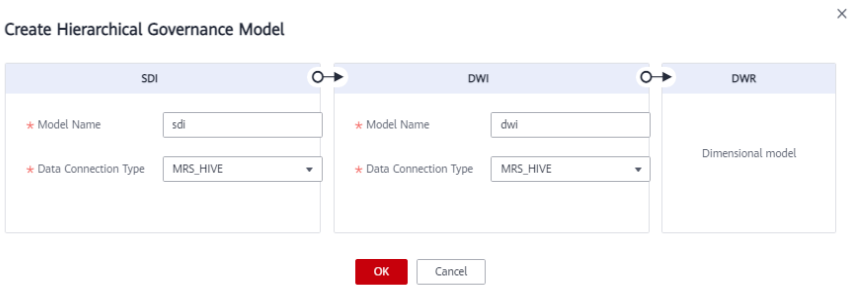
----End

Importing ER modeling tables/entities

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

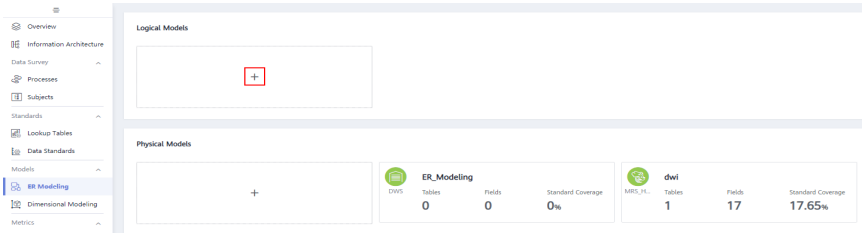
Step 2 If no ER model has been created, a dialog box is displayed, asking you to create a hierarchical governance model. Create an SDI and a DWI model for the new workspace by referring to the **ER Modeling** page of the old workspace, and then click **OK**.

Figure 3-40 Creating a hierarchical governance model



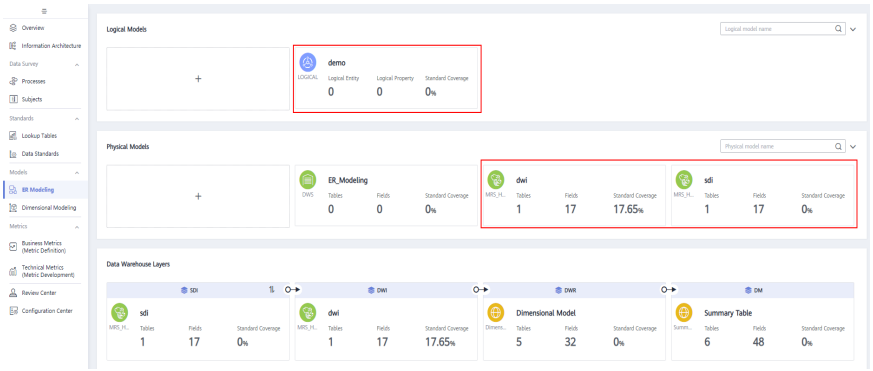
Step 3 If a logical model has been created in the old workspace, click + to create a logical model in the new workspace.

Figure 3-41 Creating a logical model



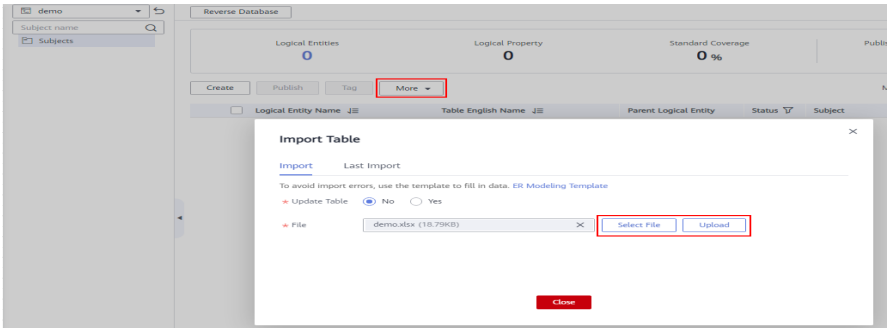
Step 4 Access a logical or physical model to import tables/entities to the model. This section uses logical model **demo** as an example.

Figure 3-42 Accessing the model



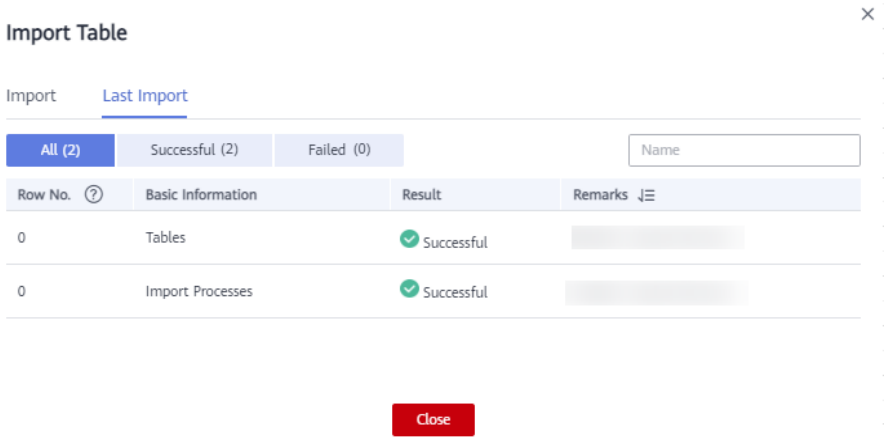
Step 5 Click **More** above the list and select **Import** from the drop-down list box. In the displayed dialog box, select and upload the table/entity file to import.

Figure 3-43 Importing ER modeling tables/entities



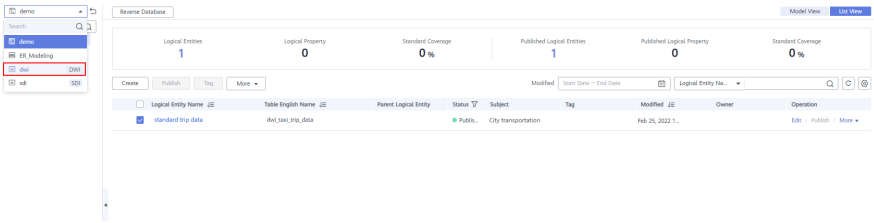
Step 6 After the file is uploaded, the system automatically starts importing it. After the file is imported, the import result is displayed.

Figure 3-44 Successful import of ER modeling tables/entities



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

Figure 3-45 Importing tables/entities to other models



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

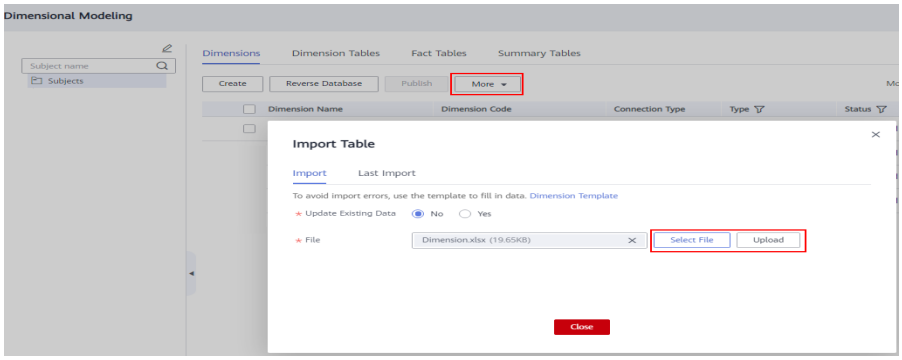
----End

Importing dimensions/fact tables

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

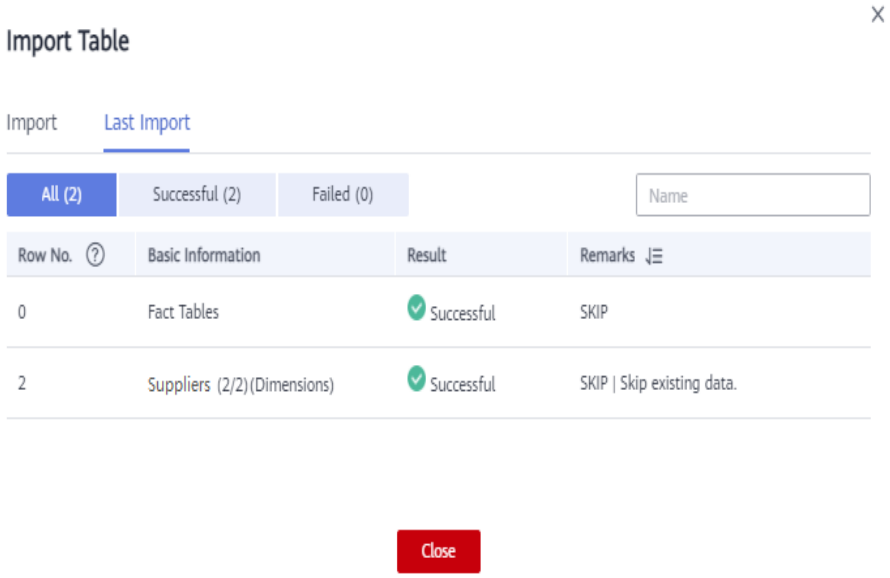
Step 2 On the displayed **Dimensions** page, click **More** above the list and select **Import** from the drop-down list box. In the displayed dialog box, select and upload the dimension file to import.

Figure 3-46 Importing dimensions



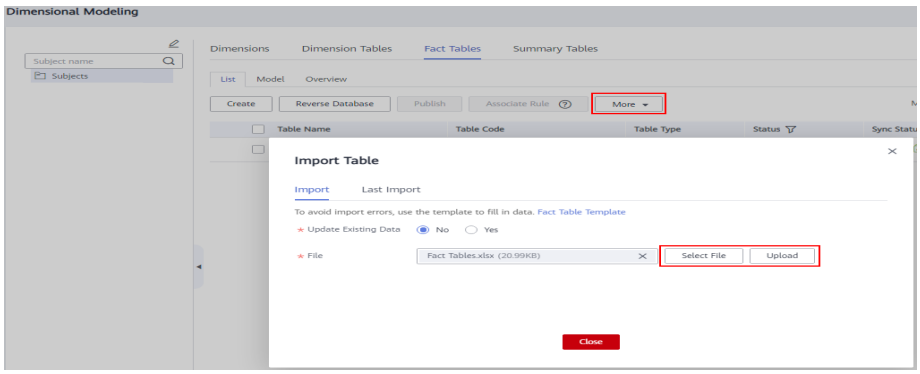
Step 3 After the file is uploaded, the system automatically starts importing it. After the file is imported, the import result is displayed.

Figure 3-47 Successful dimension import



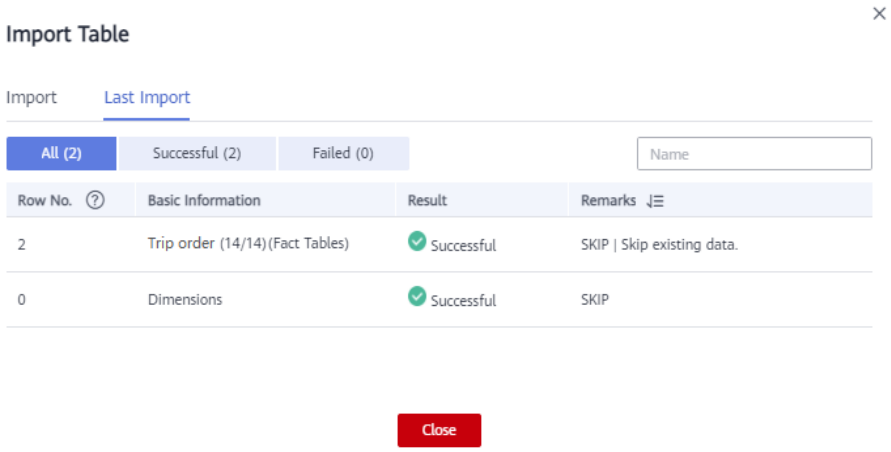
Step 4 Click the **Fact Tables** page, click **More** above the list, and select **Import** from the drop-down list box. In the displayed dialog box, select and upload the fact table file to import.

Figure 3-48 Importing fact tables



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

Figure 3-49 Successful import of fact tables



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

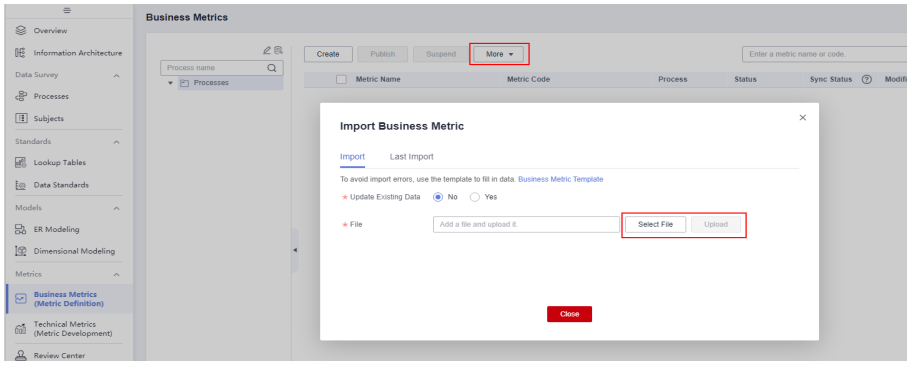
----End

Importing business metrics

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

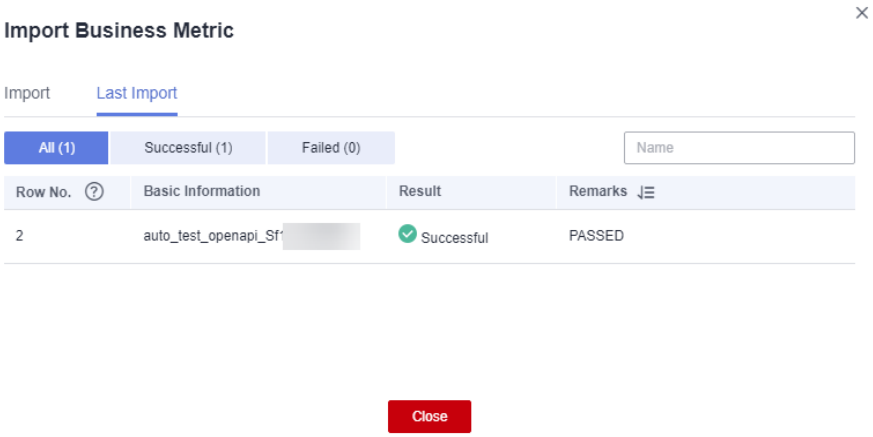
Step 2 Click **More** above the list and select **Import** from the drop-down list box. In the displayed dialog box, select and upload the business metric file to import.

Figure 3-50 Importing business metrics



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

Figure 3-51 Successful import of business metrics



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

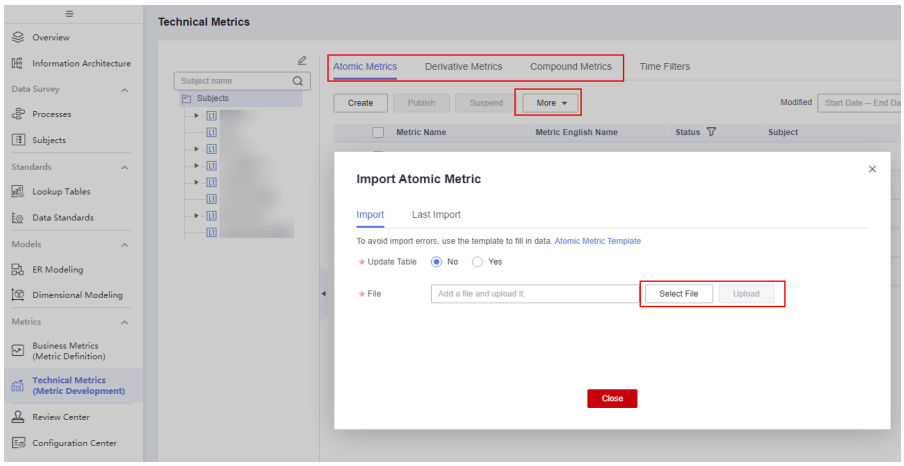
----End

Importing technical metrics

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

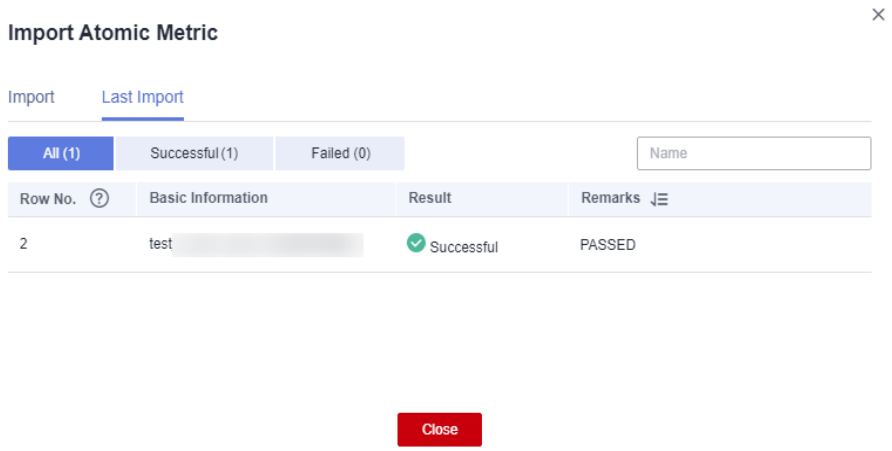
Step 2 Click the **Atomic Metrics**, **Derivative Metrics**, and **Compound Metrics** tab, respectively. Click **More** and select **Import**. In the displayed dialog box, select and upload a technical metric file.

Figure 3-52 Importing technical metrics



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

Figure 3-53 Successful import of technical metrics



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

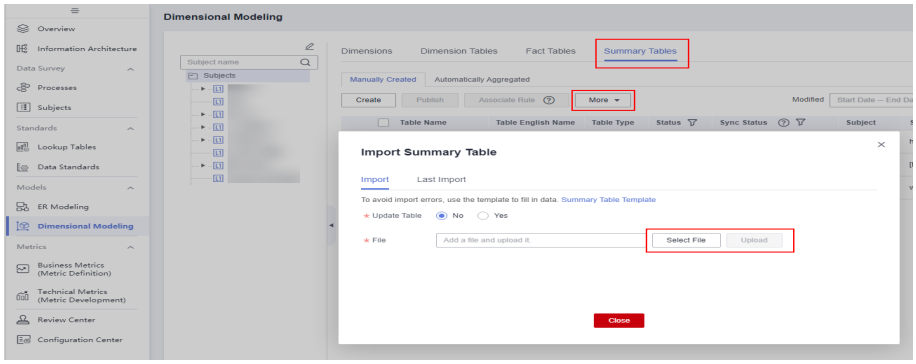
----End

Importing summary tables

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

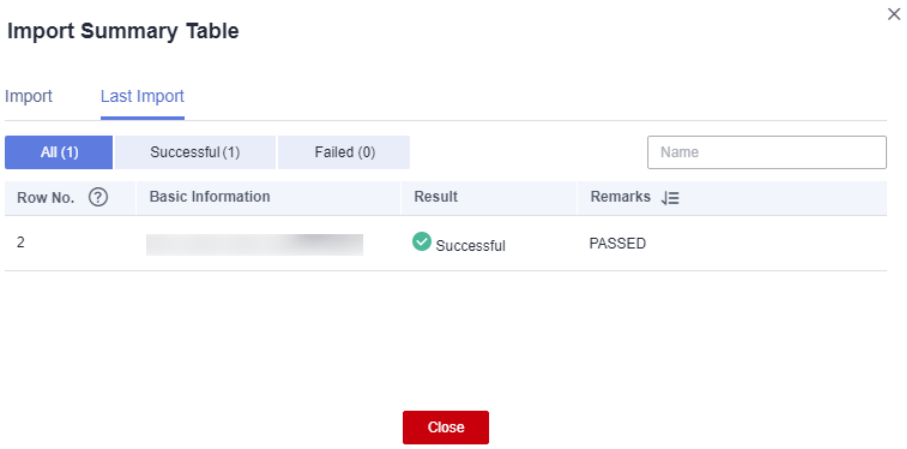
Step 2 Click the **Summary Tables** page, click **More** above the list, and select **Import** from the drop-down list box. In the displayed dialog box, select and upload the summary table file to import.

Figure 3-54 Importing summary tables



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

Figure 3-55 Successful import of summary tables



Step 4 After the import is successful, click **Publish**. The subject 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**:

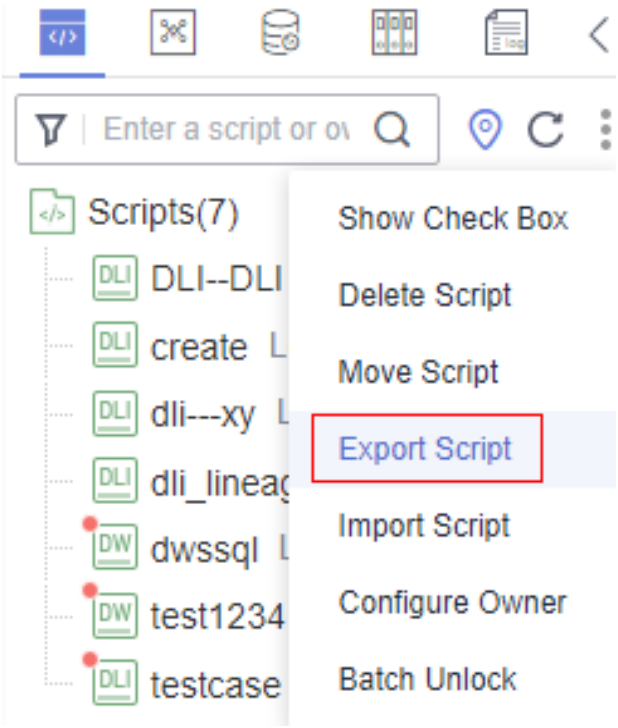
Exporting scripts

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**.

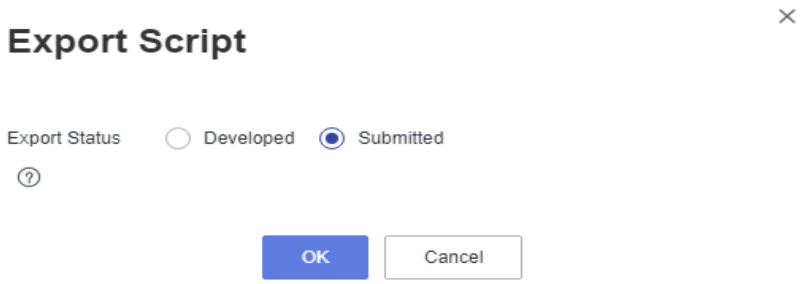
Step 3 Select scripts, click , and select **Export Script**. After the export is successful, you can obtain the exported .zip file.

Figure 3-56 Selecting and exporting scripts



Step 4 In the displayed **Export Script** dialog box, set **Export Status** and click **OK**.

Figure 3-57 Exporting scripts



----End

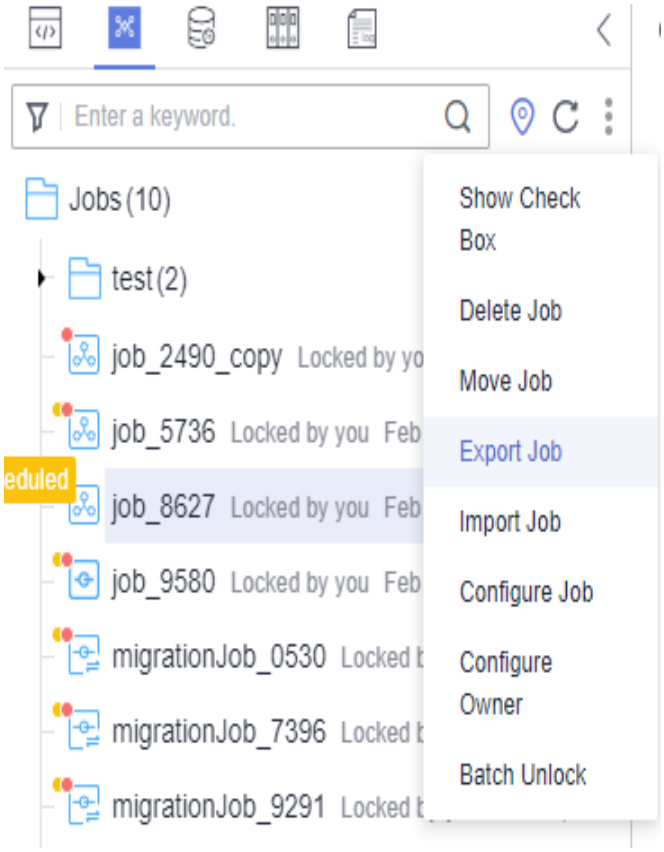
Exporting jobs

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

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

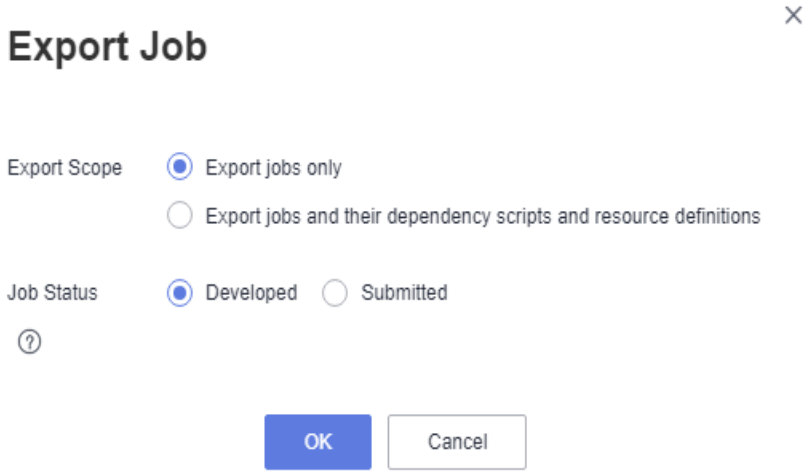
Step 3 Select jobs, click , and select Export Job. In the displayed dialog box, select **Export jobs only** or **Export jobs and their dependency scripts and resource definitions**. After the export is successful, you can obtain the exported .zip file.

Figure 3-58 Selecting and exporting jobs



Step 4 In the displayed **Export Job** dialog box, set **Export Scope** and **Job Status** and click **OK**. You can view the result in the download center.

Figure 3-59 Exporting jobs



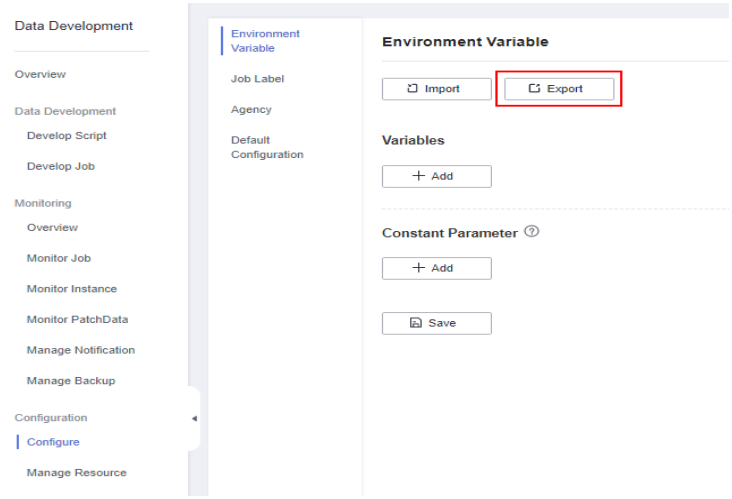
----End

Exporting environment variables

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

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

Figure 3-60 Exporting environment variables



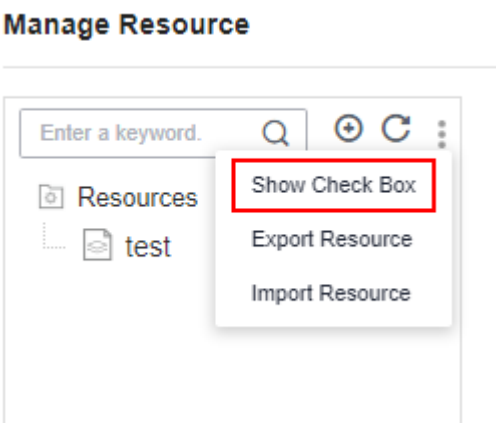
----End

Exporting resources

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

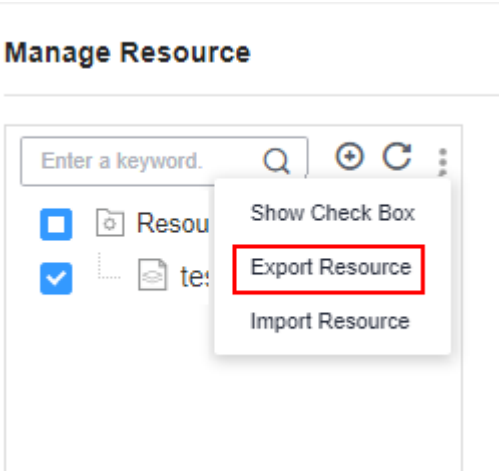
Step 2 Click in the resource directory and select **Show Check Box**.

Figure 3-61 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 3-62 Selecting and exporting resources



----End

Importing Data to the New Workspace

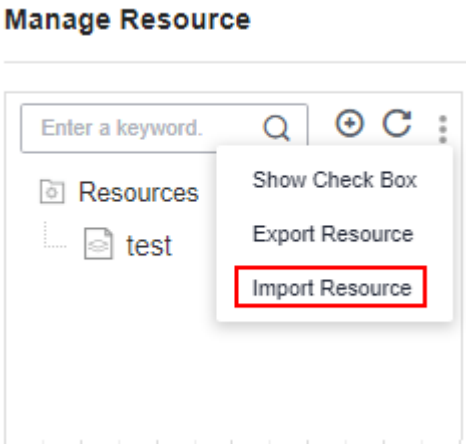
Log in to the console, access the **DataArts Factory** module of the new workspace, and perform the following operations in sequence to import **resources**, **environment variables**, **scripts**, and **jobs**:

Importing resources

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

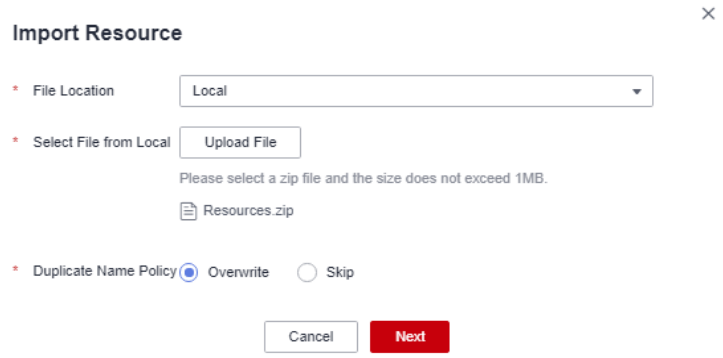
Step 2 Click in the resource directory and select **Import Resource**.

Figure 3-63 Selecting Import Resource



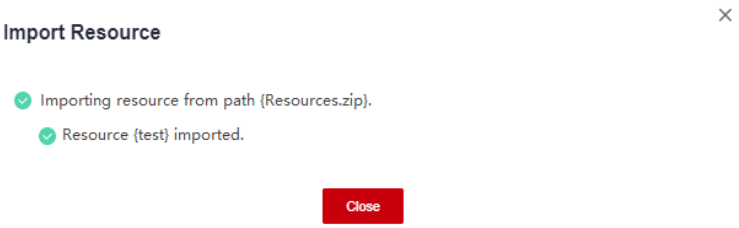
Step 3 In the displayed dialog box, select **Local** for **File Location**, select the resource file exported from the old workspace, select **Overwrite** for **Duplicate Name Policy**, and click **Next**.

Figure 3-64 Importing resources



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

Figure 3-65 Successful resource import



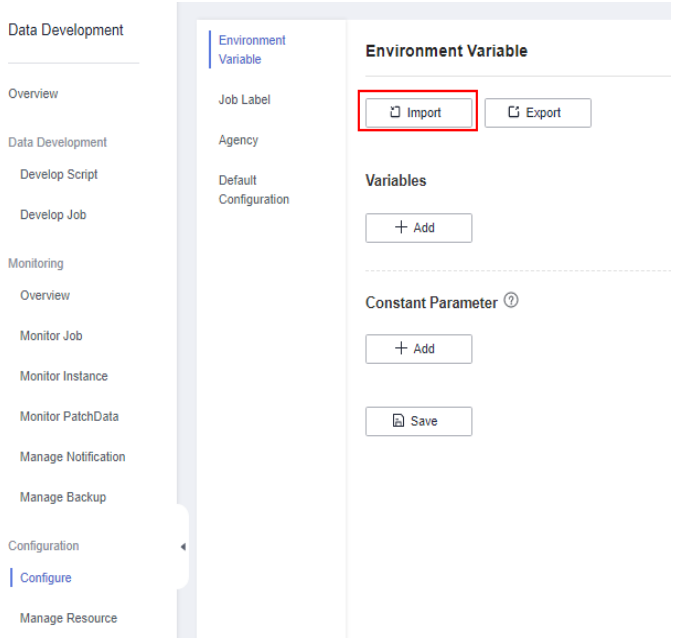
----End

Importing environment variables

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

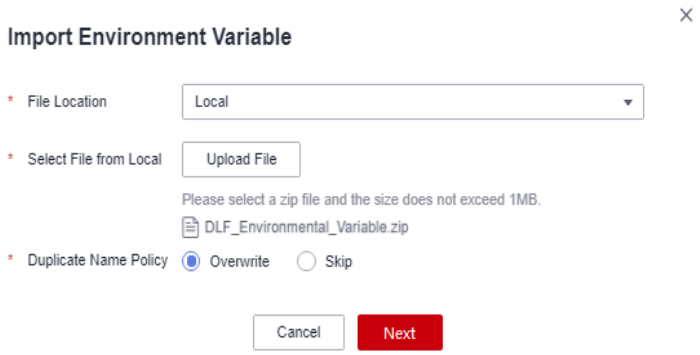
Step 2 On the **Environment Variable** page, click **Import**.

Figure 3-66 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 3-67 Importing environment variables



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

Figure 3-68 Confirming the import result

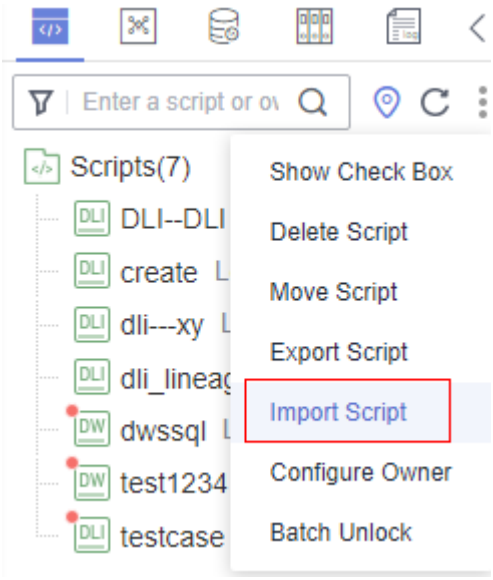
----End

Importing scripts

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

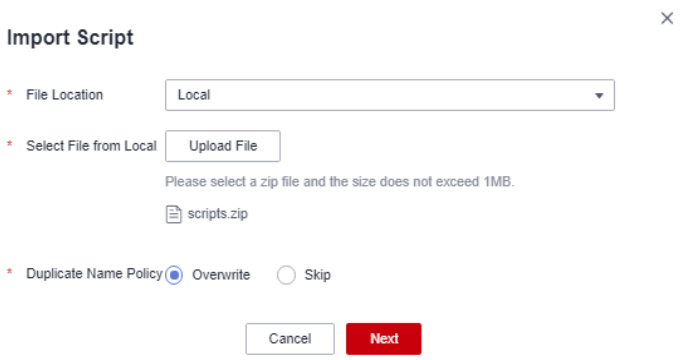
Step 2 Click in the script directory and select **Import Script**.

Figure 3-69 Selecting Import Script



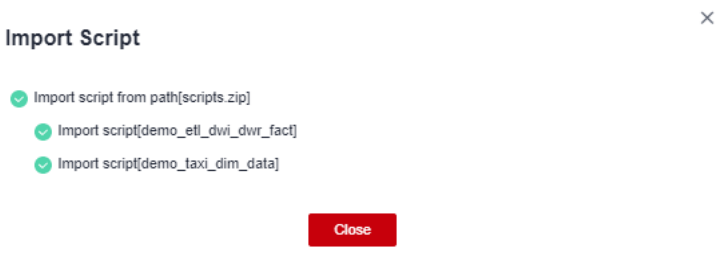
Step 3 In the displayed dialog box, select **Local** for **File Location**, select the script file exported from the old workspace, select **Overwrite** for **Duplicate Name Policy**, and click **Next**.

Figure 3-70 Importing scripts



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

Figure 3-71 Successful script import



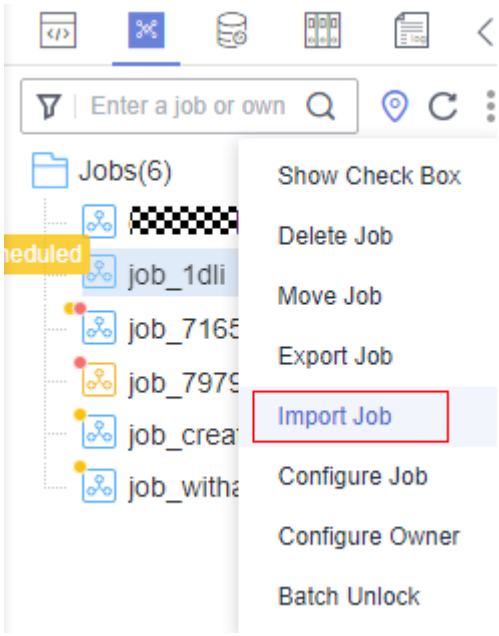
----End

Importing jobs

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

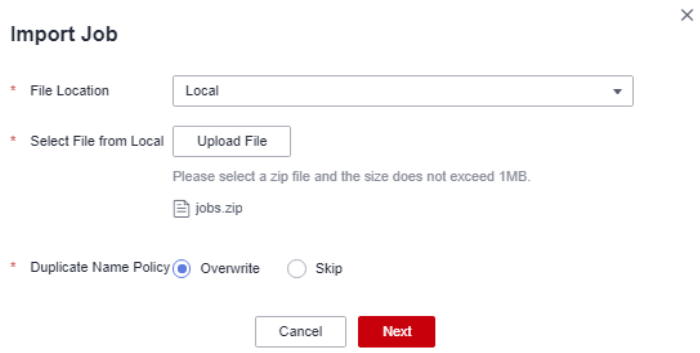
Step 2 Click in the job directory and select **Import Job**.

Figure 3-72 Selecting Import Job



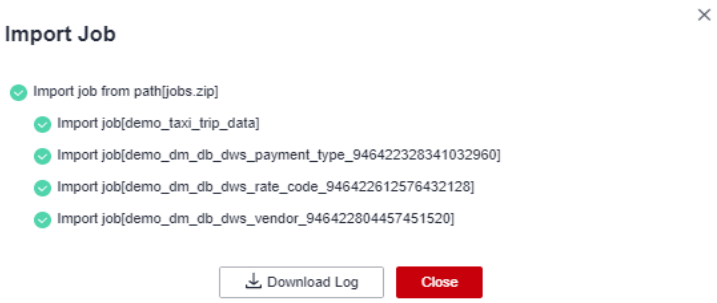
Step 3 In the displayed dialog box, select **Local** for **File Location**, select the job file exported from the old workspace, and click **Next**.

Figure 3-73 Importing jobs



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

Figure 3-74 Successful job import



----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. Each cell of the exported file can contain a maximum of 65,534 characters.
- You can import a file containing a maximum of 4 MB data. Each cell of the file to be imported can contain a maximum of 65,534 characters.
- You can export a maximum of 200 comparison jobs. Each cell of the exported file can contain a maximum of 65,534 characters.
- You can import a file containing a maximum of 4 MB data. Each cell of the file to be imported can contain a maximum of 65,534 characters.

Exporting Data from the Old Workspace

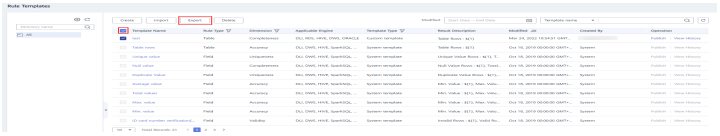
Log in to the console, access the **DataArts Quality** module of the old workspace, and perform the following operations in sequence to export **rule templates**, **quality jobs**, and **comparison jobs**:

Exporting rule templates

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

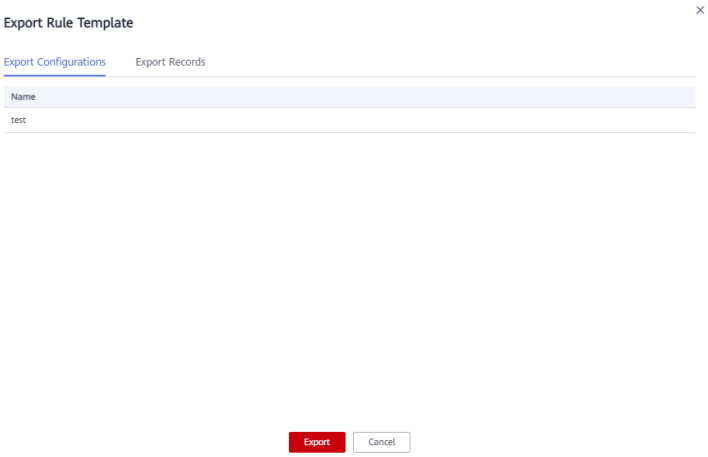
Step 2 In the rule template list, select the custom rule templates you want to export and click **Export**.

Figure 3-75 Exporting rule templates



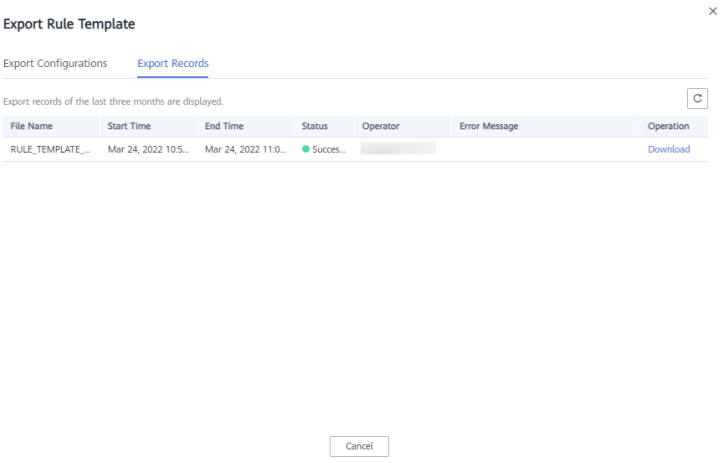
Step 3 In the displayed box, confirm the rule templates and click **Export**.

Figure 3-76 Confirming the rule templates to export



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 rule templates



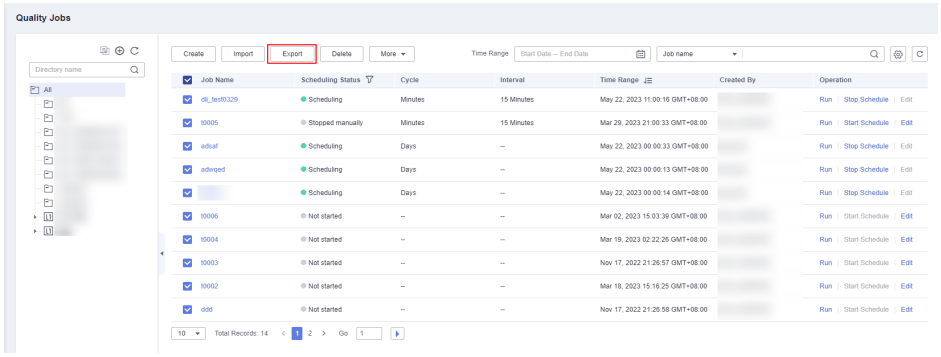
----End

Exporting quality Jobs

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

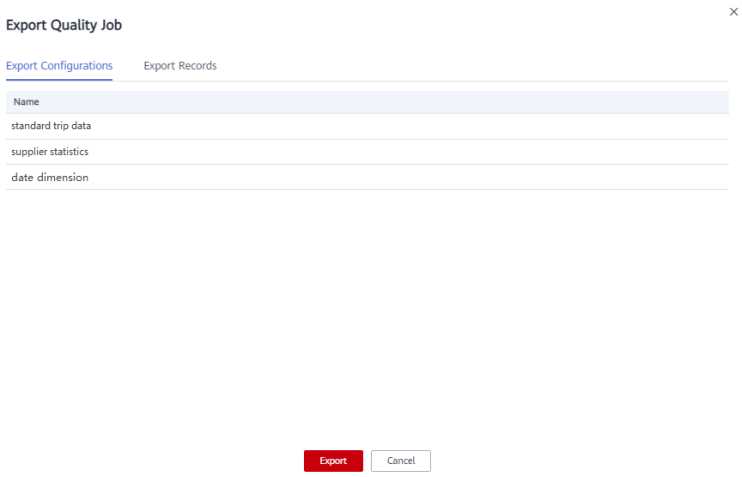
Step 2 In the quality job list, select the jobs you want to migrate and click **Export**.

Figure 3-78 Exporting quality jobs



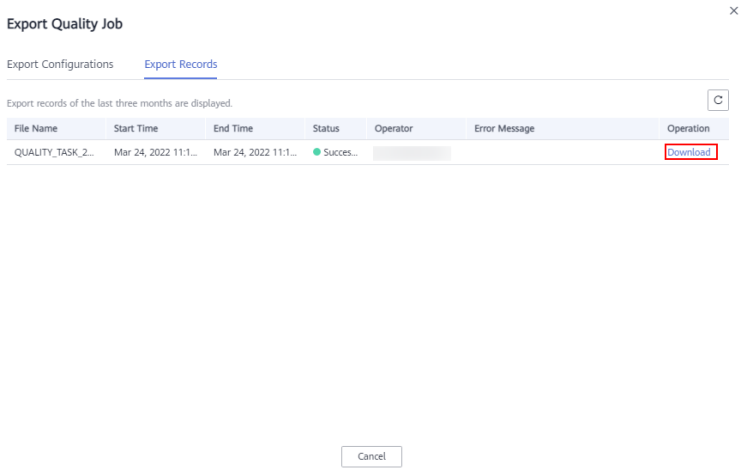
Step 3 In the displayed box, confirm the jobs and click **Export**.

Figure 3-79 Confirm the quality jobs to export



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 quality jobs



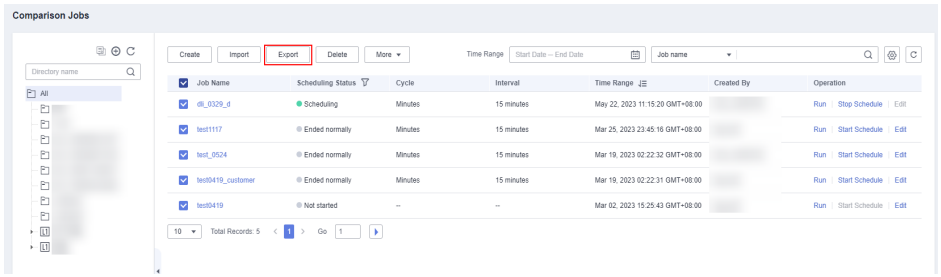
----End

Exporting comparison jobs

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

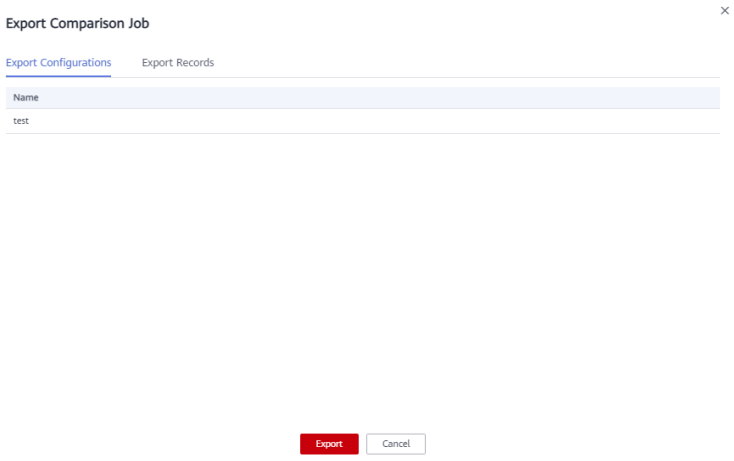
Step 2 In the comparison job list, select the jobs you want to migrate and click **Export**.

Figure 3-81 Exporting comparison jobs



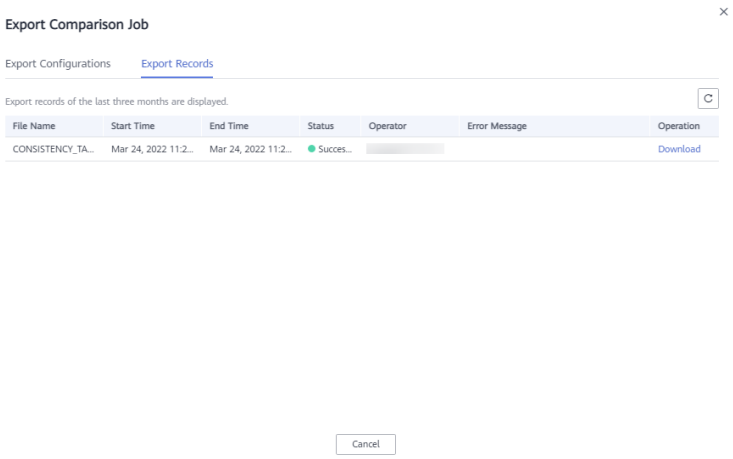
Step 3 In the displayed box, confirm the jobs and click **Export**.

Figure 3-82 Confirming the comparison jobs to export



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

Figure 3-83 Obtaining the exported file of comparison jobs



----End

Importing Data to the New Workspace

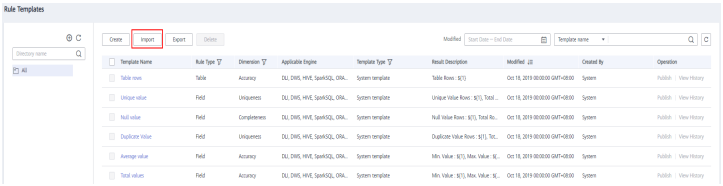
Log in to the console, access the **DataArts Quality** module of the new workspace, and perform the following operations in sequence to import **rule templates**, **quality jobs**, and **comparison jobs**:

Importing rule templates

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

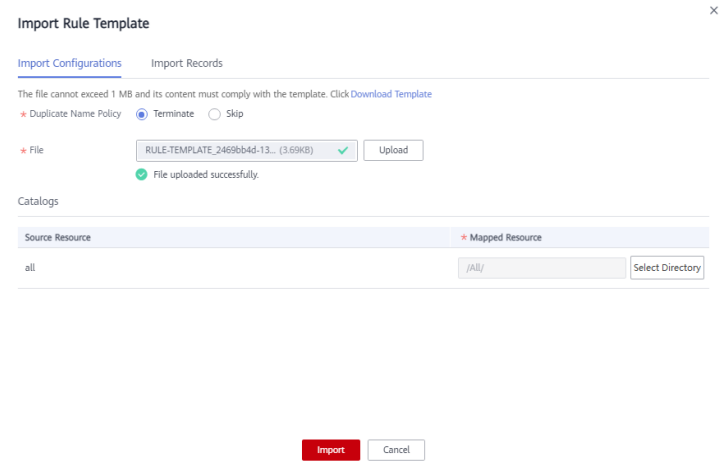
Step 2 Click **Import** above the rule template list.

Figure 3-84 Importing rule templates



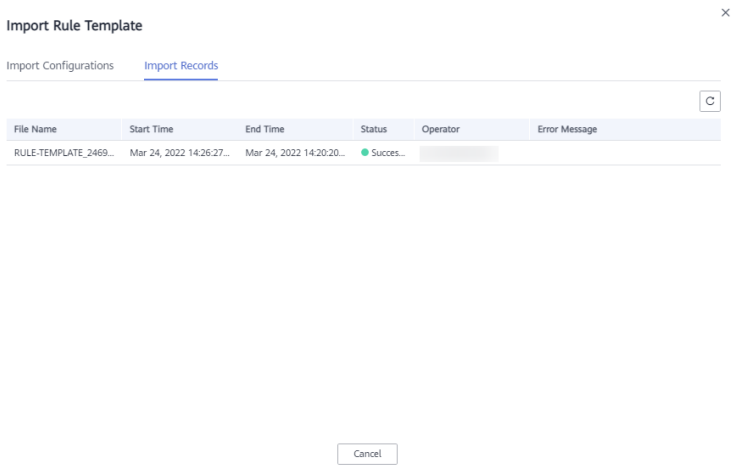
Step 3 In the displayed dialog box, select the rule template file exported from the old workspace, select the mapping directory, select **Terminate** for **Duplicate Name Policy**, and click **Import**.

Figure 3-85 Importing rule templates



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 rule template import result



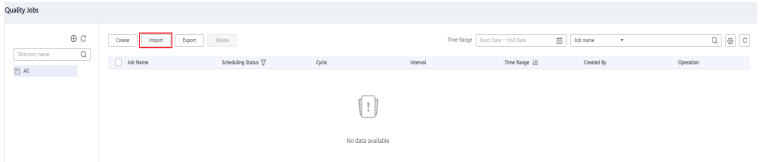
----End

Importing quality jobs

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

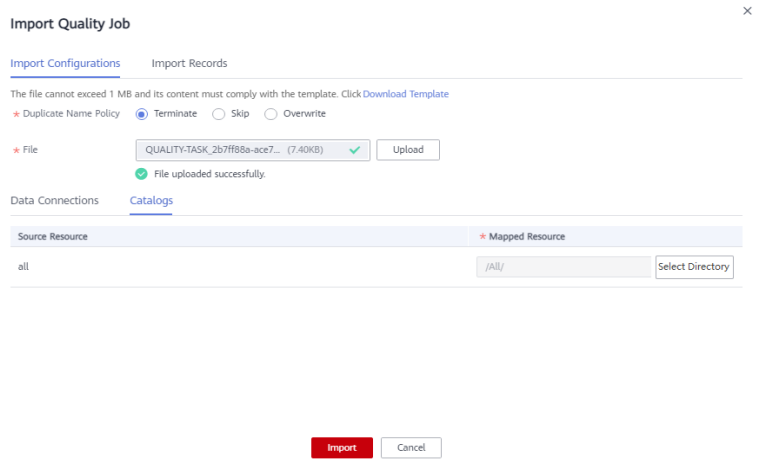
Step 2 Click **Import** above the quality job list.

Figure 3-87 Clicking Import



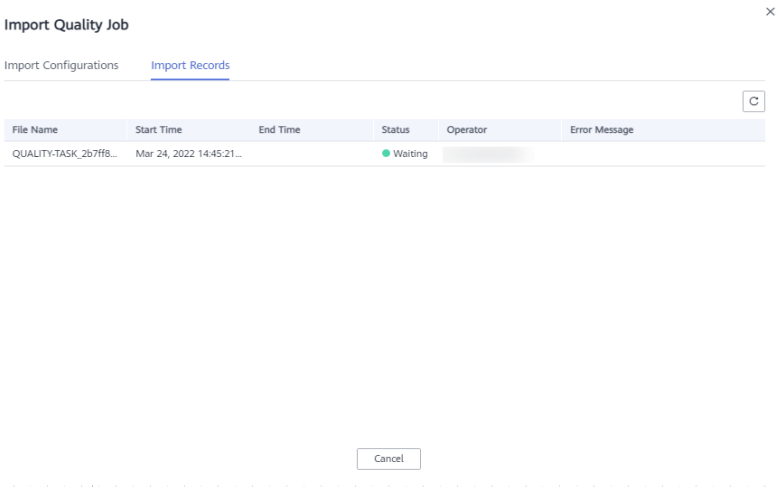
Step 3 In the displayed dialog box, select the quality job file exported from the old workspace, select the mapping directories for data connections, clusters, and directories, select **Terminate** for **Duplicate Name Policy**, and click **Import**.

Figure 3-88 Importing quality jobs



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 quality job import result



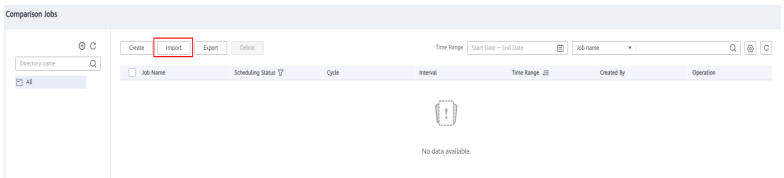
----End

Importing comparison jobs

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

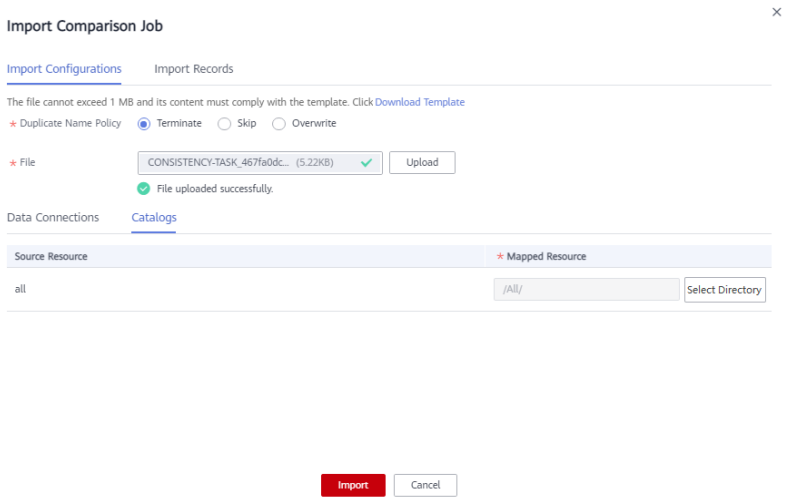
Step 2 Click **Import** above the comparison job list.

Figure 3-90 Clicking Import



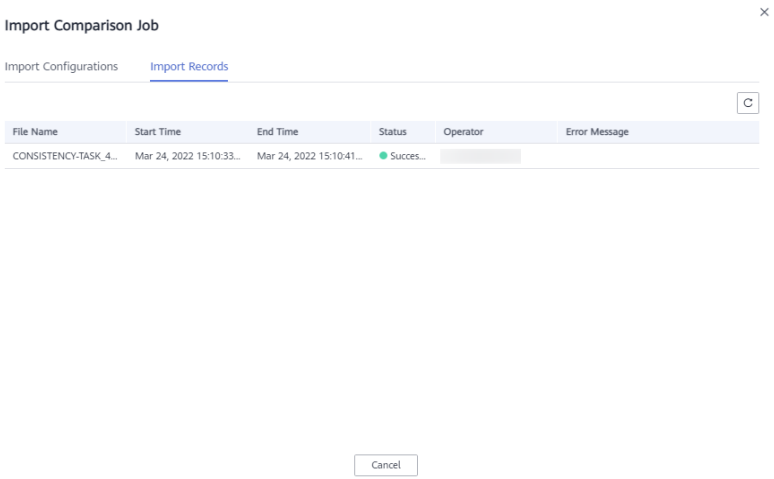
Step 3 In the displayed dialog box, select the comparison job file exported from the old workspace, select the mapping directories for data connections, clusters, and directories, select **Terminate** for **Duplicate Name Policy**, and click **Import**.

Figure 3-91 Importing comparison jobs



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 3-92 Checking the comparison job import result



----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**.

NOTE

The DataArts Catalog data that can be migrated by the Management Center includes classifications, tags, and collection tasks. Logical assets, technical assets, and metric assets cannot be directly imported or exported.

You can import the Management Center and DataArts Architecture data and run the newly imported collection task to regenerate logical assets, technical assets, and metric assets.

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 Authorizing Users to Use DataArts Studio by Complying with the Principle of Least Privilege

Scenario and Objectives

A data operations engineer is responsible for monitoring data quality and only needs the operation permissions of the DataArts Quality component of .

Figure 4-1 Permission system

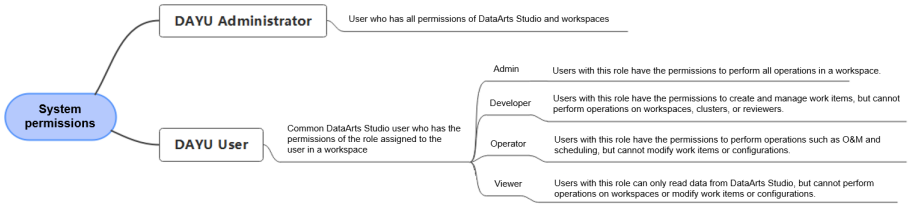


Figure 4-1 shows the permission system of . If the project administrator assigns the DAYU User system role and workspace developer role to the IAM account of the data operations engineer, the following risks arise:

1. Excessive permissions of dependent services: is a platform service that depends on other services such as MRS and GaussDB(DWS). The DAYU User system role has the administrator permissions of these dependent services. If the DAYU User role is assigned to the IAM account of the data operations engineer, the IAM account also has the administrator permissions of the dependent services.

To resolve this problem, the project administrator can configure the least privilege which meets requirements while avoiding excessive permissions.

1. Assign the DAYU User system role to the IAM account of the data operations engineer, delete permissions of dependent services from the IAM account, and grant the minimum permissions to the IAM account.

Procedure

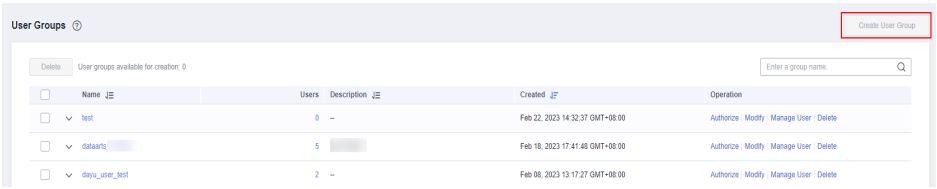
1. **Creating a User Group and Assigning the DAYU User Role to the Group**: Create a user group for the IAM account of the data operations engineer and assign the **DAYU User** role to the user group.
2. **Deleting Permissions of Dependent Services from the User Group and Configuring Minimum Permissions**: Delete the default administrator permissions of dependent services from the user group and configure the minimum permissions.
3. **Creating an IAM User and Adding It to the User Group**: Create an IAM user for the data operations engineer and add the user to the user group.
4. **Adding a Workspace Member and Assigning a Role**: Add the created IAM user to the workspace and assign the role to the user.
5. **Logging In to the Console and Verifying Permissions**: Log in to the console as the created user and check whether the permission configuration meets expectations.

Creating a User Group and Assigning the DAYU User Role to the Group

Step 1 Log in to the IAM console using the a Huawei account.

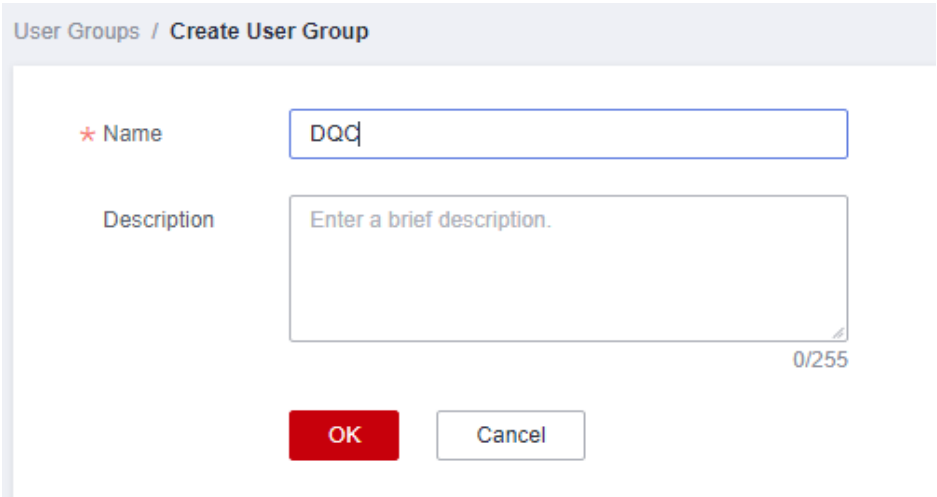
Step 2 In the navigation pane, choose **User Groups**. In the upper right corner of the **User Groups** page, click **Create User Group**.

Figure 4-2 Creating a user group



Step 3 On the displayed page, enter the user group name **DQC**.

Figure 4-3 Entering the user group name



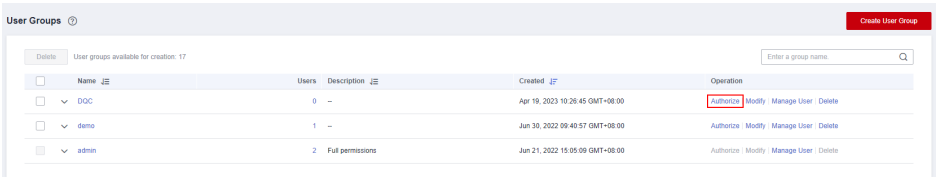
Step 4 Click **OK**. The created user group is displayed in the user group list.

NOTE

You can create a maximum of 20 user groups. If this quota does not meet your requirements, you can apply for a higher quota. For details, see **How Do I Increase My Quota?**.

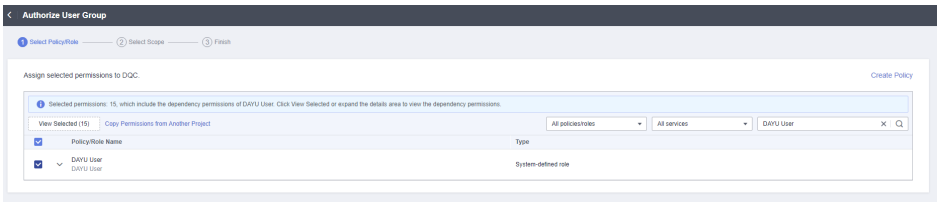
Step 5 In the user group list, click **Authorize** in the row that contains the newly created user group.

Figure 4-4 Going to the user group authorization page



Step 6 Enter **DAYU User** in the search box, select the system role, and click **Next**.

Figure 4-5 Assigning a role



NOTE

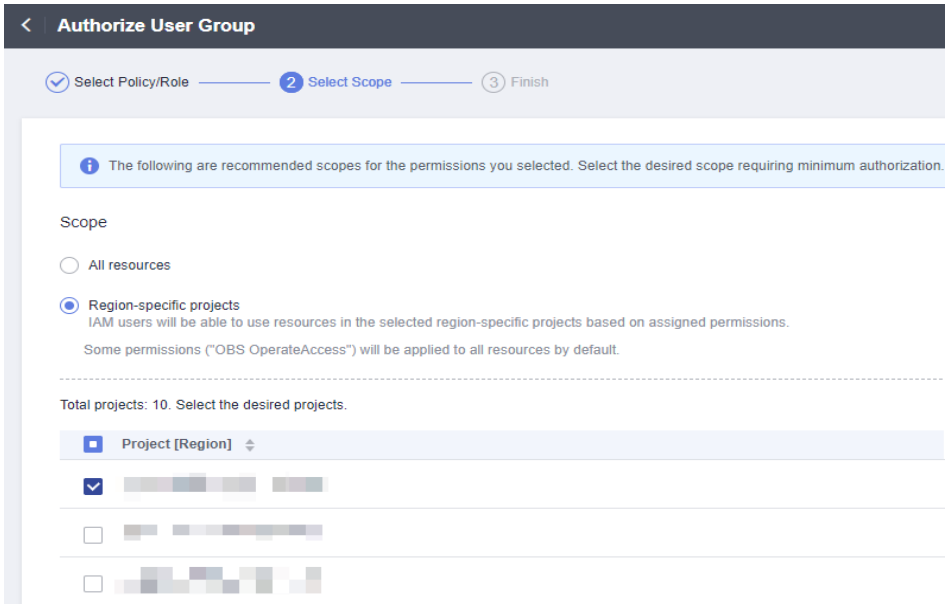
Do not select the **DAYU Administrator** role because it has all the execution permissions on DataArts Studio and is not controlled by the workspace permissions.

Step 7 Select **Region-specific projects** for **Scope** and select projects. Then click **OK**.

NOTE

DataArts Studio is a project-level service deployed in specific physical regions. If you select **All resources** for **Scope**, the permission takes effect in all projects of all regions. If you select **Region-specific projects** for **Scope**, the permission takes effect only for a specified project. When accessing DataArts Studio, the IAM user must switch to the region where they have been assigned the required permissions.

Figure 4-6 Setting the scope



----End

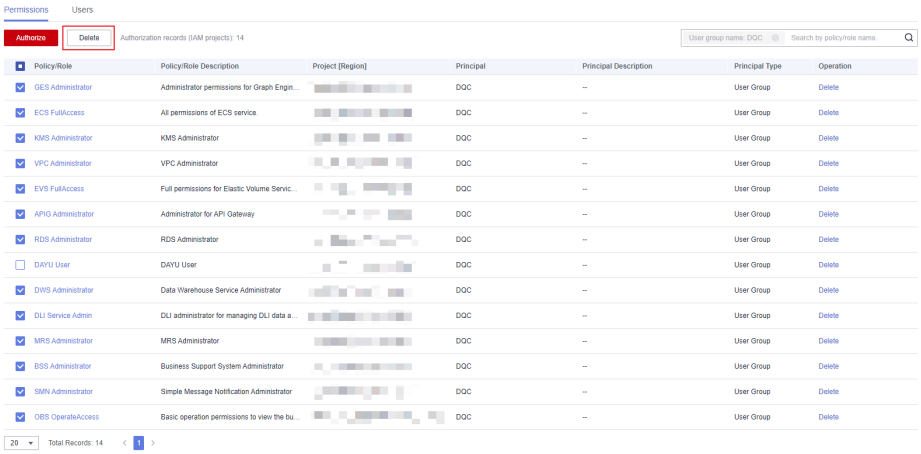
Deleting Permissions of Dependent Services from the User Group and Configuring Minimum Permissions

Step 1 In the left navigation pane on the IAM console, choose **User Groups**. On the **User Groups** page, click the **DQC** user group to go to the user group details page.

Figure 4-7 Accessing the user group details page

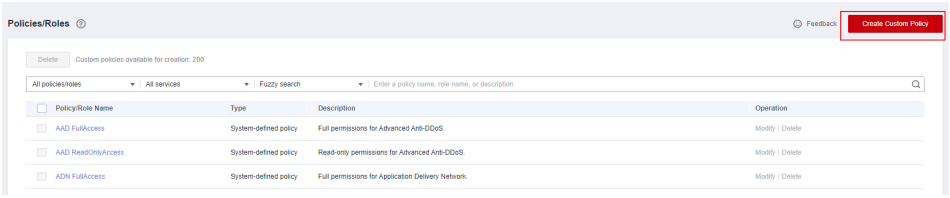
Step 2 On the **Permissions** tab page, change the number of records displayed to 20 to show all the 14 records. Select all the permissions except **DAYU User** and click **Delete** above the list.

Figure 4-8 Deleting permissions of dependent services



Step 3 Return to the homepage of the IAM console and choose **Permissions** > **Policies/Roles**. In the upper right corner of the displayed page, click **Create Custom Policy**.

Figure 4-9 Creating a custom policy

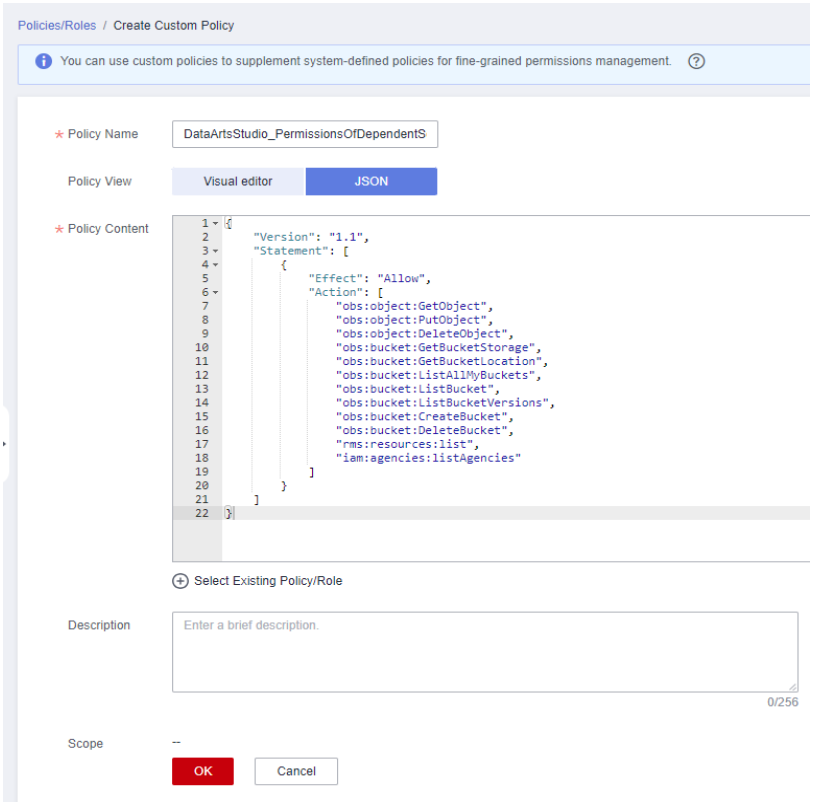


Step 4 On the **Create Custom Policy** page, select **JSON** for **Policy View** and create custom policies **DataArtsStudio_PermissionsOfDependentServices_global** and **DataArtsStudio_PermissionsOfDependentServices_region**.

NOTE

- You cannot create a custom policy for both a global dependent service and a regional dependent service. Instead, you must create two separate policies.
- The policy content is the minimum permissions of the services on which DataArts Studio components depend. For details, see **DataArts Studio Permissions Management**.

Figure 4-10 Creating a custom policy



- **DataArtsStudio_PermissionsOfDependentServices_global**: custom policy for a global dependent cloud service

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "obs:object:GetObject",
            "obs:object:PutObject",
            "obs:object:DeleteObject",
            "obs:bucket:GetBucketStorage",
            "obs:bucket:GetBucketLocation",
            "obs:bucket:ListAllMyBuckets",
            "obs:bucket:ListBucket",
            "obs:bucket:ListBucketVersions",
            "obs:bucket:CreateBucket",
            "obs:bucket:DeleteBucket",
            "rms:resources:list",
            "iam:agencies:listAgencies"
         ]
      }
   ]
}
```

- **DataArtsStudio_PermissionsOfDependentServices_region**: custom policy for a regional dependent cloud service

```
{
   "Version": "1.1",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "cdm:cluster:get",
```



```
            "cdm:cluster:list",
            "cdm:cluster:create",
            "cdm:link:operate",
            "cdm:job:operate",
            "ces:*:get",
            "ces:*:list",
            "cloudtable:*:get",
            "cloudtable:*:list",
        "css:*:get",
            "css:*:list",
            "dis:streams:list",
            "dis:transferTasks:list",
            "dli:queue:submitJob",
            "dli:queue:cancelJob",
            "dli:table:insertOverwriteTable",
            "dli:table:insertIntoTable",
            "dli:table:alterView",
            "dli:table:alterTableRename",
            "dli:table:compaction",
            "dli:table:truncateTable",
            "dli:table:alterTableDropColumns",
            "dli:table:alterTableSetProperties",
            "dli:table:alterTableChangeColumn",
            "dli:table:showSegments",
            "dli:table:alterTableRecoverPartition",
            "dli:table:dropTable",
            "dli:table:update",
            "dli:table:alterTableDropPartition",
            "dli:table:alterTableAddPartition",
            "dli:table:alterTableAddColumns",
            "dli:table:alterTableRenamePartition",
            "dli:table:delete",
            "dli:table:alterTableSetLocation",
            "dli:table:describeTable",
            "dli:table:showPartitions",
            "dli:table:showCreateTable",
            "dli:table:showTableProperties",
            "dli:table:select",
            "dli:resource:updateResource",
            "dli:resource:useResource",
            "dli:resource:getResource",
            "dli:resource:listAllResource",
            "dli:resource:deleteResource",
            "dli:database:explain",
            "dli:database:createDatabase",
            "dli:database:dropFunction",
            "dli:database:createFunction",
            "dli:database:displayAllDatabases",
        "dli:database:displayAllTables",
            "dli:database:displayDatabase",
            "dli:database:describeFunction",
            "dli:database:createView",
            "dli:database:createTable",
            "dli:database:showFunctions",
            "dli:database:dropDatabase",
            "dli:group:useGroup",
            "dli:group:updateGroup",
            "dli:group:listAllGroup",
            "dli:group:getGroup",
            "dli:group:deleteGroup",
            "dli:column:select",
            "dli:jobs:start",
            "dli:jobs:export",
            "dli:jobs:update",
            "dli:jobs:list",
            "dli:jobs:listAll",
            "dli:jobs:get",
            "dli:jobs:delete",
            "dli:jobs:create",
```



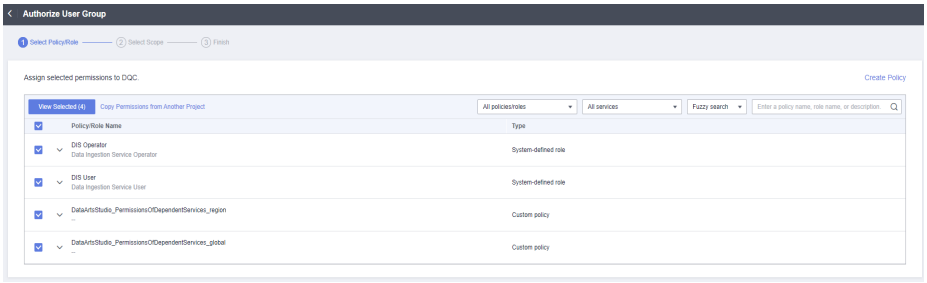
```
            "dli:jobs:stop",
        "dli:variable:update",
        "dli:variable:delete",
            "dws:cluster:list",
            "dws:cluster:getDetail",
            "dws:openAPICluster:getDetail",
            "ecs:servers:get",
        "ecs:servers:list",
            "ecs:servers:stop",
            "ecs:servers:start",
        "ecs:flavors:get",
            "ecs:cloudServerFlavors:get",
            "ecs:cloudServers:list",
            "ecs:availabilityZones:list",
            "ges:graph:access",
            "ges:metadata:create",
            "ges:jobs:list",
            "ges:graph:operate",
            "ges:jobs:getDetail",
            "ges:graph:getDetail",
            "ges:graph:list",
            "ges:metadata:list",
            "ges:metadata:getDetail",
            "ges:metadata:delete",
            "ges:metadata:operate",
            "kms:cmk:get",
            "kms:cmk:list",
            "kms:cmk:create",
            "kms:cmk:decrypt",
            "kms:cmk:encrypt",
            "kms:dek:create",
            "kms:dek:encrypt",
            "kms:dek:decrypt",
            "mrs:cluster:get",
            "mrs:cluster:list",
            "mrs:job:get",
            "mrs:job:list",
            "mrs:job:submit",
            "mrs:job:stop",
            "mrs:job:delete",
            "mrs:sql:execute",
            "mrs:sql:cancel",
            "rds:*:get",
            "rds:*:list",
            "smn:topic:publish",
            "smn:topic:list",
        "vpc:publicIps:list",
            "vpc:publicIps:get",
            "vpc:vpcs:get",
            "vpc:vpcs:list",
            "vpc:subnets:get",
            "vpc:securityGroups:get",
            "vpc:firewalls:list",
            "vpc:routeTables:list",
            "vpc:subNetworkInterfaces:list"
          ]
        }
    ]
}
```

Step 5 Go to the **User Groups** page again, locate the DQC user group, and click **Authorize** in the **Operation** column. Select **Policies/Roles** for **Authorization Model** and select the following system roles and custom policies:

- System roles: **DIS Operator** and **DIS User**
- Custom policy **DataArtsStudio_PermissionsOfDependentServices_global**
- Custom policy **DataArtsStudio_PermissionsOfDependentServices_region**

> **NOTE**
>
> The image read permission of SWR is required only when a custom image is selected for a **DLI Spark** node of a job in DataArts Factory. The account administrator is advised to grant permissions to users through image authorization. (Log in to the SWR console as the SWR administrator, choose **My Images** in the navigation pane on the left, access the details page of the required custom image, and grant the read permission of the image to users.) Otherwise, you need to grant the SWR Administrator permissions to users.

Figure 4-11 Configuring minimum permissions for the user group



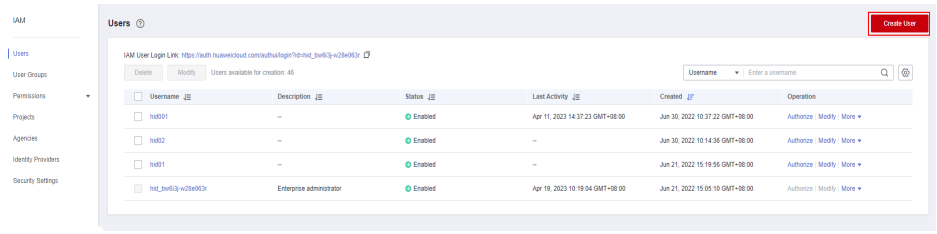
Step 6 After the authorization is successful, the minimum permissions have been configured for dependent services.

----End

Creating an IAM User and Adding It to the User Group

Step 1 On the IAM console, choose **Users** from the navigation pane on the left and click **Create User** in the upper right corner.

Figure 4-12 Creating a user



Step 2 On the **Set User Details** page, set the following parameters and click **Next** in the lower right corner.

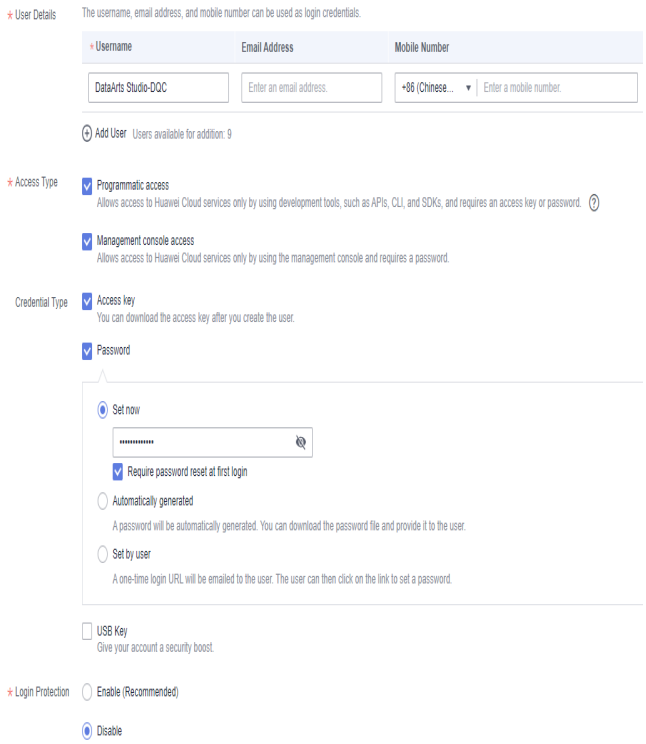
- **User Details**: Enter **DataArts Studio-DQC** for **Username**.
- **Access Type**: Select **Programmatic access** and **Management console access**.

  > **NOTE**
  >
  > An IAM user can pass the authentication and access DataArts Studio through an API or SDK only if **Programmatic access** is selected for **Access Type** during the creation of the IAM user.

- **Credential Type**: Select **Access key** and **Password**. You are advised to select **Set now** for **Password**.

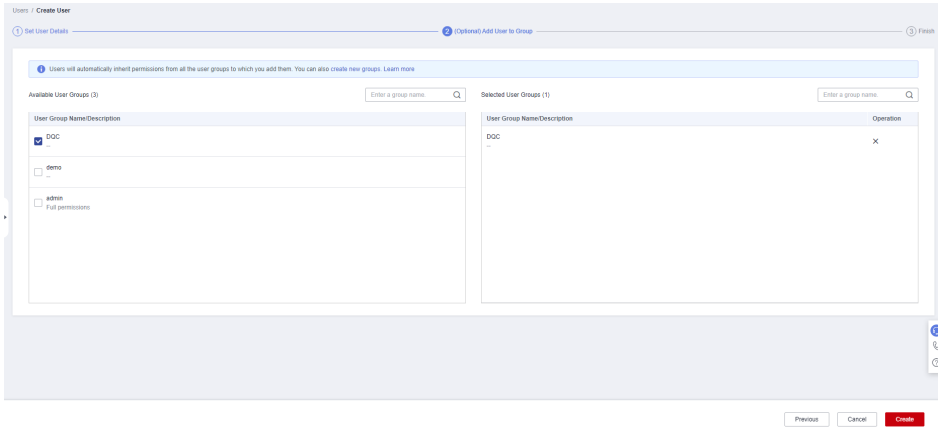
- **Login Protection**: Set this parameter as required. Generally, this function does not need to be enabled.

Figure 4-13 Configuring user information



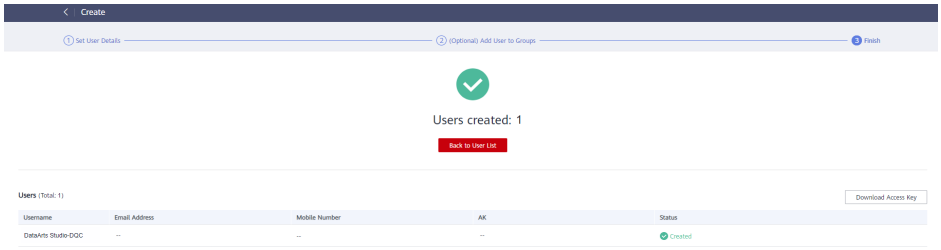
Step 3 Select the DQC user group and click **Create** in the lower right corner of the page.

Figure 4-14 Create



Step 4 Return to the user list to view the created user.

Figure 4-15 Successful creation

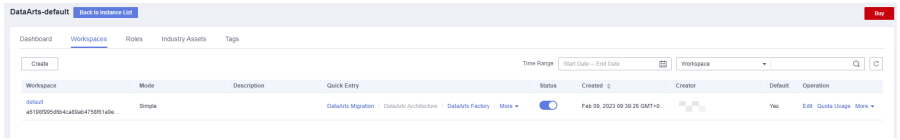


----End

Adding a Workspace Member and Assigning a Role

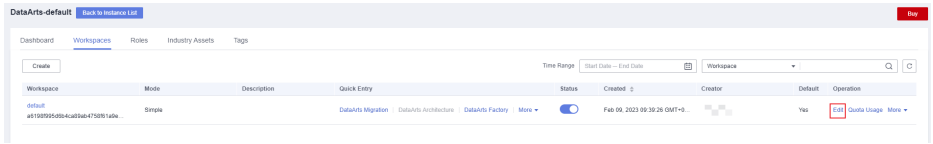
Step 1 Log in to the DataArts Studio console using the a Huawei account and click the **Workspaces** tab.

Figure 4-16 Workspaces page



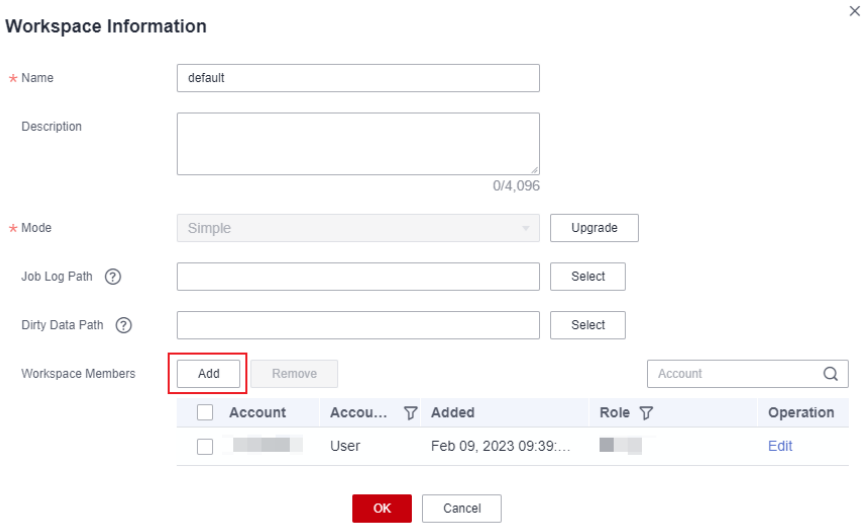
Step 2 Locate the target workspace and click **Edit**.

Figure 4-17 Editing a workspace



Step 3 In the displayed **Workspace Information** page, click **Add**.

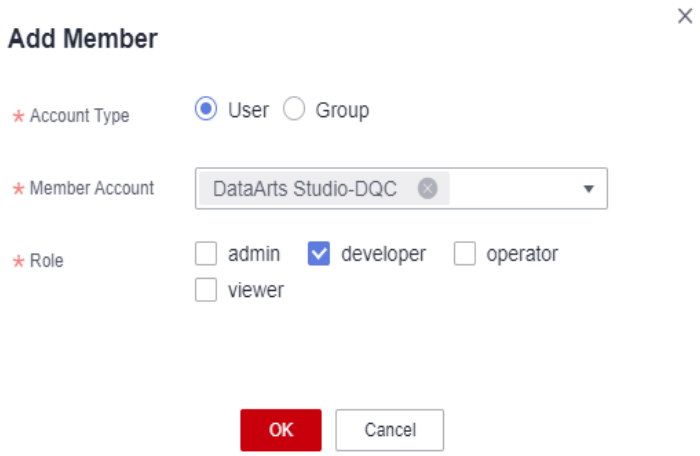
Figure 4-18 Adding a member



Step 4 Add the create IAM user to the workspace and click **OK**.

- **Account Type**: Select **Add User**.
- **Member Account**: Select the IAM user created in **Creating an IAM User and Adding It to the User Group**.
- **Role**: Select the role.

Figure 4-19 Adding a member



Step 5 After added to the workspace, the IAM user has operation permissions of DataArts Quality and view permissions of other components.

----End

Logging In to the Console and Verifying Permissions

Step 1 Log in to the Huawei Cloud console as the IAM user created in **Creating an IAM User and Adding It to the User Group** and switch to the region where the user has been granted permissions.

Step 2 Choose **Service List** > **DataArts Studio**. Locate a DataArts Studio instance and click **Access**. Check whether the workspace list is displayed.

Step 3 Access the workspace to which the current user has been added, access each component (such as Management Center and DataArts Quality), and check whether you can perform operations in DataArts Quality.

----End

5 How Do I View the Number of Table Rows and Database Size?

In the data governance process, we often need to collect statistics on the number of rows in a data table or the size of a database. The number of rows in a data table can be obtained using SQL commands or data quality jobs. The database size can be viewed in the data catalog component. For details, see the following operation guide:

- **Counting the Number of Rows in a Data Table**
- **statistics database**

Counting the Number of Rows in a Data Table

For different types of data sources, DataArts Studio provides multiple methods to view the number of rows in a table.

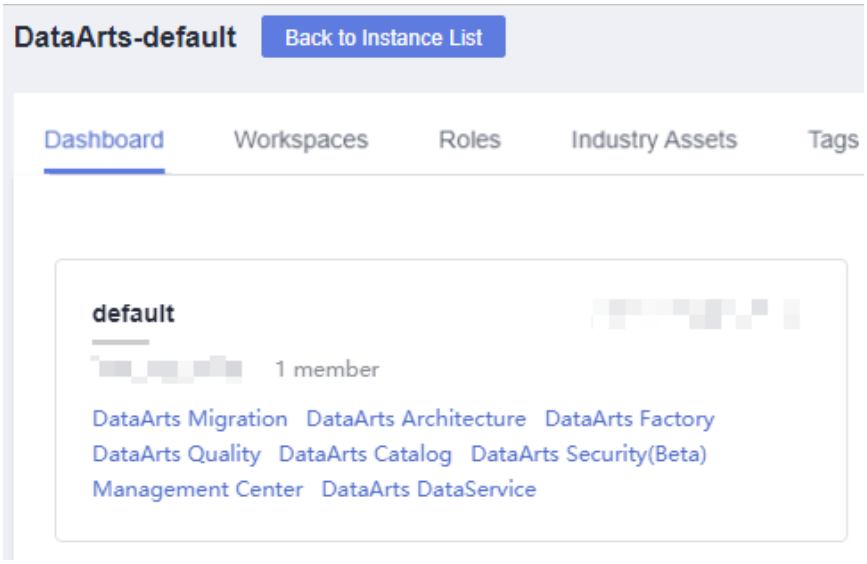
- For data sources such as DWS, DLI, RDS, MRS Presto, MRS Hive, MRS Spark, , you can run the SQL script of the corresponding type in the data development component to view the number of table rows.
  ```
  select count(*) from tablename
  ```
- For data sources such as DWS, DLI, RDS, MRS Hive, MRS Spark , and Oracle, you can execute quality jobs in the data quality component to view the number of table rows.

For other data sources, you are advised to view the number of table rows on the data source side by referring to the operation description on the data source side.

Obtain the number of rows in a table using a DataArts Studio data quality job. This method can collect statistics on the number of rows in multiple tables in the same database at the same time.

Step 1 On the DataArts Studio console, locate an instance and click **Access**. On the displayed page, locate a workspace and click **DataArts Quality**.

Figure 5-1 DataArts Quality

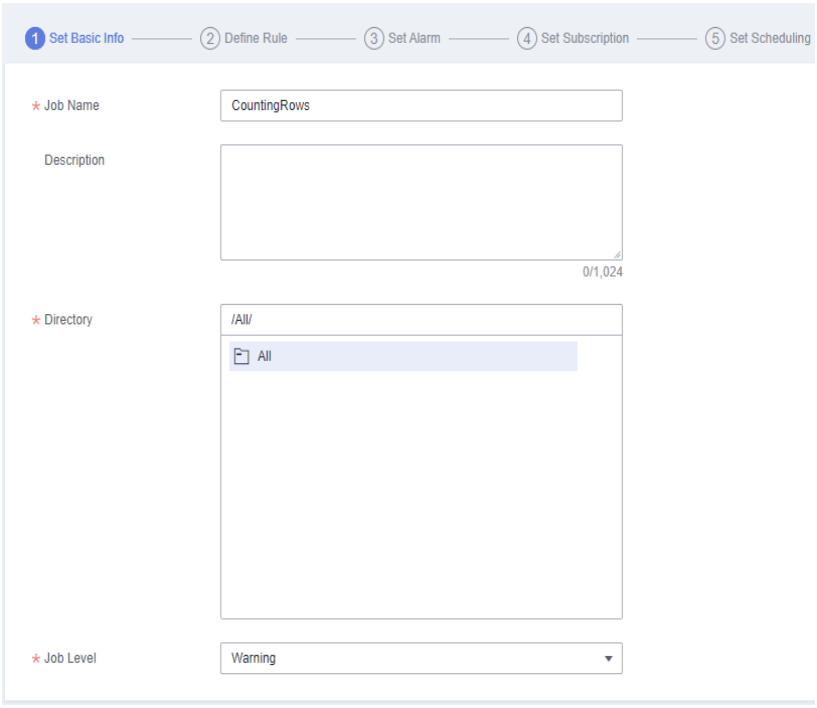


Step 2 Click Quality Job. The quality job list is displayed.

Step 3 Click Create. The quality job basic configuration page is displayed, as shown in the following figure.

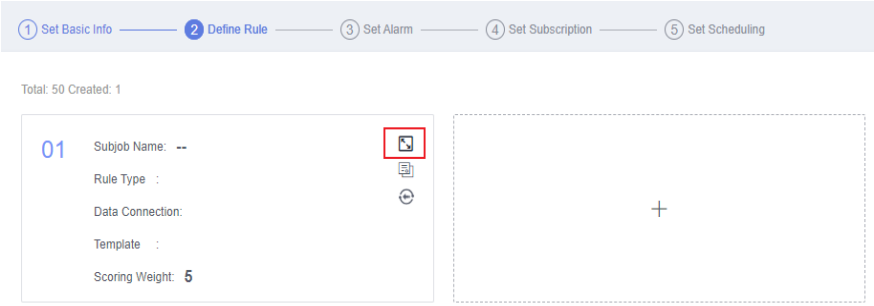
- Job name.
- Directory: Select the directory where the job is stored.
- Job Level: Retain the default value.

Figure 5-2 Basic Configuration



Step 4 Click **Next** to go to the **Define Rule** page. Click the Open icon of the subjob. The subjob configuration page is displayed.

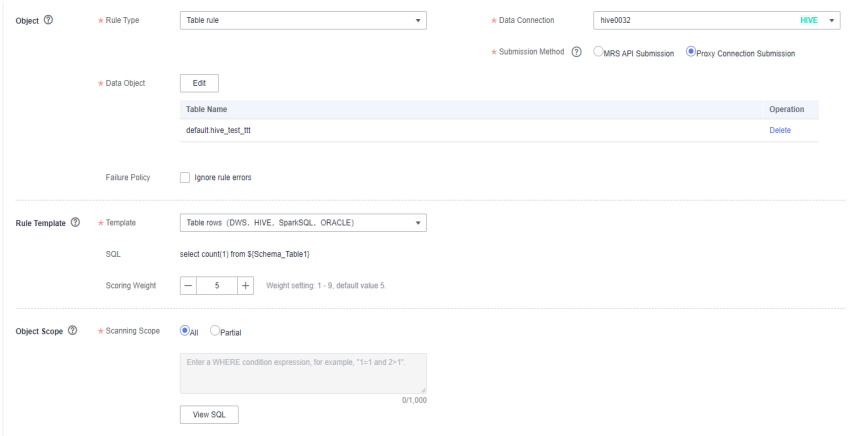
Figure 5-3 Go to the subjob configuration page.



Step 5 Click the Open icon of a subjob. On the subjob configuration page that is displayed, configure rule information.

- Basic Information: This parameter is optional. Retain the default value.
- Object
  - **Rule Type**: Select **Threshold Rule**.
  - Data Connection: Select the data source connection created in the management center.
  - Data Object: Select the data table whose statistics are to be collected.
  - Retain the default values for other parameters.
- Template
  - Template Name: Select Table Rows (DWS, HIVE, SparkSQL, ORACLE).
  - Retain the default values for other parameters.
- Select **All** for **Scanning Scope**.
- Alarm Condition: This parameter is optional. Retain the default value.

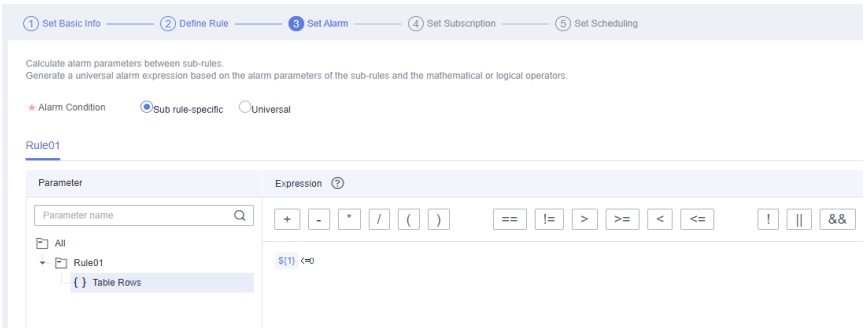
Figure 5-4 Subtask Configuration



Step 6 Click **Next** to go to the **Set Alarm Parameters** page.

Set Alarm Condition to Sub-rule Alarm Condition. The expression can be customized. In this example, the expression can be set to ${1}<=0, indicating that an alarm is triggered when the total number of rows is less than or equal to 0.

Figure 5-5 Alarm Triggering Condition



Step 7 Click **Next** to go to the **Configure Report** page.

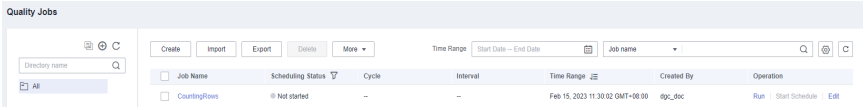
If the notification status is enabled, you need to select a notification type and a topic. There are two notification types: Trigger Alarm and Running Success. You can select a notification type based on the actual service scenario.

Step 8 Click **Next** to go to the **Configure Report** page.

There are two scheduling modes: One-time scheduling and Periodic scheduling. Select One-time scheduling for one-time statistics.

Step 9 Click Submit. The quality job list page is displayed.

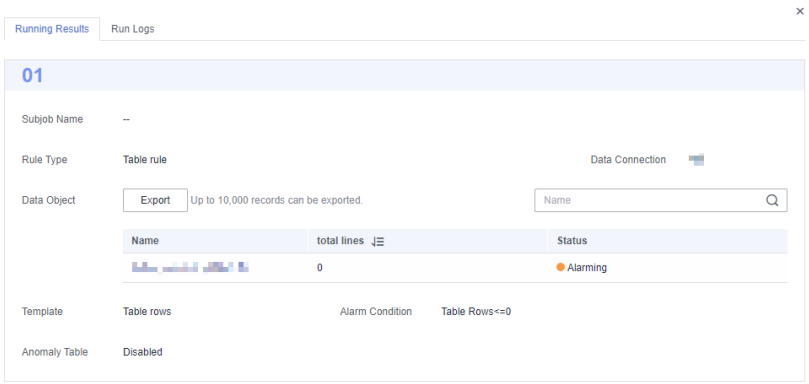
Figure 5-6 Quality job list



Step 10 In the **Operation** column of the CountingRows job, click **Run** to generate the instance corresponding to the job.

Step 11 Click O&M Management to go to the job instance list page and find the corresponding job instance. After the instance running is complete, click Result & Log. On the Running Result tab page, you can view the running result of the quality job, that is, the total number of rows in the table to be collected.

Figure 5-7 Viewing the Total Number of Rows in a Table



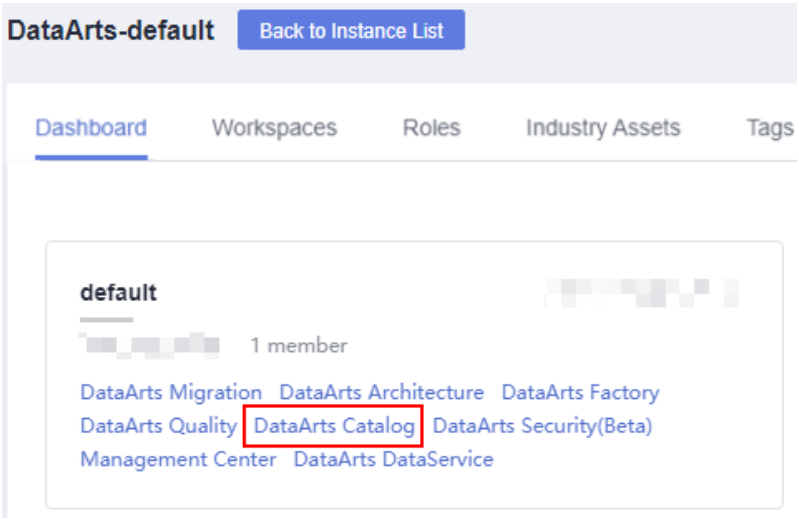
----End

statistics database

You can directly view the database size in the data catalog component.

Step 1 On the DataArts Studio console, locate an instance and click **Access**. On the displayed page, locate a workspace and click **DataArts Catalog**.

Figure 5-8 DataArts Catalog



Step 2 On the Asset Overview tab page of the Overview page, click the number of databases under Technical Assets to view the number and size of tables in each database.

Figure 5-9 This API is used to query technical assets.

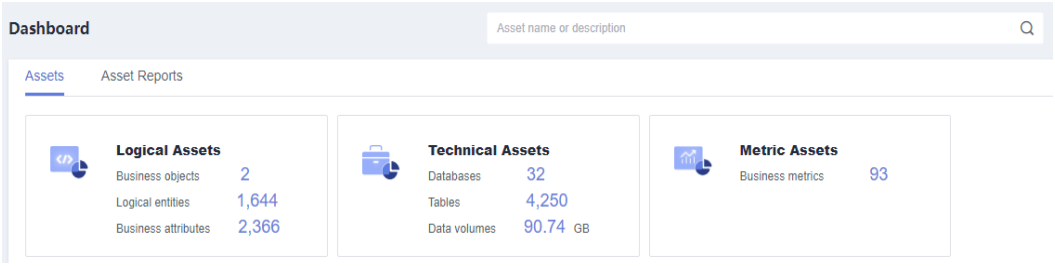
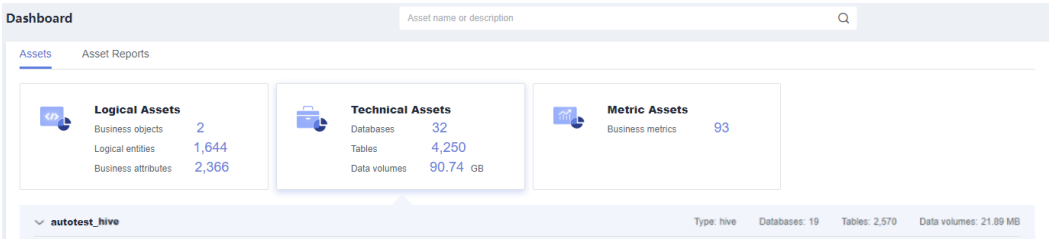


Figure 5-10 View the volume of imported data.



----End

6 Comparing Data Before and After Data Migration Using DataArts Quality

Data comparison checks data consistency before and after migration or processing.

This section describes how to use the DataArts Quality module of DataArts Studio to check consistency of the data before and after it is migrated from DWS to an MRS Hive partitioned table.

Prerequisites

- You have created a DWS cluster, which can communicate with the DataArts Studio instance. You have the permission to access the KMS key.
- You have created an MRS cluster which can communicate with the DataArts Studio instance.
- You have created a CDM cluster. For details, see buy**Buying a DataArts Studio Incremental Package**.

Creating a Data Migration Link

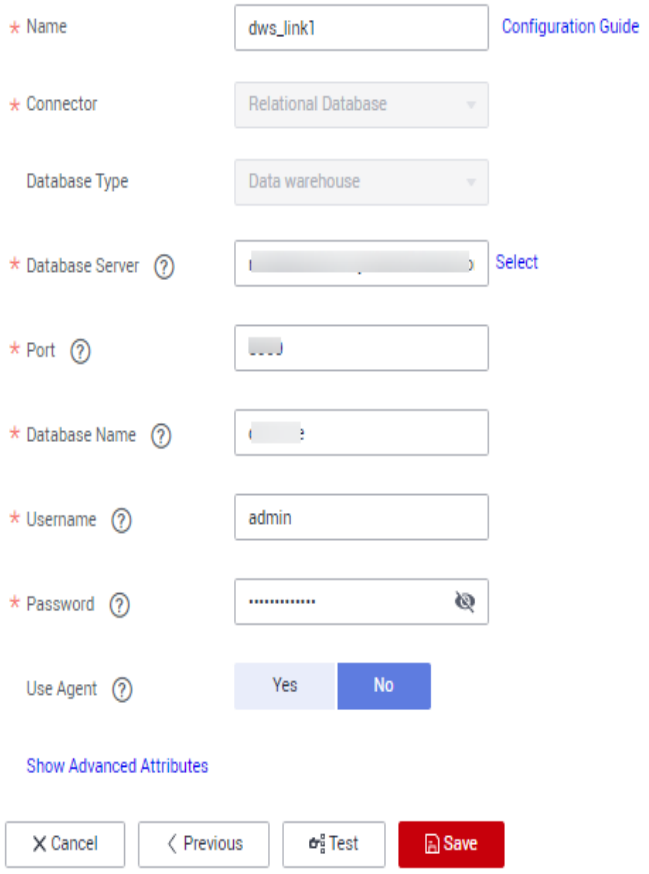
Step 1 Log in to the DataArts Studio console, locate a workspace, and click **DataArts Migration**.

Step 2 On the **Cluster Management** page, locate the prepared CDM cluster and click **Job Management** in the **Operation** column.

Figure 6-1 Job Management page

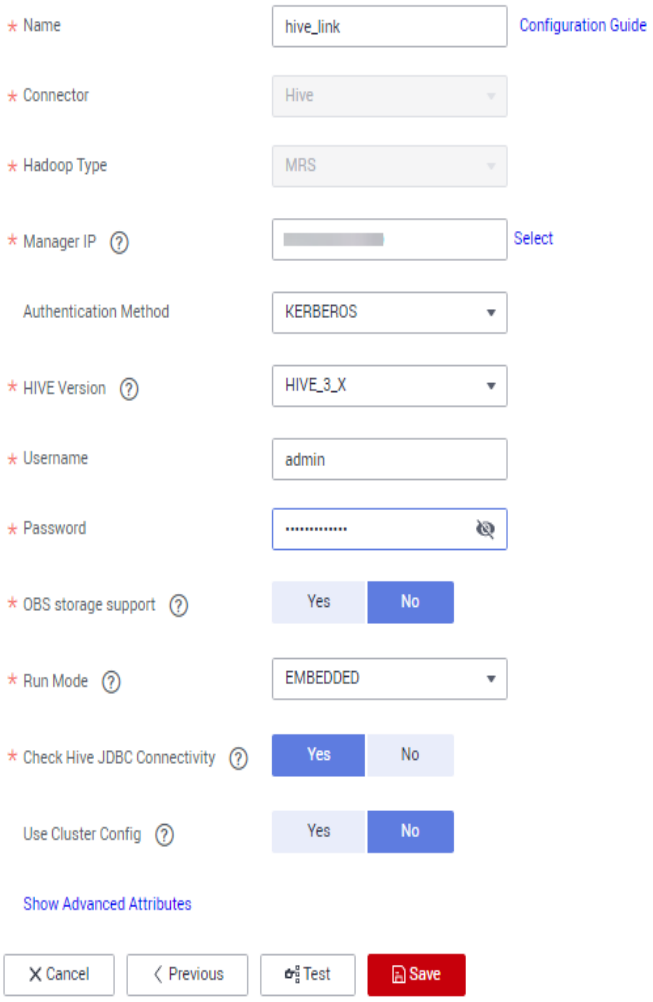
Step 3 Click the **Links** tab and **Create Link** to create a DWS link. For details about the parameters, see **Link to DWS**.

Figure 6-2 Creating a DWS link



Step 4 Create an MRS Hive link. or details about the parameters, see **Link to Hive**.

Figure 6-3 Creating an MRS Hive link



----End

Creating and Executing a Data Migration Job

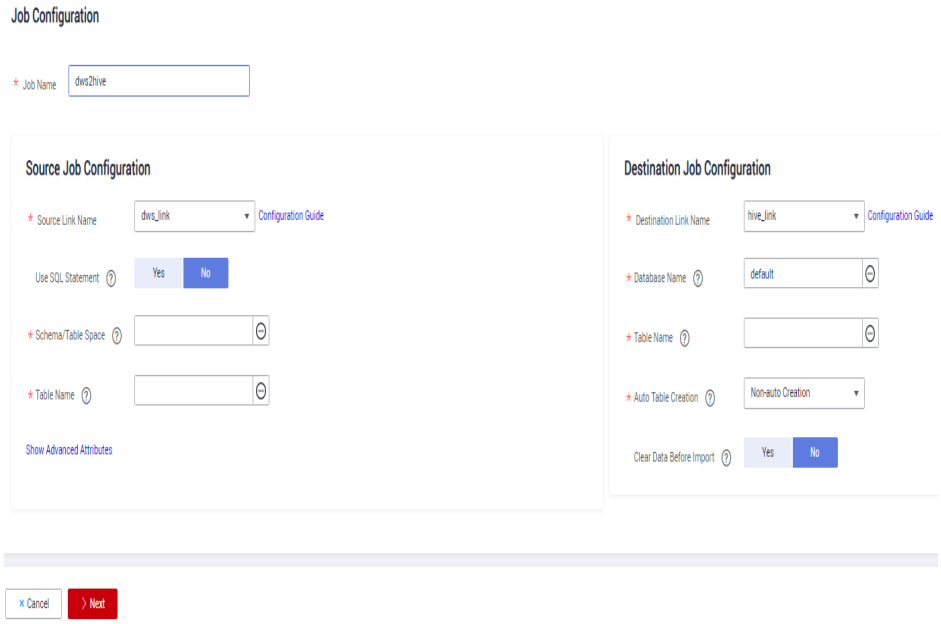
Step 1 Log in to the DataArts Studio console, locate a workspace, and click **DataArts Migration**.

Step 2 On the **Cluster Management** page, locate the prepared CDM cluster and click **Job Management** in the **Operation** column.

Step 3 Click the **Table/File Migration** tab and then **Create Job** to create a data migration job.

Step 4 Select the DWS link for **Source Link Name** and MRS Hive link for **Destination Link Name**, and set required parameters. For details about the parameters, see **From DWS** and **To Hive**.

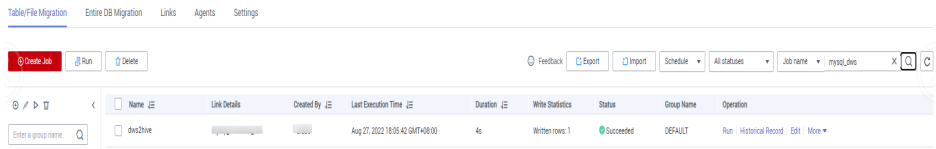
Figure 6-4 Job configuration



Step 5 Configure the field mapping and task and click **Save and Execute** to execute the CDM job.

Step 6 On the **Table/File Migration** job list, view the job status.

Figure 6-5 Viewing the job status



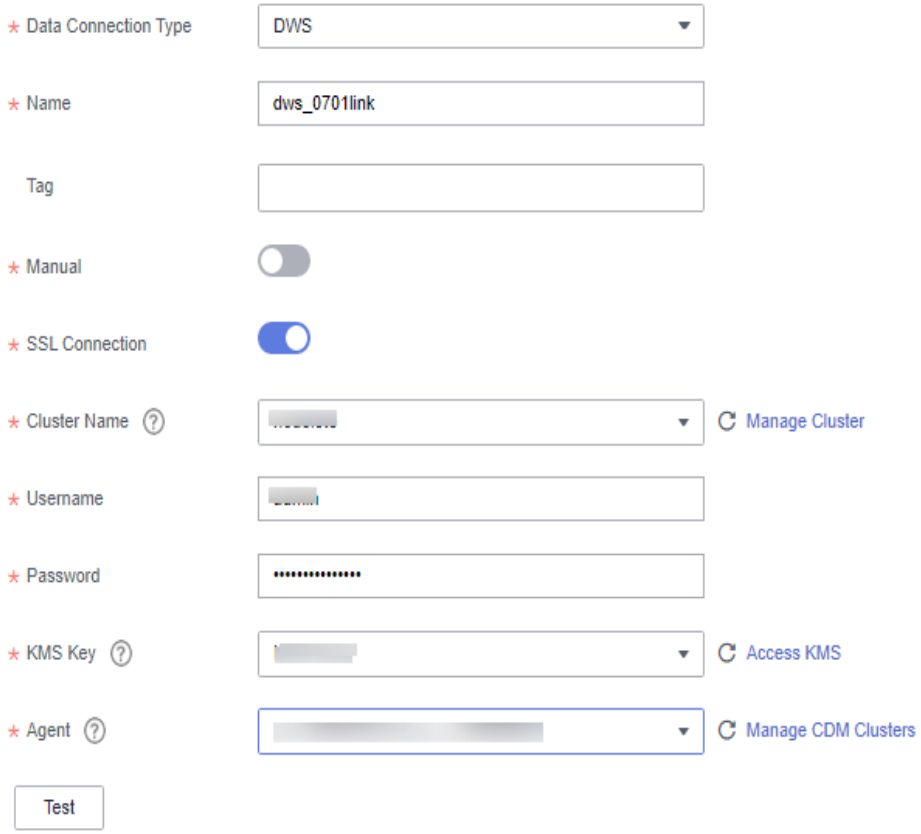
----End

Creating a Data Connection

Step 1 Log in to the DataArts Studio console, locate a workspace, and click **Management Center**.

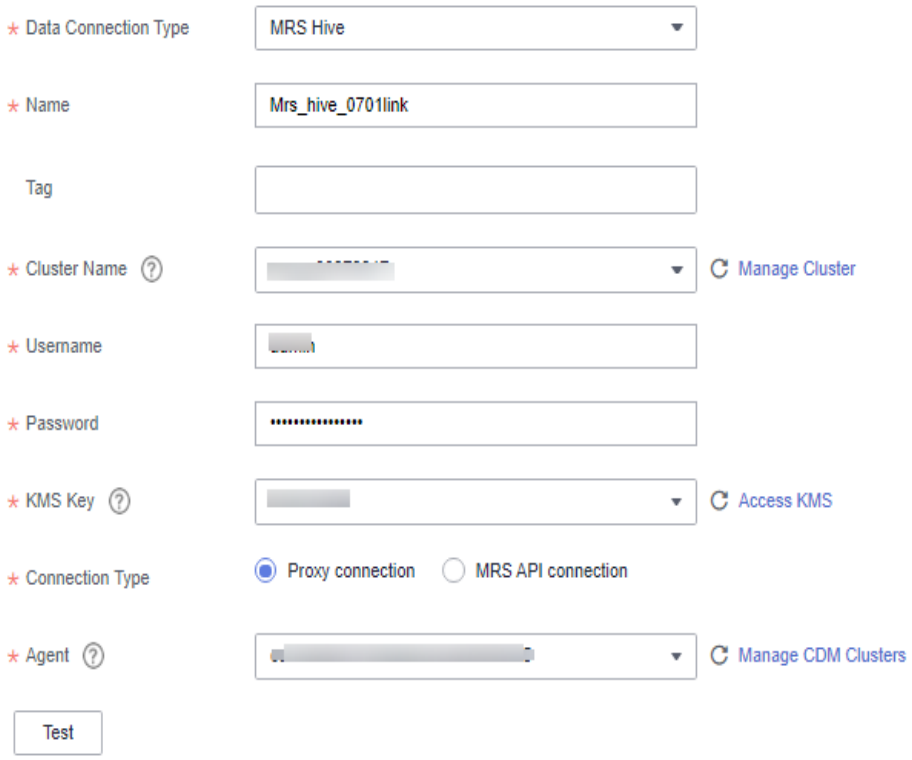
Step 2 Click **Create Data Connection** to create a DWS data connection. For details about the parameters, see **Creating a DWS Connection**.

Figure 6-6 Creating a DWS data connection



Step 3 Create an MRS Hive data connection. For details about the parameters, see **Creating an MRS Hive Connection**.

Figure 6-7 Creating an MRS Hive data connection



----End

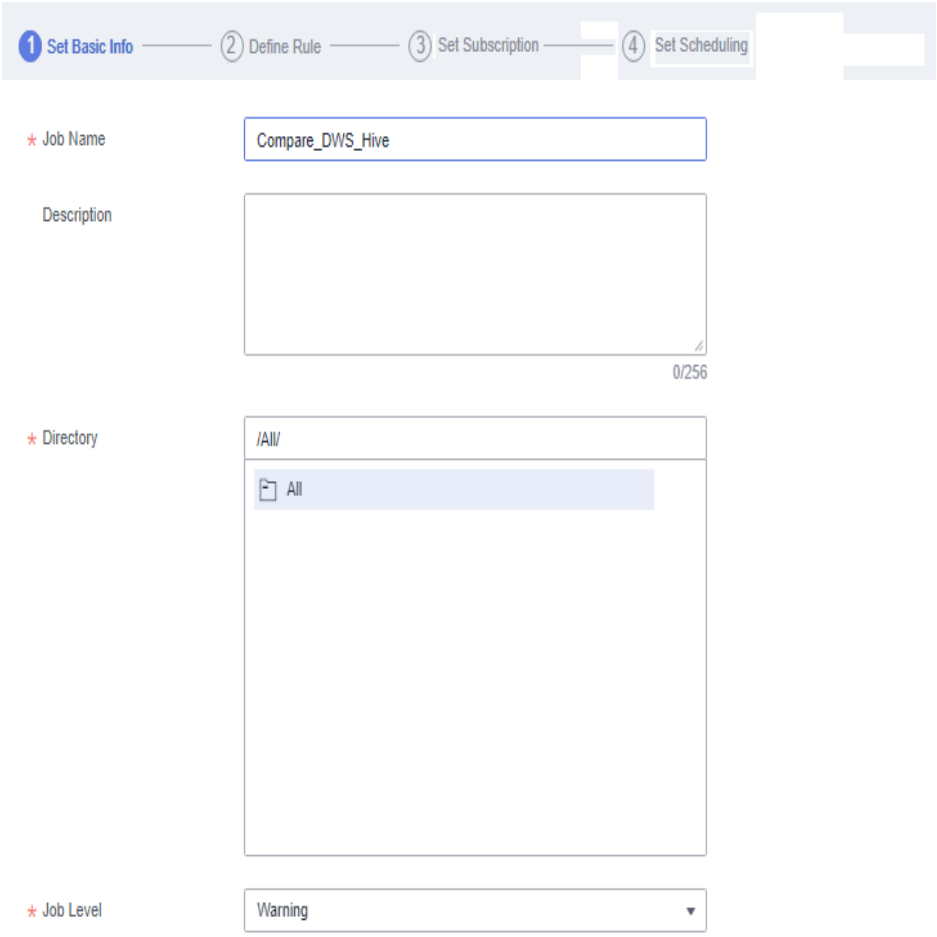
Creating a Comparison Job

Step 1 Log in to the DataArts Studio console, locate a workspace, and click **DataArts Quality**.

Step 2 In the left navigation pane, choose **Quality Monitoring** > **Comparison Jobs**.

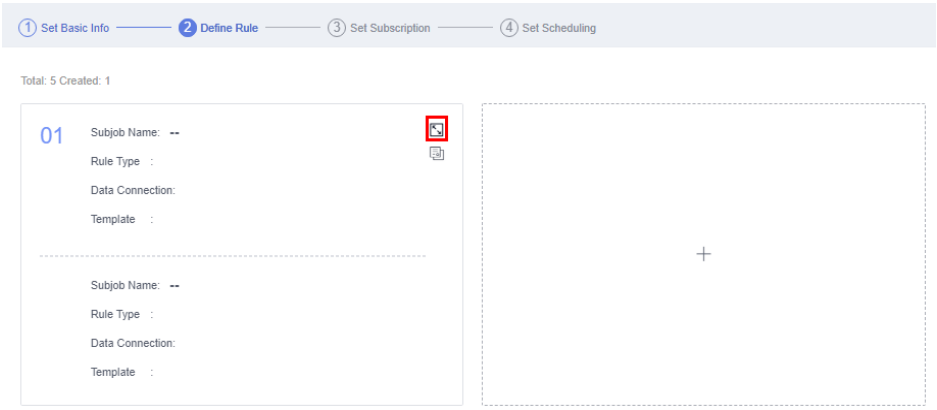
Step 3 Click **Create**. On the **Create Comparison Job** page, set basic information.

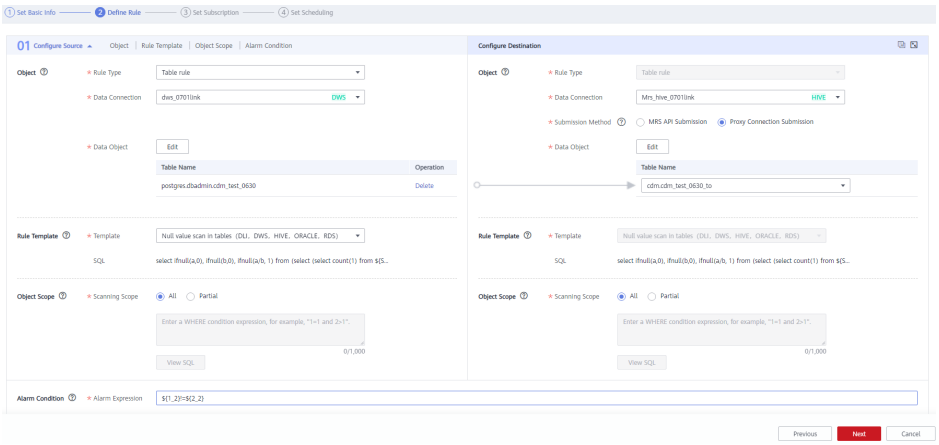
Figure 6-8 Setting basic information for the comparison job



Step 4 Click **Next** to go to the **Define Rule** page. Click , configure the comparison rule, select the data tables before and after the migration, and configure the alarm rule.

Figure 6-9 Configuring the comparison rule



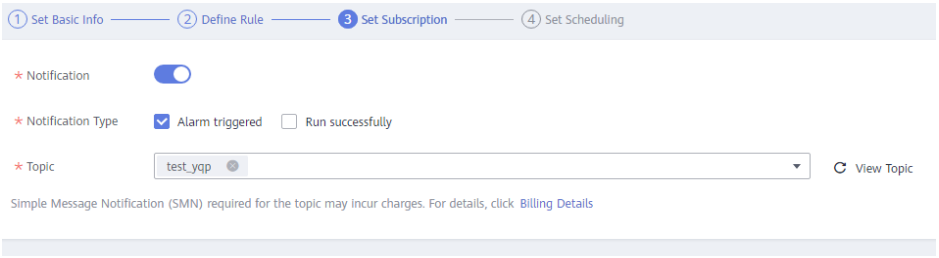


NOTE

- Configure the source and destination information separately.
- When configuring **Alarm Condition**, **${1_1}** indicates the number of rows in the source table, and **${2_1}** indicates the number of rows in the destination table. In the preceding figure, the alarm condition **${1_1}!=${2_1}** indicates that an alarm is generated when the number of rows in the source table is inconsistent with that in the destination table.

Step 5 Click **Next** and set subscription information.

Figure 6-10 Setting subscription information

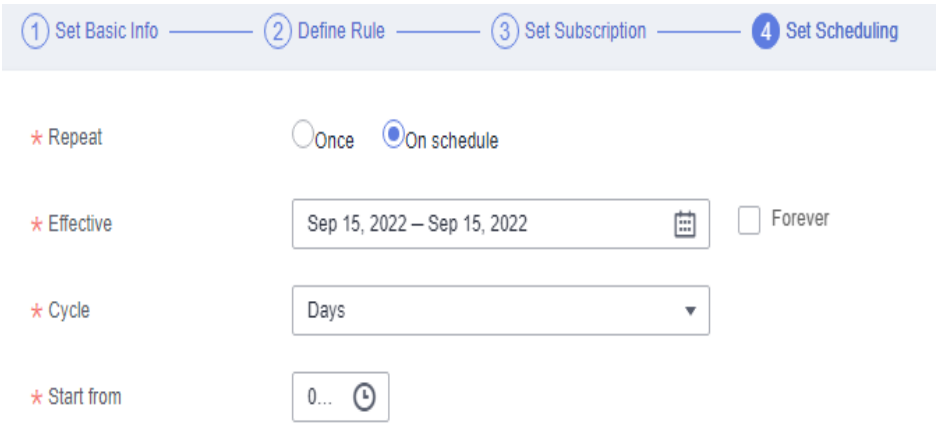


NOTE

If you enable notification, **Alarm triggered** indicates that a notification is sent to the SMN topic when an alarm is generated for the job, and **Run successfully** indicates that a notification is sent to the SMN topic when no alarm is generated for the job.

Step 6 Click **Next** and set scheduling parameters.

Figure 6-11 Setting scheduling parameters



NOTE

Once indicates that the job needs to be manually executed, and **On schedule** indicates that the job is executed automatically based on your configuration. The configuration in the preceding figure indicates that the job is automatically executed every 15 minutes.

Step 7 Click **Submit** to create the comparison job.

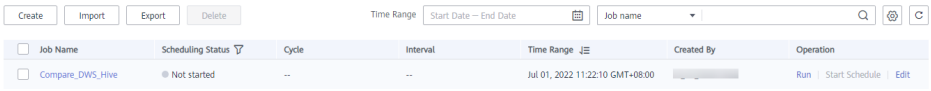
----End

Executing the Comparison Job and Viewing the Result Analysis

Step 1 In the left navigation pane, choose **Quality Monitoring** > **Comparison Jobs**.

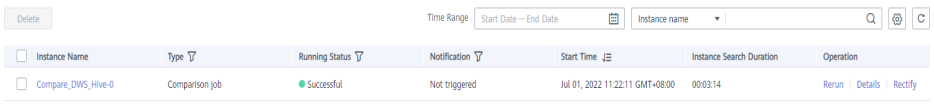
Step 2 Locate the created comparison job and click **Run** in the **Operation** column.

Figure 6-12 Running the comparison job



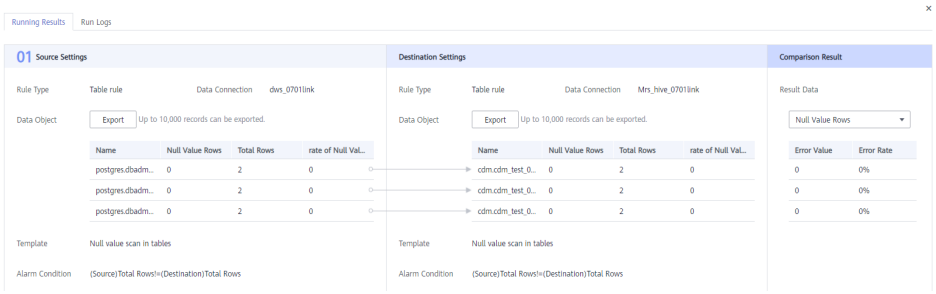
Step 3 In the left navigation pane, choose **Quality Monitoring** > **O&M**.

Figure 6-13 O&M page



Step 4 After the job is executed, click **Details** in the **Operation** column. If the source and destination tables have the same number of rows, the migration is successful.

Figure 6-14 Viewing the running result



NOTE

- In the running result, the left pane displays the execution result of the rule for source table rows, and the right pane displays the execution result of the rule for destination table rows.
- The error rate indicates the difference between the number of rows of the source and destination tables. If the error rate is 0, the source and destination tables have the same number of rows.

----End

7 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

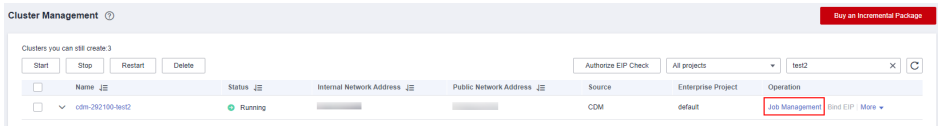
A CDM incremental package is available.

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 7-1 Cluster Management

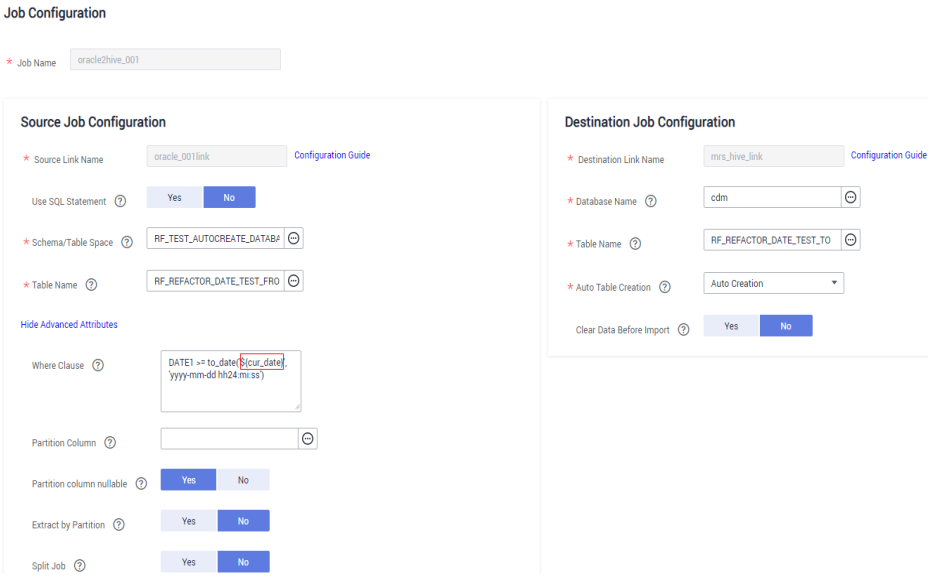


Step 3 Click the **Links** tab and then **Create Link** to create an Oracle link and an MRS Hive link. For details, see **Link to an Oracle Database** and **Link to Hive**.

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 7-2 Creating a job



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

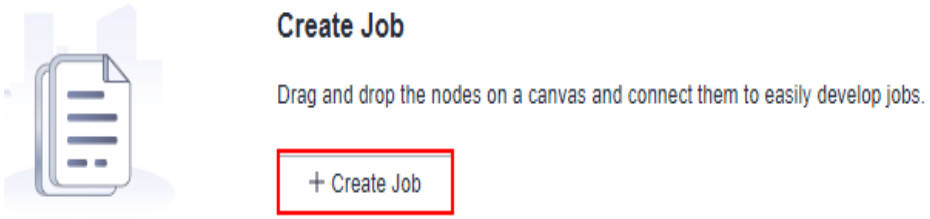
Creating and Executing a Data Development Job

Step 1 On the DataArts Studio console, locate a workspace and click **DataArts Factory**.

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 7-3 Create Job



Step 4 In the displayed dialog box, configure job parameters and click **OK**.

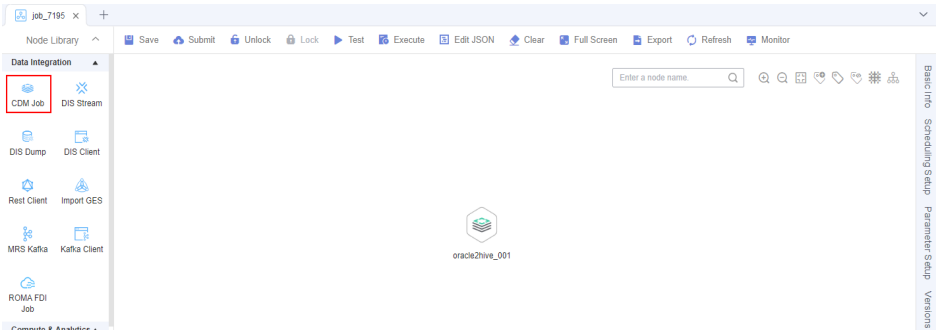
Table 7-1 Job parameters

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.
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.

Parameter	Description
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.

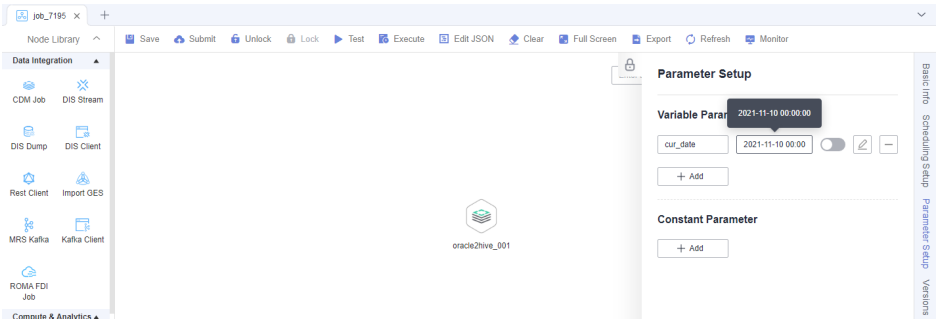
Step 5 Add a CDM Job node in the data development job and associate the node with the created CDM job.

Figure 7-4 Associating the CDM Job node with the created CDM job



Step 6 Configure the parameter to be transferred to the CDM job.

Figure 7-5 Configuring the parameter to be transferred



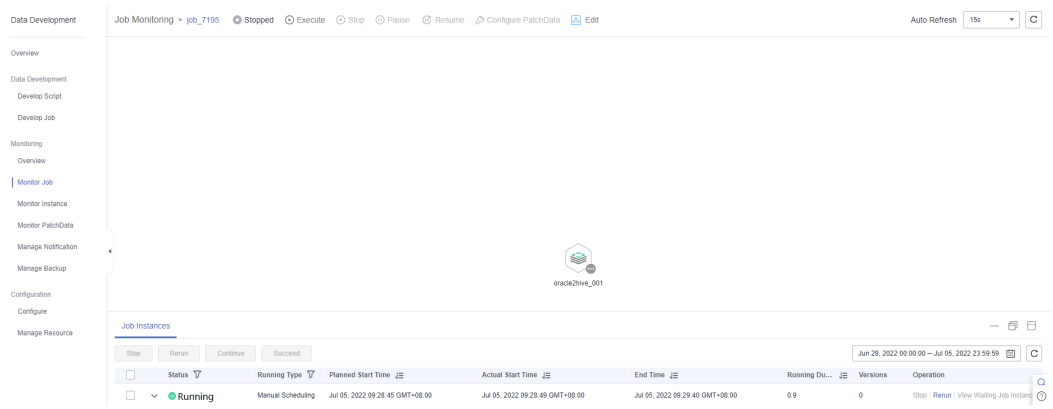
NOTE

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 7-6 Viewing the execution result



----End

8 Enabling Incremental Data Migration Through DataArts Factory

The DataArts Factory module of DataArts Studio is a one-stop, collaborative big data development platform. You can enable incremental data migration through online script editing in DataArts Factory and periodic scheduling of CDM jobs.

This section describes how to use DataArts Factory together with CDM to migrate incremental data from GaussDB(DWS) to OBS.

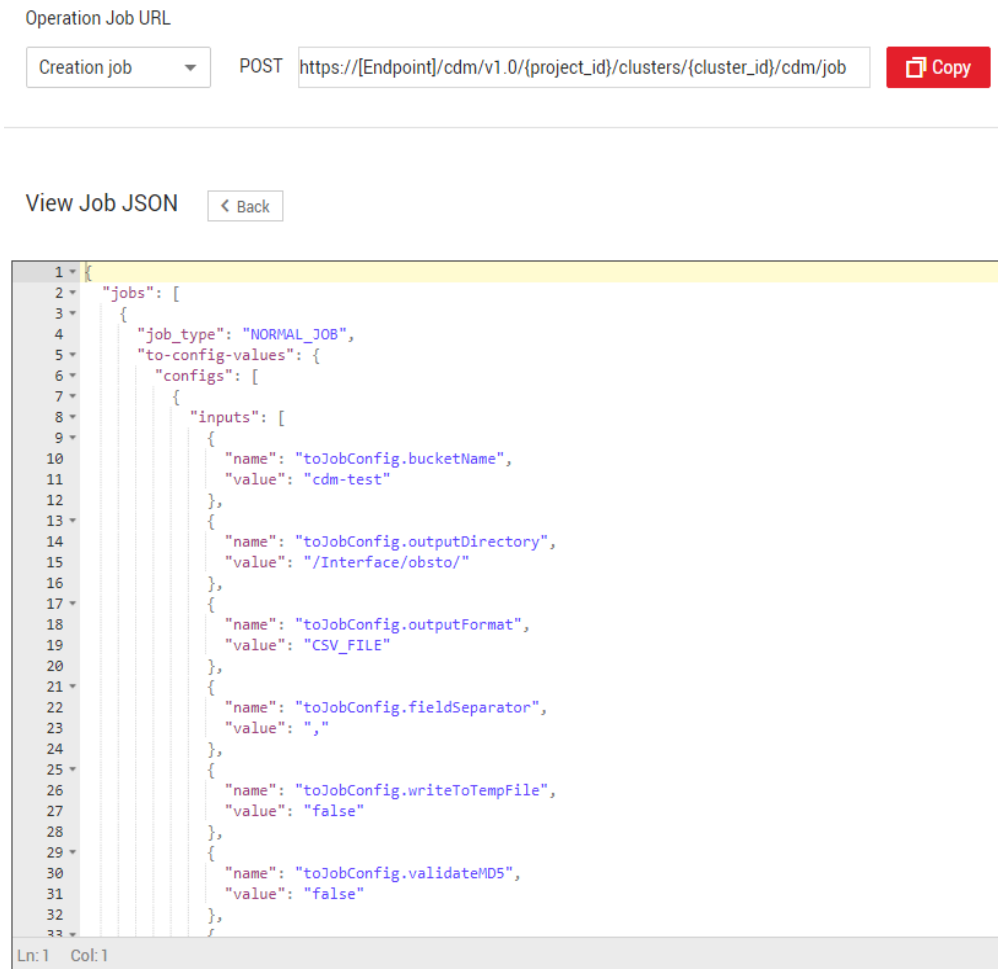
1. **Obtaining the CDM Job JSON**
2. **Modifying JSON**
3. **Creating a Job in DataArts Factory**

Obtaining the CDM Job JSON

1. On the CDM console, create a table/file migration job from GaussDB(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 8-1 Viewing job JSON



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"
      },
      {
```



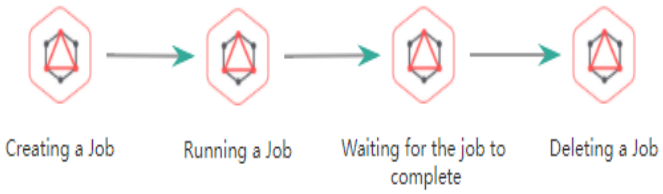
```
        "inputs": [],
        "name": "transformConfig"
      },
      {
        "inputs": [],
        "name": "smnConfig"
      },
      {
        "inputs": [],
        "name": "retryJobConfig"
      }
    ]
  },
  "to-connector-name": "obs-connector",
  "from-link-name": "dws_link",
  "name": "dws2obs-${currentTime}"
 }
]
}
```

Creating a Job in DataArts Factory

1. On the DataArts Factory console, create a data development job with Rest Client nodes shown in **Figure 8-2**. 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 8-2 DataArts Factory job

2. Configure the **CreatingJob** node.

   DataArts Factory uses a Rest Client node to call a RESTful API to create a CDM migration job. Configure the properties of the Rest Client node.

   a. **Node Name**: Enter a custom name, for example, **CreatingJob**. Note that the CDM job is only used as a node in the DataArts Factory 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.

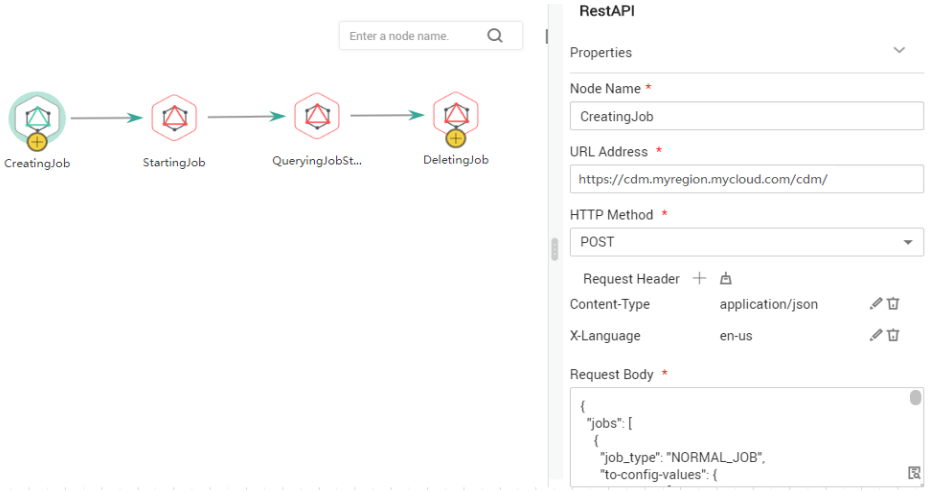
   c. **HTTP Method**: Enter **POST**.

   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 8-3 Properties of the node for creating the CDM job



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.

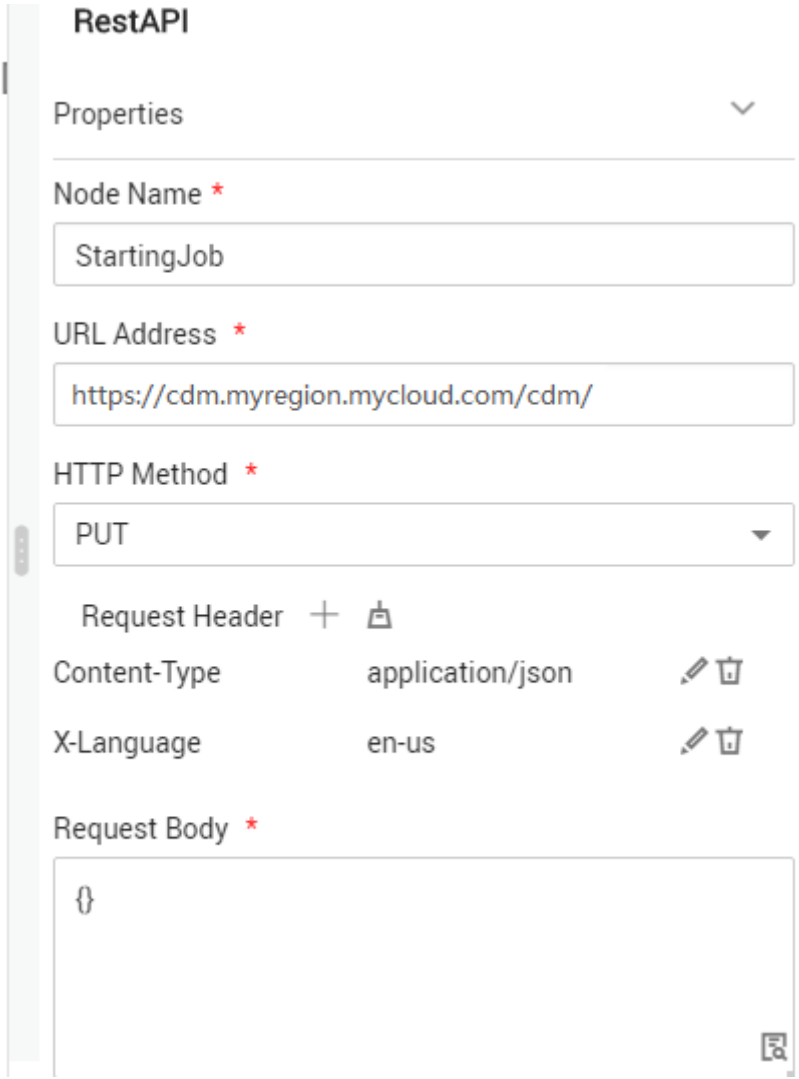
   b. **URL Address**: Keep the values of **project_id** and **cluster_id** consistent with those in **2**. Set the job name to **dws2obs-${currentTime}**. The format is https://{*Endpoint*}/cdm/v1.0/{*project_id*}/clusters/{*cluster_id*}/cdm/job/{*job_name*}/start.

   c. **HTTP Method**: Enter **PUT**.

   d. **Request Header**:

      - Content-Type = application/json
      - X-Language = en-us

Figure 8-4 Properties of the node for running the CDM job



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. DataArts Factory 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. DataArts Factory 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**.

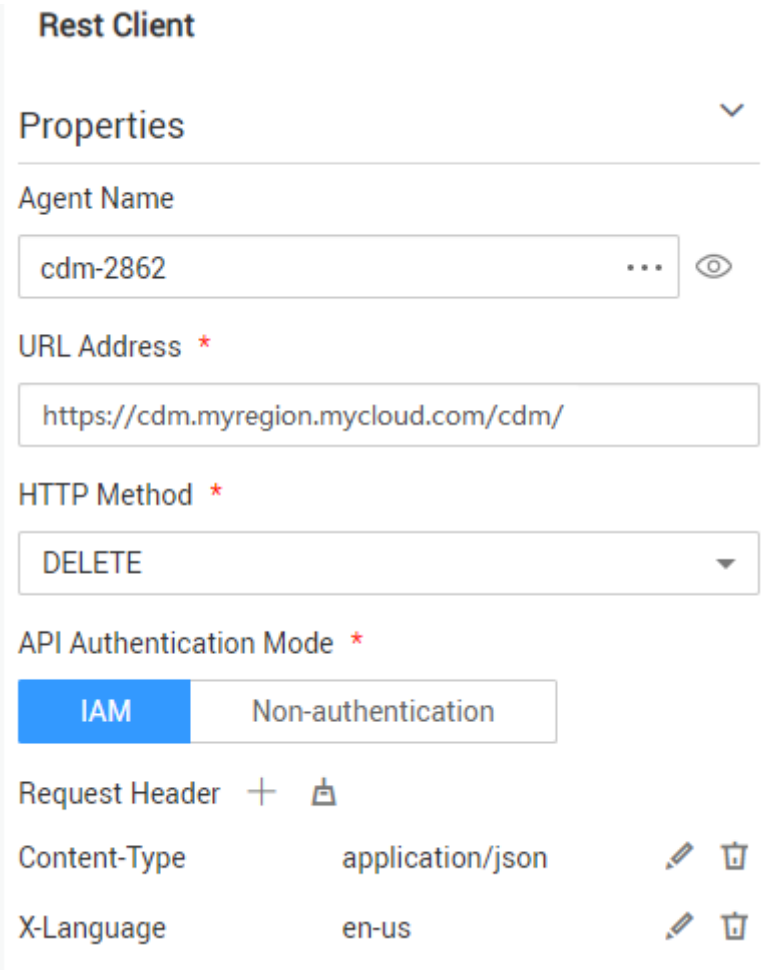
   b. **URL Address**: The format is https://{*Endpoint*}/cdm/v1.0/{*project_id*}/clusters/{*cluster_id*}/cdm/job/{*job_name*}. Keep the values of **project_id** and **cluster_id** consistent with those in **2**. Set the job name to **dws2obs-${currentTime}**.

   c. **HTTP Method**: Enter **DELETE**.

   d. **Request Header**:
      - Content-Type = application/json
      - X-Language = en-us

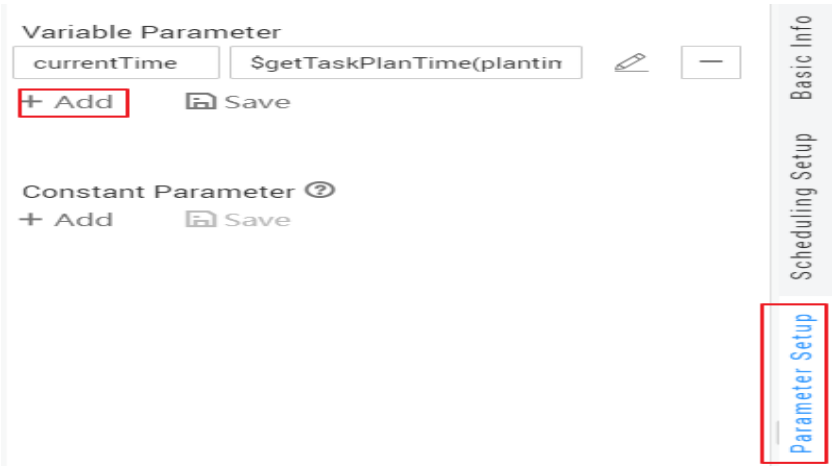
   e. Retain default values for other parameters.

Figure 8-5 Properties of the node for deleting the CDM job



6. To perform computing operations after the migration is complete, you can add various computing nodes.
7. Configure job parameters in DataArts Factory.
   a. Configure the job parameters shown in **Figure 8-6**.
      - startTime = $getTaskPlanTime(plantime,@@yyyyMMddHHmmss@@,-24*60*60)
      - currentTime = $getTaskPlanTime(plantime,@@yyyyMMdd-HHmm@@,0)

Figure 8-6 Configuring job parameters in DataArts Factory



b. After saving the job, choose **Scheduling Configuration** > **Periodic Scheduling** and set the scheduling period to one day.

   In this way, DataArts Factory works with CDM to migrate data generated on the previous day every day.

9 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.

Table 9-1 Parameters for a link to a MySQL database

Parameter	Description	Example Value
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_link
Database Server	IP address or domain name of the database to connect Click **Select** next to the text box and select a MySQL DB instance in the displayed dialog box.	192.168.0.1
Port Number	Port of the database to connect	3306
Database	Name of the database to connect	dbname
Username	Username used for accessing the database This account must have the permissions required to read and write data tables and metadata.	cdm
Password	Password of the user	-
Use Local API	(Optional) Whether to use the local API of the database for acceleration. When you create a MySQL link, CDM automatically enables the **local_infile** system variable of the MySQL database to enable the LOAD DATA function, which accelerates data import to the MySQL database. If this parameter is enabled, the date type that does not meet the format requirements will be stored as 0000-00-00. For details, visit the official MySQL website. If CDM fails to enable this function, contact the database administrator to enable the **local_infile** system variable. Alternatively, set **Use Local API** to **No** to disable API acceleration. If data is imported to RDS for MySQL, the LOAD DATA function is disabled by default. In such a case, you need to modify the parameter group of the MySQL instance and set **local_infile** to **ON** to enable the LOAD DATA function. **NOTE** If **local_infile** on RDS is uneditable, it is the default parameter group. You need to create a parameter group, modify its values, and apply it to the RDS for MySQL instance. For details, see the *Relational Database Service User Guide*.	Yes
Use Agent	Whether to extract data from the data source through an agent	Yes
Agent	Click **Select** and select the created agent.	-

Parameter	Description	Example Value
local_infile Character Set	When using local_infile to import data to MySQL, you can configure the encoding format.	utf8
Driver Version	Select a driver version that adapts to the database type.	-
Fetch Size	(Optional) Displayed when you click **Show Advanced Attributes**. Number of rows obtained by each request. Set this parameter based on the data source and the job's data size. If the value is either too large or too small, the job may run for a long time.	1000
Commit Size	(Optional) Displayed when you click **Show Advanced Attributes**. Number of records submitted each time. Set this parameter based on the data destination and the job's data size. If the value is either too large or too small, the job may run for a long time.	-

Parameter	Description	Example Value
Link Attributes	(Optional) Click **Add** to add the JDBC connector attributes of multiple specified data sources. For details, see the JDBC connector document of the corresponding database. The following are some examples: ● **connectTimeout=360000** and **socketTimeout=360000**: When a large amount of data needs to be migrated or the entire table is retrieved using query statements, the migration fails due to connection timeout. In this case, you can customize the connection timeout interval (ms) and socket timeout interval (ms) to prevent failures caused by timeout. ● **tinyInt1isBit=false** or **mysql.bool.type.transform=false**: By default, **tinyInt1isBit** is **true**, indicating that **TINYINT(1)** is processed as a bit, that is, **Types.BOOLEAN**, and **1** or **0** is read as **true** or **false**. As a result, the migration fails. In this case, you can set **tinyInt1isBit** to **false** to avoid migration failures. ● **useCursorFetch=false**: By default, **useCursorFetch** is enabled, indicating that the JDBC connector communicates with relational databases using a binary protocol. Some third-party systems may have compatibility issues, causing migration time conversion errors. In this case, you can disable this function. Open-source MySQL databases support the **useCursorFetch** parameter, and you do not need to set this parameter. ● **allowPublicKeyRetrieval=true**: By default, public key retrieval is disabled for MySQL databases. If TLS is unavailable and an RSA public key is used for encryption, connection to an MySQL database may fail. In this case, you can enable public key retrieval to avoid connection failures.	sslmode=require
Reference Sign	(Optional) Delimiter between the names of the referenced tables or columns. For details, see the product documentation of the corresponding database.	'

Parameter	Description	Example Value
Batch Size	Number of rows written each time. It should be less than **Commit Size**. When the number of rows written reaches the value of **Commit Size**, the rows will be committed to the database.	100

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 9-1 Configuring the MRS Hive link

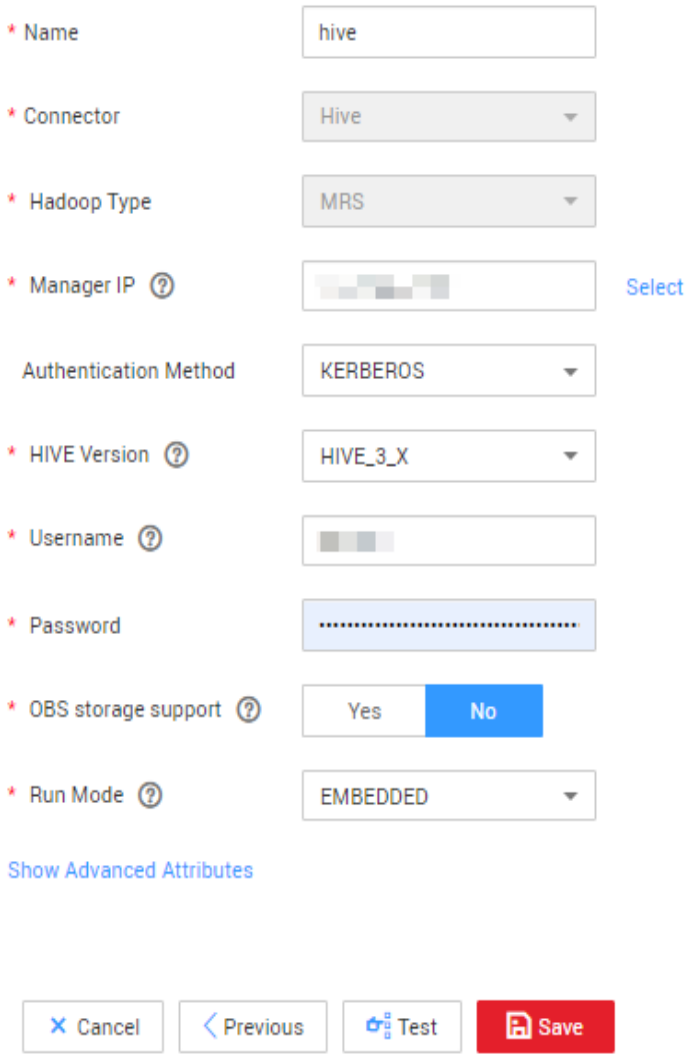


Table 9-2 MRS Hive link parameters

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
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

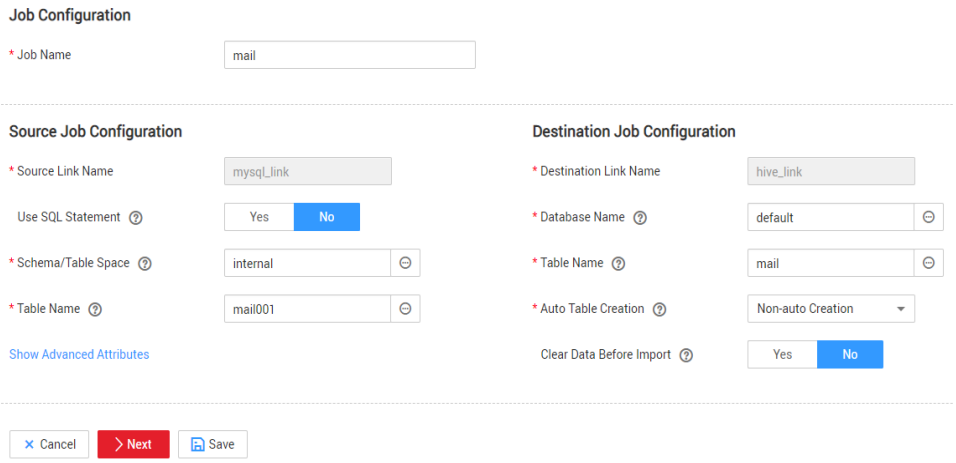
Parameter	Remarks	Example
Run Mode	This parameter is used only when the Hive version is **HIVE_3_X**. Possible values are: • **EMBEDDED**: The link instance runs with CDM. This mode delivers better performance. • **STANDALONE**: The link instance runs in an independent process. If CDM needs to connect 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

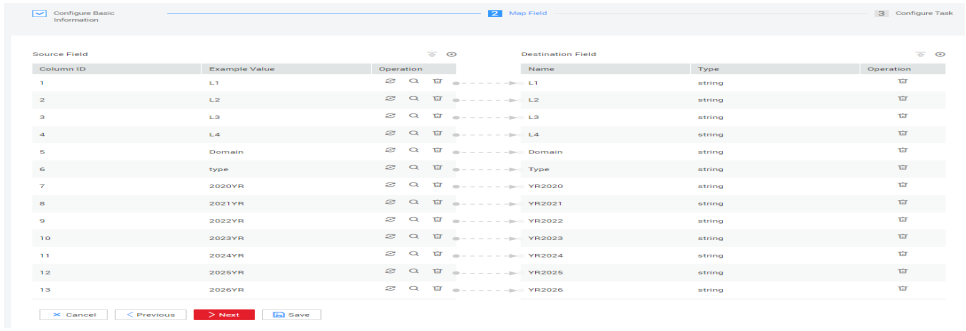
----End

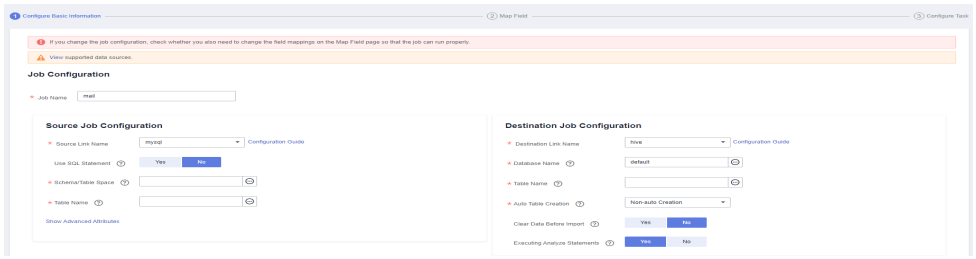
Creating a Sample Job

Step 1 In the **Operation** column, click **Job Management**.

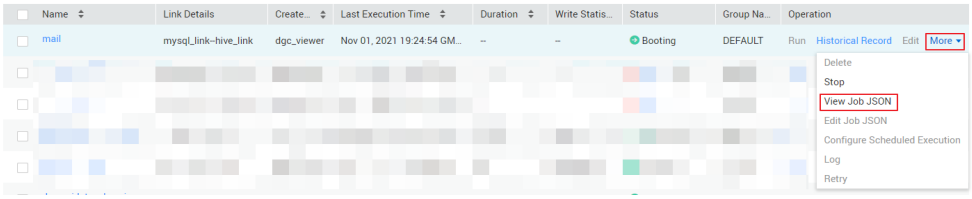
Step 2 Click the **Table/File Migration** tab and then **Create Job** to create a job for migrating data from the first MySQL subtable **mail001** to the MRS Hive table **mail**.

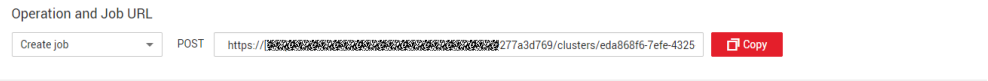






Step 3 After the sample job is created, view and copy the job JSON for subsequent configuration of data development jobs.



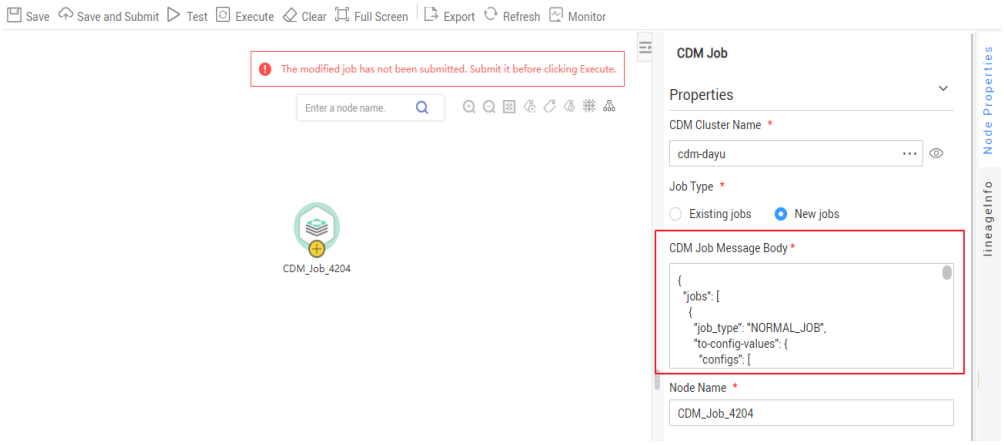


----End

Creating a Data Development Job

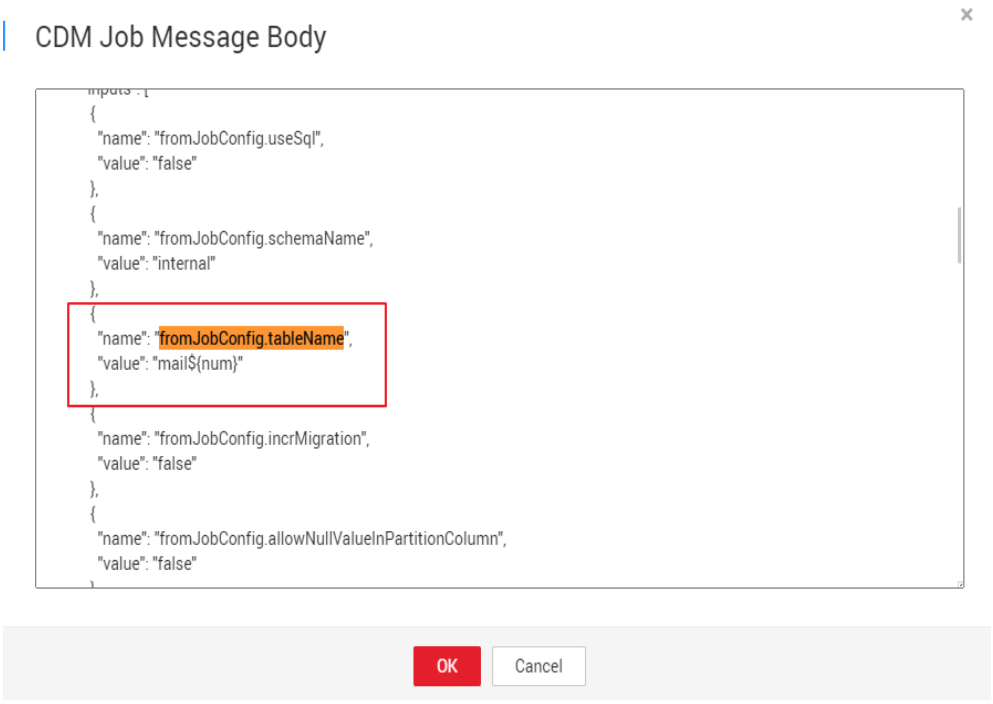
Step 1 Locate a workspace and click **DataArts Factory**.

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**.



Step 3 Edit the CDM job message body.

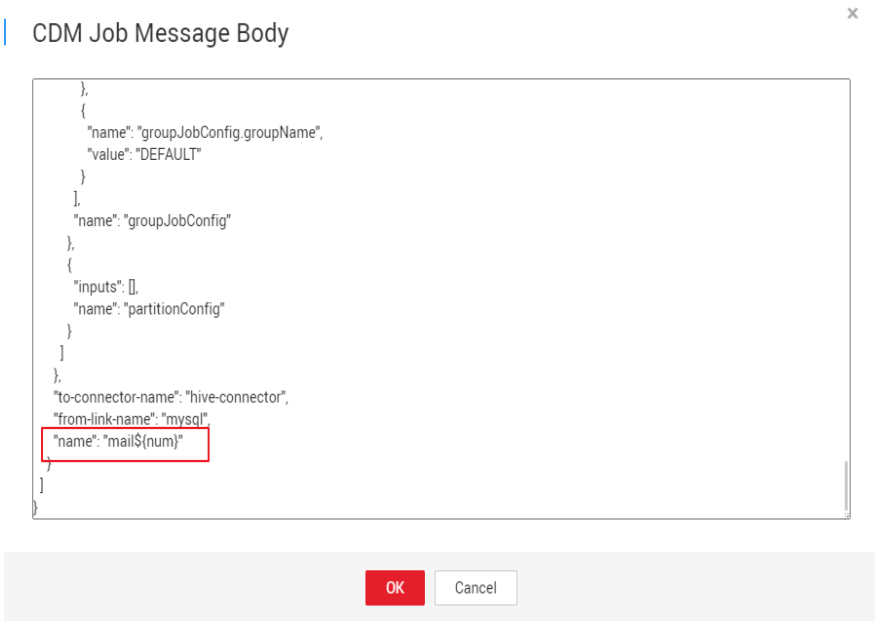
1. Since there are three source tables **mail001**, **mail002**, and **mail003**, you need to set **fromJobConfig.tableName** to **mail${num}** in the JSON file of the job. The following figure shows the parameters for creating a main job.



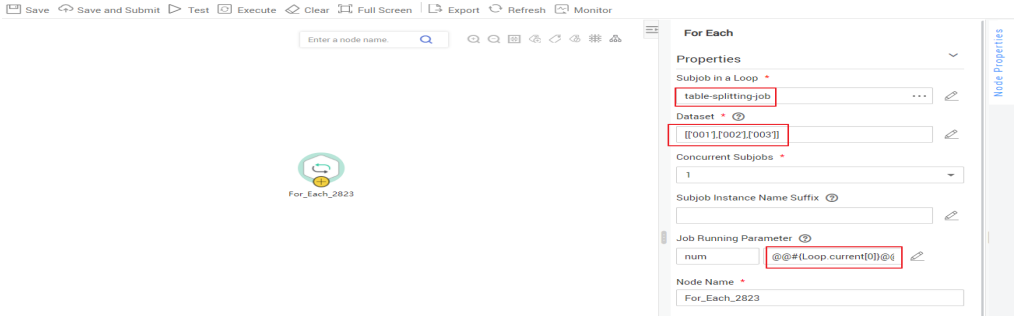
2. The name of each data migration job must be unique. Therefore, you need to change the value of **name** in the JSON file to **mail${num}** to create multiple CDM jobs. The following figure shows the parameters for creating a main job.

   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.



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.

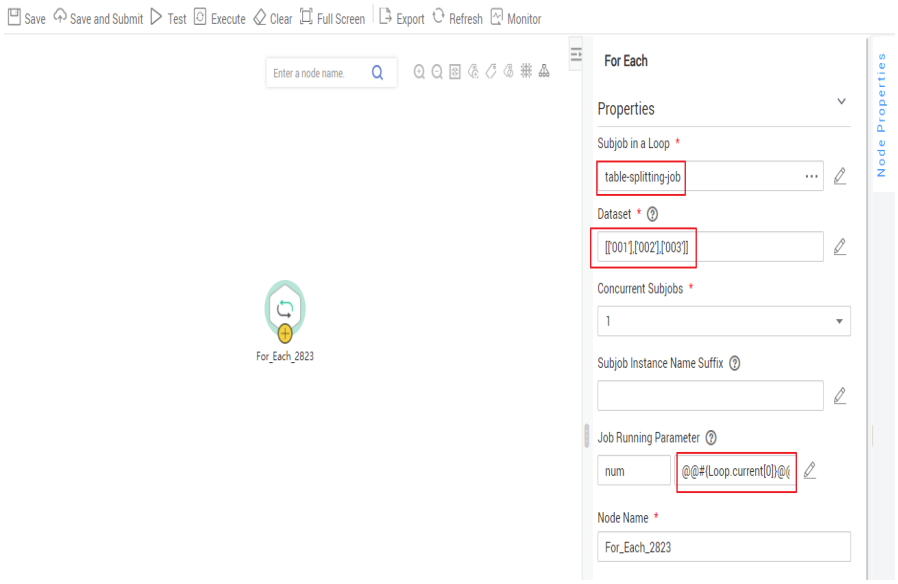
The key configurations are as follows:

- **Subjob in a Loop**: Select **table**.
- **Dataset**: Enter **[['001'],['002'],['003']]**.
- **Subjob Parameter Name**: Enter **@@#{Loop.current[0]}@@**.

  NOTE

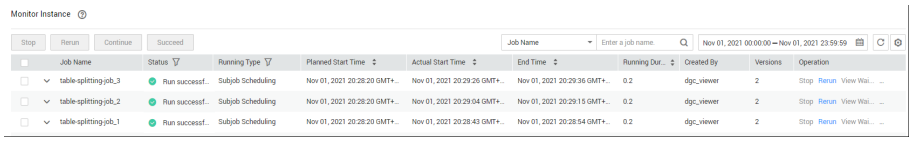
  Add **@@** to the EL expression of the subjob parameter. If **@@** is not added, dataset 001 will be identified as 1. As a result, the source table name does not exist.

The following figure shows the parameters for creating a main job.



Click **Save and Submit** to save the main job.

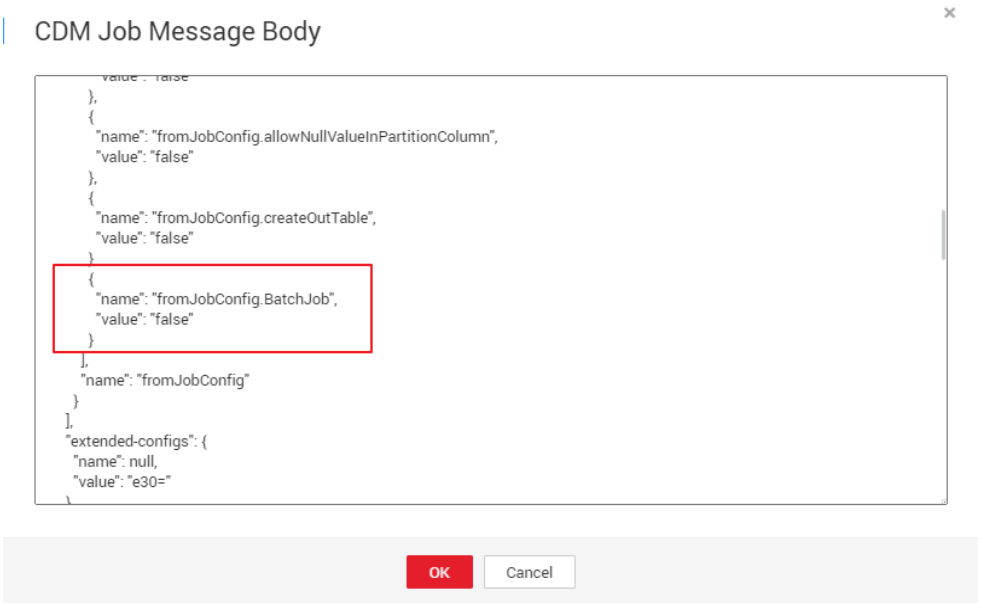
Step 6 After the main job and subjobs are created, test and run the main job to check whether it is successfully created. If the job is successfully executed, the CDM subjobs are successfully created and executed.



----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.



- 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.

10 Building Graph Data Based on MRS Hive Tables and Automatically Importing the Data to GES

10.1 Scenario

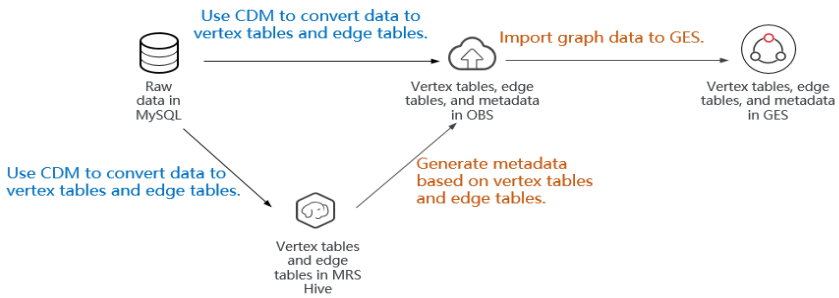
Graph Engine Service (GES) is a service for querying and analyzing graph-structure data based on relationships. It is specifically suited for scenarios requiring analysis of rich relationship data, including social relationship analysis, marketing and recommendations, public opinions and social listening, information communication, and anti-fraud.

In DataArts Studio, you can convert raw data tables into standard vertex data sets and edge data sets based on GES data import requirements, and periodically import graph data (vertex data sets, edge data sets, and metadata) to GES using the automatic metadata generation function, perform visualized graphical analysis on the latest data in GES.

Scenario Description

In this case, DataArts Studio is used to convert the user and rating data of a movie website stored in MySQL database to standard vertex and edge data sets, and to synchronize the data to OBS and MRS Hive. Then, the Import GES node automatically generates metadata and imports the graph data to GES.

Figure 10-1 Business scenarios



Note that the GES graph data formats include vertex data set, edge data set, and metadata. If your raw data is not in these formats, convert it to these formats.

- Vertex data sets store vertex data.
- Edge data sets store edge data.
- Metadata is used to describe the formats of data in vertex and edge data sets.

For details about GES concepts and graph data, see **Graph Data Formats**.

Constraints

When metadata is automatically generated using the Import GES node, note the following constraints:

1. You can only select a vertex table and a edge table that use a single label. If you select a vertex table or a edge table that has multiple labels, the generated metadata may be missing.
2. The metadata XML file is generated after you click **Create**. If the structure of the vertex table and edge table changes during subsequent job scheduling, the metadata XML file will not be updated automatically. In this case, you need to open the **New** dialog box and click **Create** again to generate a new metadata XML file.
3. In the generated metadata XML file, the value of **Cardinality** (data composite type) in **Property** is **single** and cannot be changed.
4. You can generate metadata XML files for multiple pairs of vertex tables and edge tables at a time. However, only one table can be selected for the **Edge Data Set** and **Vertex Data Set** parameters of the Import GES node. If there are multiple pairs of vertex tables and edge tables, you are advised to create metadata XML files on multiple Import GES nodes. In this way, you can ensure that each piece of metadata corresponds to each pair of vertex tables and edge tables during the import of graph data.

10.2 Making Preparations

Preparing the Environment

- If you use DataArts Studio for the first time, register a a Huawei account, buy a DataArts Studio instance, and create a workspace by following the instructions provided in **Preparations**. Then you can go to the created workspace and start using DataArts Studio.
- Create an MRS cluster that contains the Hive component on the MRS console. Metadata can be generated from the vertex and edge data sets in the cluster. When creating the MRS cluster, ensure that its network parameter settings (including the region, VPC, subnet, and security group) are consistent with those of the CDM cluster in the DataArts Studio instance so that the MRS cluster can communicate with the CDM cluster through the internal network. Otherwise, you need to manually enable the communication between the MRS cluster and the CDM cluster. In addition, ensure that the two clusters use the same enterprise project.

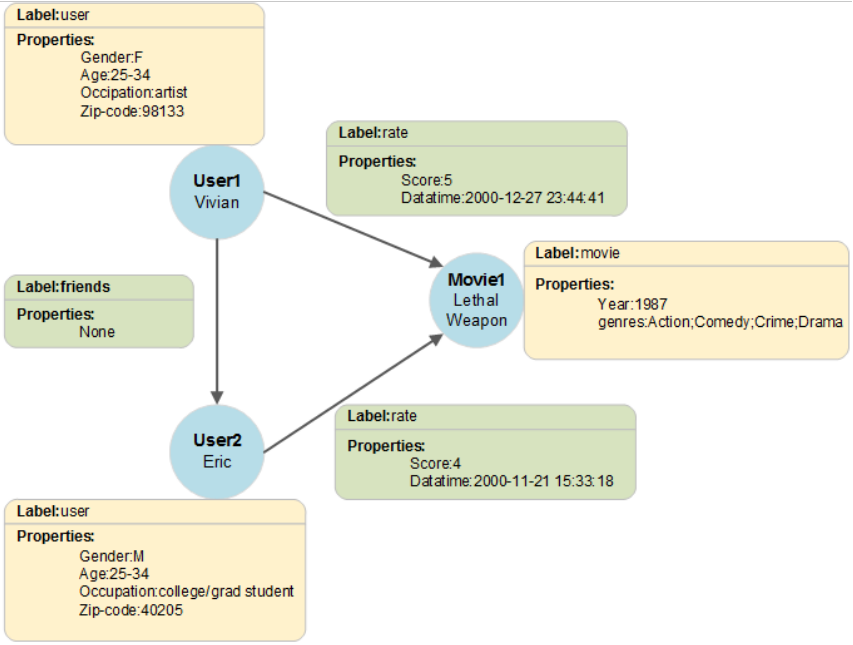
> **NOTE**
>
> During the creation of an MRS cluster, a security group is automatically created. You are advised to create an MRS security cluster first and then buy a DataArts Studio instance, selecting the same VPC and subnet as the MRS cluster and the security group (named in **mrs**_*Cluster name*_*Random character* format) that is automatically created. This ensures that the DataArts Studio instance can communicate with the MRS cluster by default.
>
> If you already have a DataArts Studio instance before creating an MRS cluster, you need to choose **Access Control** > **Security Groups** on the VPC console and add a rule to allow inbound traffic to the security group (named in **mrs**_*Cluster name*_*Random character* format) created by the MRS cluster. For details, see **Configuring Security Group Rules**.

- Create a MySQL DB instance on the RDS console to simulate the data source. When creating the MySQL DB instance, ensure that its network parameter settings (including the region, VPC, subnet, and security group) are consistent with those of the CDM cluster in the DataArts Studio instance so that the MySQL DB instance can communicate with the CDM cluster through the internal network. Otherwise, you need to manually enable the communication between the MySQL DB instance and the CDM cluster. In addition, ensure that the MySQL DB instance and the CDM cluster use the same enterprise project.
- Prepare an OBS bucket to store the generated metadata. The OBS bucket must be in the same region as the CDM cluster in the DataArts Studio instance, and the enterprise project of the OBS bucket must be the same as that of the CDM cluster.
- Create a graph on the GES console to import graph data for visualized graph analysis. GES must be in the same region as the CDM cluster in the DataArts Studio instance, and the enterprise project of GES must be the same as that of the CDM cluster.

Preparing Data Sources

In this practice, the raw data includes the **user table vertex_user**, **movie table vertex_movie**, **friend relationship table edge_friends**, and **movie rating table edge_rate**. **Figure 10-2** shows the relationships between them.

Figure 10-2 Graph data description



To facilitate demonstration, this practice provides some data used to simulate the original data. To integrate the source data into the cloud, you need to store the sample data in CSV files and upload them to an OBS bucket.

Step 1 Create CSV files (UTF-8 without BOM), name the files with the corresponding data table names, copy the sample data to different CSV files, and save the files.

To generate a CSV file in Windows, you can perform the following steps:

1. Use a text editor (for example, Notepad) to create a .txt document and copy the sample data to the document. Then check the total number of rows and check whether the data of rows is correctly separated. (If the sample data is copied from a PDF document, the data in a single row will be wrapped if the data is too long. In this case, you must manually adjust the data to ensure that it is in a single row.)
2. Choose **File** > **Save as**. In the displayed dialog box, set **Save as type** to **All files (*.*)**, enter the file name with the .csv suffix for **File name**, and select the UTF-8 encoding format (without BOM) to save the file in CSV format.

Step 2 Upload the CSV file to OBS.

1. Log in to the management console and choose **Storage** > **Object Storage Service** to access the OBS console.
2. Click **Create Bucket** and set parameters as prompted to create an OBS bucket named **fast-demo**.

   NOTE

   To ensure network connectivity, select the same region for OBS bucket as that for the DataArts Studio instance. If an enterprise project is required, select the enterprise project that is the same as that of the DataArts Studio instance.

   For details about how to create a bucket on the OBS console, see **Creating a Bucket** in *Object Storage Service Console Operation Guide*.

3. Upload data to OBS bucket **fast-demo**.

   For details about how to upload a file on the OBS console, see **Uploading a File** in *Object Storage Service Console Operation Guide*.

----End

This practice involves four sample data tables: **user table vertex_user**, **movie table vertex_movie**, **friend relationship table edge_friends**, and **movie rating table edge_rate**. The details are as follows:

- User table **vertex_user.csv**:

```
Vivian,F,25-34,artist,98133
Mercedes,F,Under 18,K-12 student,10562
Katherine,F,35-44,lawyer,79101
Stuart,M,25-34,programmer,30316
Jacob,M,25-34,artist,55408
Editha,F,56+,homemaker,46911
Cassandra,F,56+,artist,55113
Sarah,F,18-24,other or not specified,55105
Hayden,M,56+,academic/educator,30030
Jeffery,M,25-34,self-employed,45242
Bonnie,F,50-55,technician/engineer,19716
Serena,F,35-44,programmer,44106
Sidney,M,18-24,writer,85296
Leander,M,50-55,doctor/health care,98237
Fred,M,35-44,other or not specified,30906
Roger,M,45-49,technician/engineer,73069
Ella,F,25-34,other or not specified,94402
Ray,M,18-24,college/grad student,90241
Eric,M,18-24,college/grad student,40205
Frances,F,56+,retired,1234
Allison,F,18-24,sales/marketing,49505
Willy,M,25-34,technician/engineer,38104
Lance,M,18-24,college/grad student,6459
June,F,25-34,other or not specified,13326
Marshal,M,50-55,scientist,7746
Max,M,35-44,executive/managerial,91107
Hardy,M,35-44,academic/educator,22181
Jordan,M,25-34,artist,8817
Reed,M,18-24,college/grad student,89146
Glendon,M,35-44,self-employed,46214
Kevin,M,56+,retired,2356
Evan,M,45-49,programmer,53718
Clark,M,56+,academic/educator,85718
Johnny,M,56+,retired,52003
Caleb,M,50-55,retired,41076
Janet,F,35-44,homemaker,61270
Sue,F,50-55,self-employed,13207
Margaret,F,45-49,academic/educator,1609
Luke,M,35-44,executive/managerial,44306
William,M,45-49,programmer,37914
Lena,F,35-44,other or not specified,42420
Solomon,M,45-49,scientist,64081-8102
Cary,M,35-44,executive/managerial,55124
Colin,M,25-34,executive/managerial,44115
Kenny,M,25-34,college/grad student,74074
Gavin,M,25-34,programmer,24060
Donald,M,35-44,programmer,95864
Wayne,M,18-24,scientist,94606
Frank,M,18-24,college/grad student,2906
Alexander,M,18-24,college/grad student,61801
Isaiah,M,25-34,other or not specified,33142
Josephine,F,25-34,college/grad student,78728
Joshua,M,35-44,executive/managerial,54016
August,M,35-44,customer service,64801
Jessie,F,18-24,clerical/admin,60640
Yvette,F,35-44,artist,94109
Albert,M,25-34,other or not specified,40515
```



```
Eugene,M,35-44,other or not specified,40504
Rachel,F,35-44,doctor/health care,33314
Constance,F,50-55,executive/managerial,10022
Larry,M,45-49,technician/engineer,2067
Mike,M,25-34,other or not specified,30606
Hank,M,50-55,programmer,44286
Daniel,M,45-49,technician/engineer,37923
Wesley,M,25-34,executive/managerial,35244
Gina,F,35-44,sales/marketing,60202
Teresa,F,45-49,academic/educator,43202
Terry,M,35-44,writer,80222
Leo,M,50-55,academic/educator,93105
Bruce,M,50-55,academic/educator,19087-3622
Terence,M,25-34,writer,14450
Alice,F,25-34,academic/educator,79928
Benjamin,M,25-34,technician/engineer,48092
Sharon,F,18-24,college/grad student,55406
Ryan,M,18-24,college/grad student,26241
Mason,M,25-34,technician/engineer,92584
Gloria,F,56+,retired,60506
Tom,M,25-34,writer,10010
Melissa,F,35-44,doctor/health care,23507
David,M,25-34,clerical/admin,19147
Alex,M,18-24,college/grad student,10013
Florence,F,35-44,academic/educator,23508
Darwin,M,45-49,customer service,98502
Michael,M,18-24,other or not specified,31211
Brown,M,25-34,executive/managerial,90210
Jimmy,M,25-34,writer,94122
Jay,M,18-24,programmer,43650
Gladys,F,18-24,programmer,5055
Denny,M,45-49,tradesman/craftsman,2557
Jack,M,50-55,other or not specified,94025
Edison,M,45-49,executive/managerial,85287-2702
Neil,M,35-44,scientist,48187
Jennifer,F,35-44,writer,75093
Caspar,M,25-34,other or not specified,3766
Mickey,M,18-24,programmer,97205
Arthur,M,25-34,executive/managerial,2139
Christine,F,25-34,academic/educator,32303
Adeline,F,Under 18,other or not specified,1036
Cody,M,18-24,college/grad student,78705
Hillary,F,35-44,executive/managerial,21117
```

- Movie table **vertex_movie.csv**:

```
American Beauty,1999,Comedy;Drama
Airplane!,1980,Comedy
Rushmore,1998,Comedy
Predator,1987,Action;Sci-Fi;Thriller
There's Something About Mary,1998,Comedy
The Shawshank Redemption,1994,Drama
Election,1999,Comedy
Clueless,1995,Comedy;Romance
The Crying Game,1992,Drama;Romance;War
Back to the Future,1985,Comedy;Sci-Fi
The Talented Mr. Ripley,1999,Drama;Mystery;Thriller
Life Is Beautiful (La vita ии bella),1997,Comedy;Drama
2001: A Space Odyssey,1968,Drama;Mystery;Sci-Fi;Thriller
Jaws,1975,Action;Horror
Jerry Maguire,1996,Drama;Romance
The Hunt for Red October,1990,Action;Thriller
Close Encounters of the Third Kind,1977,Drama;Sci-Fi
Star Wars: Episode IV - A New Hope,1977,Action;Adventure;Fantasy;Sci-Fi
Rocky,1976,Action;Drama
The Usual Suspects,1995,Crime;Thriller
A Clockwork Orange,1971,Sci-Fi
Psycho,1960,Horror;Thriller
The Godfather: Part II,1974,Action;Crime;Drama
Annie Hall,1977,Comedy;Romance
Terminator 2: Judgment Day,1991,Action;Sci-Fi;Thriller
```



```
Pleasantville,1998,Comedy
Chinatown,1974,Film-Noir;Mystery;Thriller
Independence Day (ID4),1996,Action;Sci-Fi;War
Star Wars: Episode V - The Empire Strikes Back,1980,Action;Adventure;Drama;Sci-Fi;War
Face/Off,1997,Action;Sci-Fi;Thriller
Total Recall,1990,Action;Adventure;Sci-Fi;Thriller
Blade Runner,1982,Film-Noir;Sci-Fi
The Terminator,1984,Action;Sci-Fi;Thriller
Robocop,1987,Action;Crime;Sci-Fi
The Rock,1996,Action;Adventure;Thriller
Superman,1978,Action;Adventure;Sci-Fi
The Full Monty,1997,Comedy
Raising Arizona,1987,Comedy
Lethal Weapon,1987,Action;Comedy;Crime;Drama
Platoon,1986,Drama;War
The Fifth Element,1997,Action;Sci-Fi
The Patriot,2000,Action;Drama;War
Clerks,1994,Comedy
Being John Malkovich,1999,Comedy
The Mask,1994,Comedy;Crime;Fantasy
Grosse Pointe Blank,1997,Comedy;Crime
```

- Friend relationship table **edge_friends.csv**:

```
Gloria,David
Brown,Mason
Terence,Kenny
Clark,Brown
Mickey,Janet
Mickey,Margaret
Hayden,Constance
Frank,Janet
Lena,Darwin
Leo,Jimmy
Mercedes,Gavin
Hillary,Bruce
Leo,Neil
Terence,August
Sue,Wayne
Max,Denny
Max,Josephine
Hillary,Michael
Constance,Janet
Florence,Donald
Alice,Jacob
Roger,Sidney
Margaret,Frances
Roger,Fred
Fred,Donald
Margaret,Gavin
Fred,Gavin
Rachel,Janet
Alexander,Clark
Darwin,Cassandra
Jordan,Vivian
Terry,Larry
Hardy,Kevin
Terry,Rachel
Mercedes,Marshal
Marshal,Sharon
Jeffery,Tom
Terence,Max
Katherine,Stuart
Luke,Cassandra
Michael,Arthur
Luke,Editha
Neil,Mason
Darwin,Jessie
Marshal,Alex
Hardy,Margaret
Alexander,Eric
```



```
Mercedes,Caspar
Brown,Clark
Roger,Kevin
Benjamin,Max
Jessie,Adeline
Michael,Luke
Jimmy,Gloria
Isaiah,Frances
June,Darwin
Editha,Vivian
Caspar,Cassandra
Bruce,Denny
Caspar,Jacob
Isaiah,Ella
Mason,Ryan
Mercedes,Eugene
Roger,Josephine
Wayne,Alice
Hayden,Denny
Alexander,Colin
Larry,August
Jimmy,Brown
Jacob,William
Hardy,Gladys
Jessie,Caspar
Mason,Terence
June,Jennifer
Hardy,Arthur
Alexander,Solomon
Larry,Wayne
Larry,Gavin
Ella,Ray
Ella,Eric
Alice,Janet
Larry,Willy
Isaiah,Solomon
Benjamin,Leander
Isaiah,Sue
Caspar,Jordan
Ella,Jordan
Vivian,Eric
Max,Jay
Ryan,Hank
Ella,Colin
Luke,Alexander
Luke,Joshua
Wayne,Caspar
Wayne,Denny
Editha,Marshal
Ryan,Jessie
Michael,Cassandra
Solomon,Hillary
Jordan,Josephine
```

- Movie rating table **edge_rate.csv**:

```
Vivian,Lethal Weapon,5,2000/12/27 23:44
Mercedes,Raising Arizona,4,2000/12/27 23:51
Katherine,The Rock,3,2000/12/27 20:12
Stuart,The Mask,2,2000/12/27 20:00
Jacob,Face/Off,4,2000/12/27 20:12
Editha,There's Something About Mary,5,2000/12/27 20:06
Cassandra,Superman,4,2000/12/27 20:11
Sarah,American Beauty,4,2000/12/27 20:13
Hayden,Lethal Weapon,3,2000/12/27 20:09
Jeffery,2001: A Space Odyssey,4,2000/12/23 1:48
Bonnie,A Clockwork Orange,3,2000/12/22 23:23
Serena,Lethal Weapon,4,2000/12/22 23:24
Sidney,Raising Arizona,4,2000/12/22 23:24
Leander,Clerks,5,2000/12/12 16:58
Fred,Superman,5,2000/12/18 1:17
```



```
Roger,A Clockwork Orange,5,2000/12/13 23:54
Ella,Robocop,5,2000/12/13 23:44
Ray,The Talented Mr. Ripley,3,2000/12/14 0:24
Eric,Psycho,5,2002/1/3 20:29
Frances,The Godfather: Part II,2,2000/12/10 18:45
Allison,Independence Day (ID4),3,2000/12/13 23:58
Willy,Clerks,4,2002/1/3 20:46
Lance,There's Something About Mary,5,2000/12/13 23:43
June,Superman,4,2002/1/3 20:41
Marshal,Being John Malkovich,5,2000/12/10 18:40
Max,Predator,4,2000/12/10 18:32
Hardy,Total Recall,3,2000/12/10 18:39
Jordan,American Beauty,4,2000/12/13 23:57
Reed,Lethal Weapon,1,2000/12/10 18:37
Glendon,Airplane!,4,2000/12/13 23:46
Kevin,Raising Arizona,4,2000/12/13 23:51
Evan,Jerry Maguire,1,2000/12/13 23:58
Clark,The Hunt for Red October,5,2000/12/13 23:46
Johnny,2001: A Space Odyssey,3,2000/12/14 0:16
Caleb,Clerks,4,2000/12/9 16:45
Janet,Lethal Weapon,2,2000/12/9 16:16
Sue,Close Encounters of the Third Kind,4,2000/12/9 16:14
Margaret,Star Wars: Episode IV - A New Hope,2,2000/12/9 16:04
Luke,Clueless,2,2000/12/8 19:02
William,The Terminator,2,2000/12/8 19:03
Lena,Robocop,5,2000/12/8 18:59
Solomon,Lethal Weapon,5,2000/12/8 18:59
Cary,Airplane!,5,2000/12/8 19:00
Colin,The Usual Suspects,4,2000/12/5 20:59
Kenny,Clueless,5,2000/12/5 20:52
Gavin,A Clockwork Orange,4,2000/12/5 20:52
Donald,The Talented Mr. Ripley,3,2000/12/5 20:52
Wayne,Back to the Future,3,2000/12/5 20:56
Frank,Being John Malkovich,4,2000/12/5 20:53
Alexander,Predator,5,2000/12/5 20:52
Isaiah,Jaws,4,2000/12/5 20:48
Josephine,Chinatown,3,2000/12/5 20:55
Joshua,The Mask,4,2000/12/5 20:54
August,Platoon,4,2000/12/5 20:53
Jessie,Election,4,2000/12/5 20:52
Yvette,Rocky,5,2000/12/5 20:52
Albert,The Fifth Element,4,2000/12/5 20:55
Eugene,Clueless,4,2000/12/5 17:59
Rachel,Lethal Weapon,5,2000/12/5 17:58
Constance,Raising Arizona,4,2000/12/5 17:59
Larry,The Usual Suspects,4,2000/12/5 15:07
Mike,The Crying Game,5,2000/12/5 15:21
Hank,Independence Day (ID4),4,2000/12/5 15:21
Daniel,There's Something About Mary,4,2000/12/5 15:10
Wesley,Lethal Weapon,5,2000/12/2 19:51
Gina,The Godfather: Part II,3,2000/12/2 19:55
Teresa,Total Recall,4,2000/12/2 19:44
Terry,2001: A Space Odyssey,4,2000/12/2 19:53
Leo,A Clockwork Orange,5,2000/11/28 23:22
Bruce,The Full Monty,2,2000/11/28 23:12
Terence,Predator,5,2000/11/28 23:07
Alice,Jaws,5,2000/11/28 23:20
Benjamin,Psycho,3,2000/11/28 23:08
Sharon,Total Recall,5,2000/11/28 23:13
Ryan,Election,5,2000/11/28 23:18
Mason,The Fifth Element,2,2000/11/28 23:26
Gloria,The Usual Suspects,5,2000/11/28 12:57
Tom,Clueless,3,2000/11/28 13:09
Melissa,A Clockwork Orange,3,2000/12/8 15:10
David,The Talented Mr. Ripley,5,2000/12/25 13:24
Alex,Independence Day (ID4),4,2000/11/28 13:14
Florence,Star Wars: Episode V - The Empire Strikes Back,2,2000/12/8 15:23
Darwin,The Full Monty,2,2000/11/28 13:16
Michael,Being John Malkovich,4,2000/12/25 14:44
```



```
Brown,Predator,5,2000/11/28 13:01
Jimmy,Lethal Weapon,4,2000/12/8 15:07
Jay,Jaws,4,2000/11/28 13:07
Gladys,Psycho,4,2000/11/28 13:08
Denny,The Godfather: Part II,3,2000/12/25 13:25
Jack,Annie Hall,4,2000/12/8 15:05
Edison,The Mask,3,2000/11/28 13:11
Neil,Face/Off,4,2000/12/8 15:22
Jennifer,There's Something About Mary,3,2000/12/25 6:17
Caspar,Superman,3,2000/12/8 15:09
Mickey,Total Recall,1,2000/11/28 13:14
Arthur,American Beauty,3,2000/12/8 15:18
Christine,Platoon,3,2000/12/2 13:21
Adeline,Raising Arizona,4,2000/12/8 15:15
Cody,Blade Runner,1,2000/12/8 15:22
Hillary,Election,3,2000/11/28 12:57
```

Creating a Data Connection in Management Center

In this practice, you need to synchronize data from the MySQL database to MRS Hive, standardize the data based on the GES graph import requirements, and generate metadata using MRS Hive.

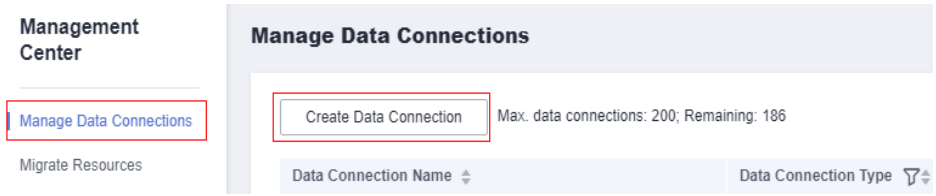
Therefore, you need to create an MRS connection in Management Center. The procedure is as follows:

Step 1 Log in to the DataArts Studio console by following the instructions in **Accessing the DataArts Studio Instance Console**.

Step 2 On the DataArts Studio console, locate a workspace and click **Management Center**.

Step 3 On the displayed **Manage Data Connections** page, click **Create Data Connection**.

Figure 10-3 Creating a data connection



Step 4 In the dialog box displayed, set data connection parameters and click **OK**.

The following part describes how to create an MRS Hive connection. See **Figure 10-4** for details.

- Data connection type: **MRS Hive**
- **Name**: Enter **mrs_hive_link**.
- **Tag**: Enter a new tag name or select an existing tag from the drop-down list box. This parameter is optional.
- **Cluster Name**: Select an existing MRS cluster.
- **Username**: Enter the Kerberos authentication user. In an MRS policy, user **admin** is the default management user and cannot be used as the authentication user of the cluster that uses Kerberos authentication.

Therefore, to create a connection for an MRS cluster that uses Kerberos authentication, perform the following operations:

a. Log in to MRS Manager as user **admin**.

b. Choose **System** > **Permission** > **Security Policy** > **Password Policy**. Click **Add Password Policy** and add a policy under which the password never expires.

   - Set **Password Policy Name** to **neverexp**.
   - Set **Password Validity Period (Days)** to **0**, indicating that the password never expires.
   - Set **Password Expiration Notification (Days)** to **0**.
   - Retain the default values for other parameters.

c. Choose **System** > **Permission** > **User**. On the page displayed, click **Create** to add a dedicated user as the Kerberos authentication user and set the password policy to **neverexp**. Select the user group **superGroup** for the user, and assign all roles to the user.

   NOTE

   - For clusters of MRS 3.1.0 or later, the user must at least have permissions of the **Manager_viewer** role to create data connections in Management Center. To perform database, table, and data operations on components, the user must also have user group permissions of the components.
   - For clusters earlier than MRS 3.1.0, the user must have permissions of the **Manager_administrator** or **System_administrator** role to create data connections in Management Center.
   - A user with only the **Manager_tenant** or **Manager_auditor** permission cannot create connections.

d. Log in to Manager as the new user and change the initial password. Otherwise, the connection fails to be created.

e. Synchronize IAM users.

   i. Log in to the MRS console.

   ii. Choose **Clusters** > **Active Clusters**, select a running cluster, and click its name to go to its details page.

   iii. In the **Basic Information** area of the **Dashboard** page, click **Synchronize** on the right side of **IAM User Sync** to synchronize IAM users.

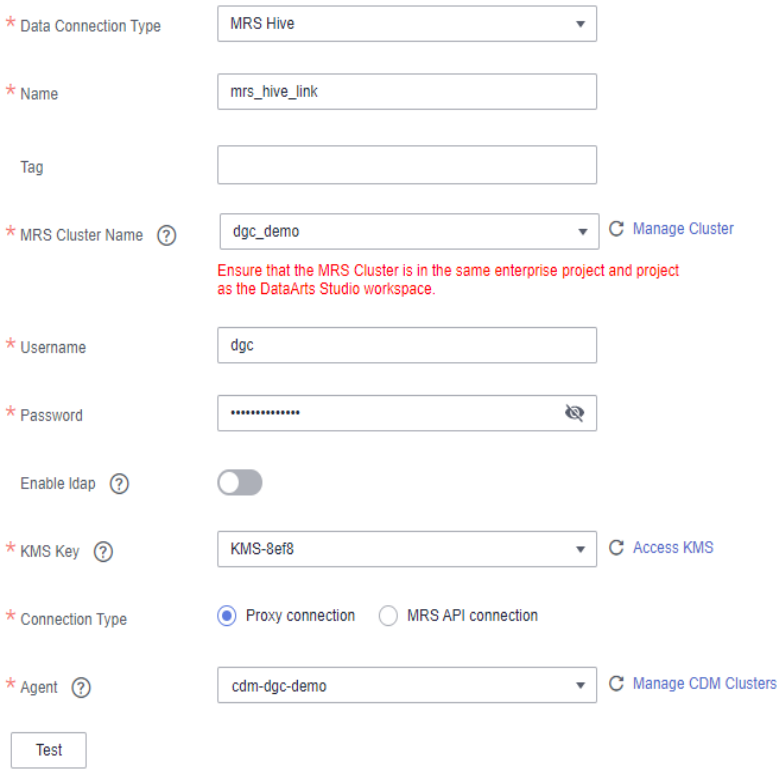
NOTE

- When the policy of the user group to which the IAM user belongs changes from **MRS ReadOnlyAccess** to **MRS CommonOperations**, **MRS FullAccess**, or **MRS Administrator**, wait for 5 minutes until the new policy takes effect after the synchronization is complete because the **SSSD** (System Security Services Daemon) cache of cluster nodes needs time to be updated. Then, submit a job. Otherwise, the job may fail to be submitted.
- When the policy of the user group to which the IAM user belongs changes from **MRS CommonOperations**, **MRS FullAccess**, or **MRS Administrator** to **MRS ReadOnlyAccess**, wait for 5 minutes until the new policy takes effect after the synchronization is complete because the **SSSD** cache of cluster nodes needs time to be updated.

- **Password**: Enter the password of the Kerberos authentication user.
- **KMS Key**: Select a KMS key and use it to encrypt sensitive data. If no KMS key is available, click **Access KMS** to go to the KMS console and create one.
- **Connection Type**: Select **Proxy connection**.
- **Agent**: Select a DataArts Migration cluster as the connection agent. The DataArts Migration cluster and MRS cluster must be in the same region, AZ, VPC, and subnet, and the security group rule must allow communication between the two clusters. In this example, select the DataArts Migration cluster that is automatically created during DataArts Studio instance creation.

  To connect to an MRS 2.*x* cluster, select the DataArts Migration cluster of the 2.*x* version as the agent.

Figure 10-4 Creating an MRS Hive data connection



----End

Creating Data Tables

To facilitate demonstration, you need to import the sample data in CSV format to the MySQL database using DataArts Migration. Then, the MySQL database functions as the data source. You need to create raw data tables in the MySQL database before importing data.

In the formal service process, the source data of the MySQL database needs to be imported to the OBS database as the vertex and edge data sets. In that case, you do not need to create tables in advance. However, before importing source data from the MySQL database to MRS Hive, you need to create standard data tables in the MRS Hive database in advance.

Therefore, in this practice, you need to create raw data tables in the MySQL database and standard data tables in the MRS Hive database. This section describes how to create tables using SQL statements.

Step 1 Create a raw data table in the MySQL database. Run the following SQL statements in the MySQL database to create four raw data tables based on the raw data structure in **Preparing Data Sources**:

```
DROP TABLE IF EXISTS `edge_friends`;
CREATE TABLE `edge_friends` (
   `user1` varchar(32) DEFAULT NULL,
   `user2` varchar(32) DEFAULT NULL
);

DROP TABLE IF EXISTS `edge_rate`;
CREATE TABLE `edge_rate` (
   `user` varchar(32) DEFAULT NULL,
   `movie` varchar(64) DEFAULT NULL,
   `score` int(11) unsigned DEFAULT NULL,
   `datatime` varchar(32) DEFAULT NULL
);

DROP TABLE IF EXISTS `vertex_movie`;
CREATE TABLE `vertex_movie` (
   `movie` varchar(64) DEFAULT NULL,
   `year` varchar(32) DEFAULT NULL,
   `genres` varchar(64) DEFAULT NULL
);

DROP TABLE IF EXISTS `vertex_user`;
CREATE TABLE `vertex_user` (
   `user` varchar(32) DEFAULT NULL,
   `gender` varchar(32) DEFAULT NULL,
   `age` varchar(32) DEFAULT NULL,
   `occupation` varchar(32) DEFAULT NULL,
   `zip-code` varchar(32) DEFAULT NULL
);
```

Step 2 Create standard data tables in the MRS Hive database.

Standardize the raw data structure based on the GES graph import requirements, that is, add labels to the second columns of vertex tables **vertex_user** and **vertex_movie**, and add labels to the third columns of edge tables **edge_rate** and **edge_friends**.

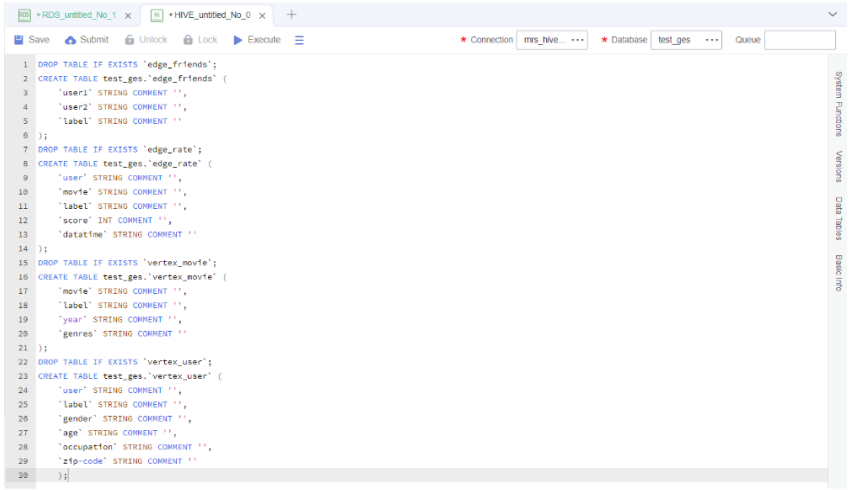
NOTICE

The vertex and edge data sets must comply with the data format requirements of GES graphs. The graph data format requirements are briefed as follows. For details, see **Graph Data Formats**.

- The vertex data set contains the data of each vertex. Each row is the data of a vertex. The format is as follows. **id** is the unique identifier of vertex data.
  ```
  id,label,property 1,property 2,property 3,...
  ```
- The edge data set contains the data of each edge. Each row is the data of an edge. Graph specifications in GES are defined based on the edge quantity, for example, one million edges. The format is as follows. **id 1** and **id 2** are the IDs of the two endpoints of an edge.
  ```
  id 1, id 2, label, property 1, property 2,...
  ```

On the DataArts Factory console, you can select the MRS Hive connection created in **Creating a Data Connection in Management Center**, select a database, and run the following SQL statement to create a standard data table in the MRS Hive database.

Figure 10-5 Creating a standard data table in the MRS Hive database




```
DROP TABLE IF EXISTS `edge_friends`;
CREATE TABLE test_ges.`edge_friends` (
   `user1` STRING COMMENT '',
   `user2` STRING COMMENT '',
   `label` STRING COMMENT ''
);

DROP TABLE IF EXISTS `edge_rate`;
CREATE TABLE test_ges.`edge_rate` (
   `user` STRING COMMENT '',
   `movie` STRING COMMENT '',
   `label` STRING COMMENT '',
   `score` INT COMMENT '',
   `datatime` STRING COMMENT ''
);

DROP TABLE IF EXISTS `vertex_movie`;
CREATE TABLE test_ges.`vertex_movie` (
   `movie` STRING COMMENT '',
   `label` STRING COMMENT '',
```



```
    `year` STRING COMMENT '',
    `genres` STRING COMMENT ''
);

DROP TABLE IF EXISTS `vertex_user`;
CREATE TABLE test_ges.`vertex_user` (
    `user` STRING COMMENT '',
    `label` STRING COMMENT '',
    `gender` STRING COMMENT '',
    `age` STRING COMMENT '',
    `occupation` STRING COMMENT '',
    `zip-code` STRING COMMENT ''
);
```

----End

10.3 Creating a Data Integration Job

This section described how to create a DataArts Studio data integration job.

In this example, you need to create the following three types of integration jobs:

1. **Migrating Data from OBS to MySQL**: To facilitate demonstration, you need to import the sample data in CSV format from OBS to the MySQL database.
2. **Migrating Data from MySQL to OBS**: In the formal service process, the original sample data in MySQL needs to be imported to OBS and normalized as the vertex data sets and edge data sets.
3. **Migrating Data from MySQL to MRS Hive**: In the formal service process, the original sample data in MySQL needs to be imported to MRS Hive and converted to standard vertex and edge data sets.

Creating a Cluster

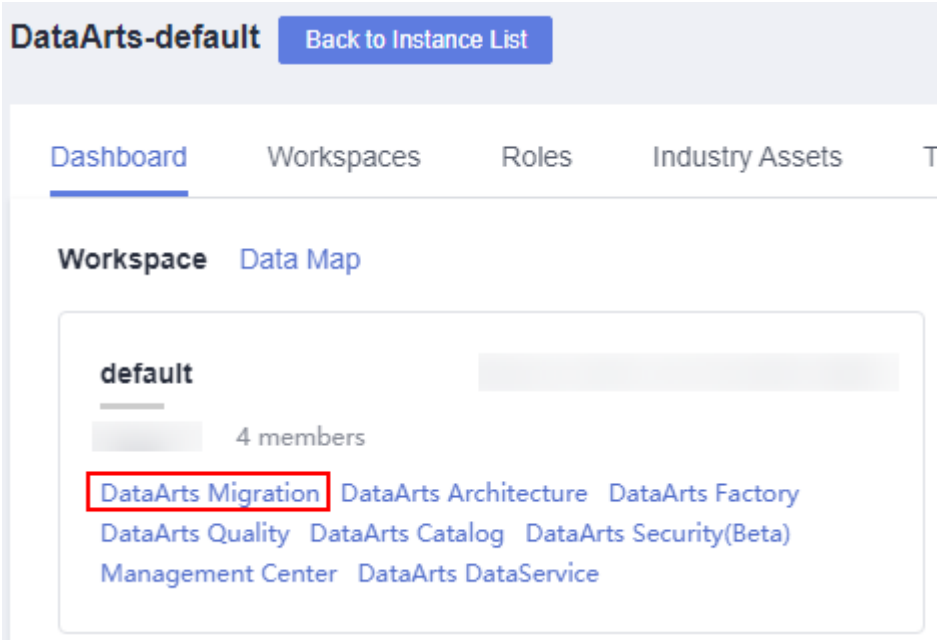
CDM clusters can migrate data to the cloud and integrate data into the data lake. It provides wizard-based configuration and management and can integrate data from a single table or an entire database incrementally or periodically. The DataArts Studio basic package contains a CDM cluster. If the cluster cannot meet your requirements, you can buy a DataArts Migration incremental package.

For details about how to buy DataArts Studio incremental packages, see **Buying a DataArts Studio Incremental Package**.

Creating Data Integration Connections

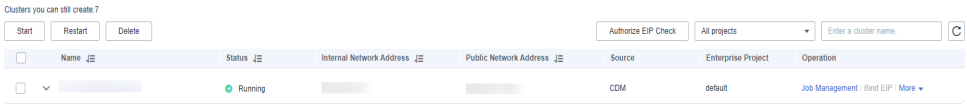
Step 1 Log in to the DataArts Studio console. Locate an instance and click **Access**. On the displayed page, locate a workspace and click **DataArts Migration**.

Figure 10-6 DataArts Migration



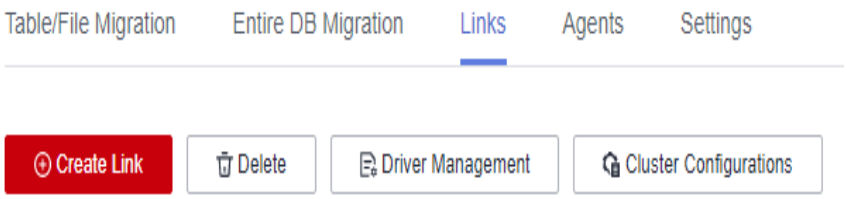
Step 2 In the left navigation pane, choose **Cluster Management**. In the cluster list, locate the required cluster and click **Job Management**.

Figure 10-7 Cluster management



Step 3 On the **Job Management** page, click **Links**.

Figure 10-8 Links



Step 4 Create the OBS connection, Cloud Database MySQL connection, and MRS Hive connection required for an integration task.

Click **Create Link**. On the page displayed, select **Object Storage Service (OBS)** and click **Next**. Then, set the link parameters and click **Save**.

Figure 10-9 Creating an OBS link

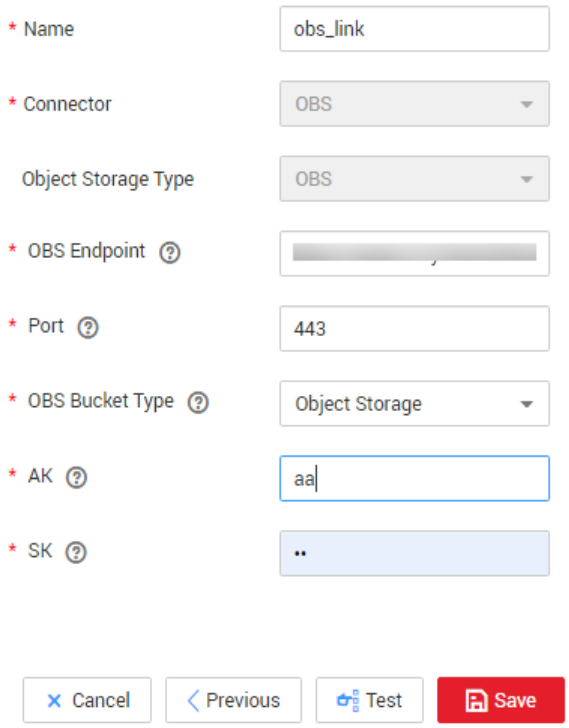
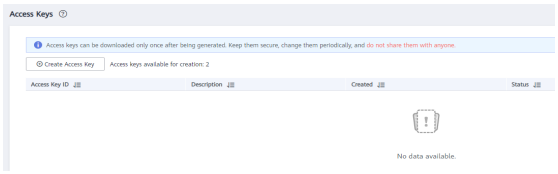


Table 10-1 Parameter description

Parameter	Description	Example Value
Name	Link name, which should be defined based on the data source type, so it is easier to remember what the link is for	obs_link
OBS Endpoint	An endpoint is the **request address** for calling an API. Endpoints vary depending on services and regions. You can obtain the OBS bucket endpoint by either of the following means: To obtain the endpoint of an OBS bucket, go to the OBS console and click the bucket name to go to its details page. **NOTE** • If the CDM cluster and OBS bucket are not in the same region, the CDM cluster cannot access the OBS bucket. • Do not change the password or user when the job is running. If you do so, the password will not take effect immediately and the job will fail.	-
Port	Data transmission port. The HTTPS port number is 443 and the HTTP port number is 80.	443

Parameter	Description	Example Value
OBS Bucket Type	Select a value from the drop-down list, generally, **Object Storage**.	Object Storage
AK	AK and SK are used to log in to the OBS server.	-
SK	You need to create an access key for the current account and obtain an AK/SK pair. To obtain an access key, perform the following steps: 1. Log in to the management console, move the cursor to the username in the upper right corner, and select **My Credentials** from the drop-down list. 2. On the **My Credentials** page, choose **Access Keys**, and click **Create Access Key**. See **Figure 10-10**. **Figure 10-10** Clicking Create Access Key  3. Click **OK** and save the access key file as prompted. The access key file will be saved to your browser's configured download location. Open the **credentials.csv** file to view **Access Key Id** and **Secret Access Key**. **NOTE** – Only two access keys can be added for each user. – To ensure access key security, the access key is automatically downloaded only when it is generated for the first time and cannot be obtained from the management console later. Keep them properly.	-

On the **Links** tab page, click **Create Link** again. On the page displayed, select the **RDS for MySQL** connector and click **Next**. Then, set the connection parameters and click **Save**.

Figure 10-11 Creating a MySQL link

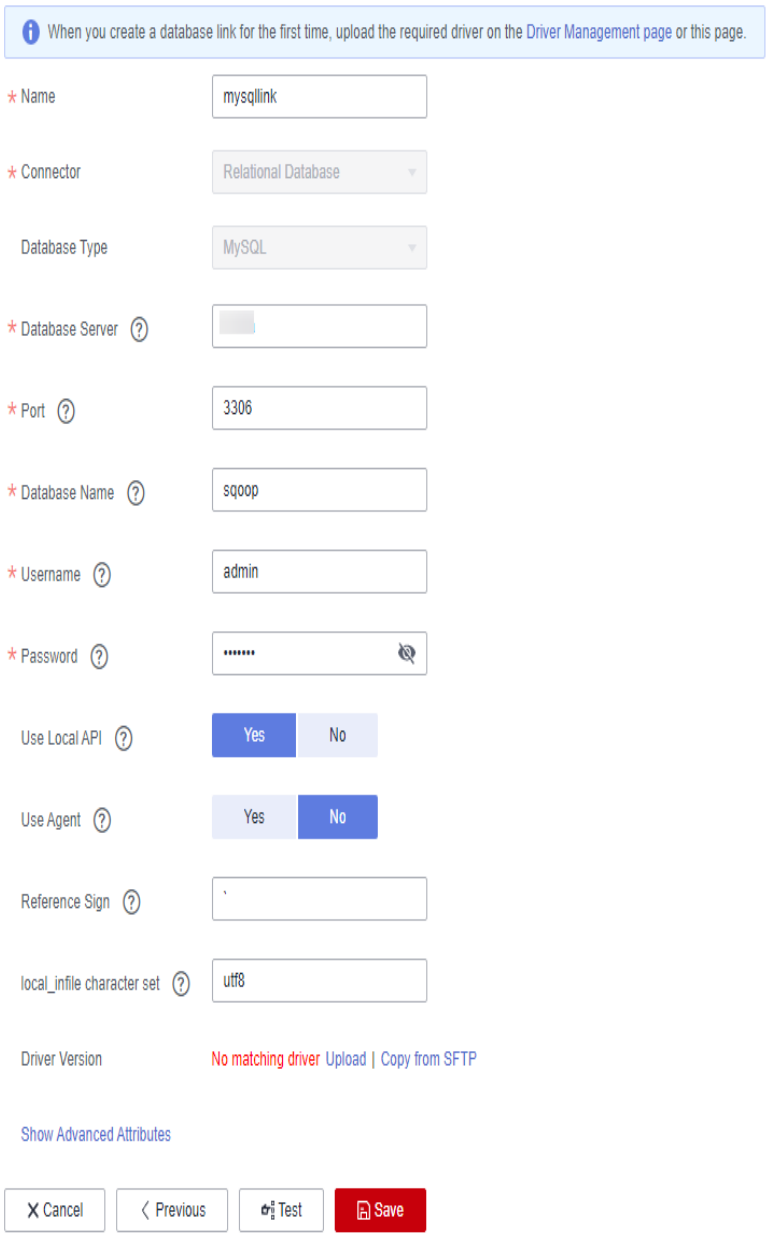


Table 10-2 MySQL link parameters

Parameter	Description	Example Value
Name	Unique link name	mysqllink
Database Server	IP address or domain name of the MySQL database server	-
Port	MySQL database port	3306
Database Name	Name of the MySQL database	sqoop

Parameter	Description	Example Value
Username	User who has the read, write, and delete permissions on the MySQL database	admin
Password	Password of the user	-
Use Local API	Whether to use the local API of the database for acceleration. (The system attempts to enable the **local_infile** system variable of the MySQL database.)	Yes
Use Agent	Whether to extract data from the data source through an agent	No
local_infile Character Set	When using local_infile to import data to MySQL, you can configure the encoding format.	utf8
Driver Version	Before connecting CDM to a relational database, you need to upload the JDK 8 .jar driver of the relational database. Download the MySQL driver 5.1.48 from **https://downloads.mysql.com/archives/c-j/**, obtain **mysql-connector-java-5.1.48.jar**, and upload it.	-

On the **Links** tab page, click **Create Link** again. On the page displayed, select **MRS Hive** and click **Next**. Then, set the link parameters and click **Save**.

Figure 10-12 Creating an MRS Hive link

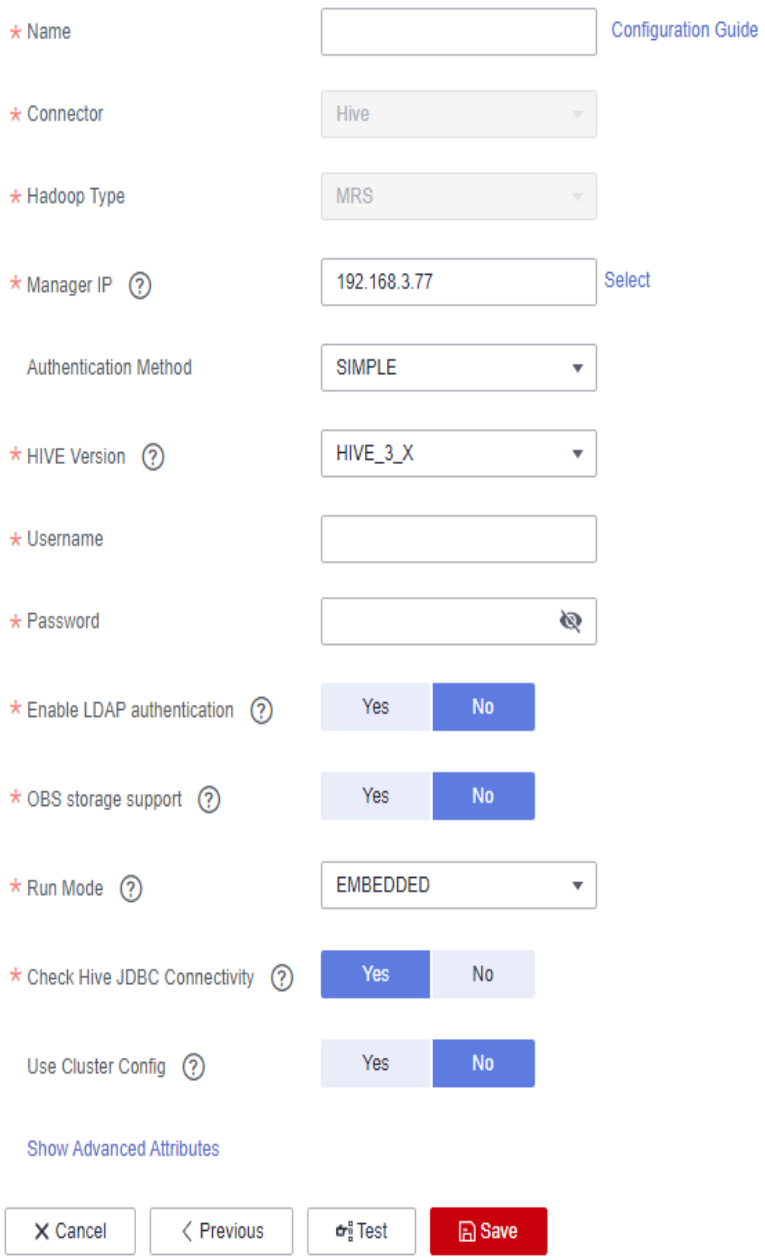
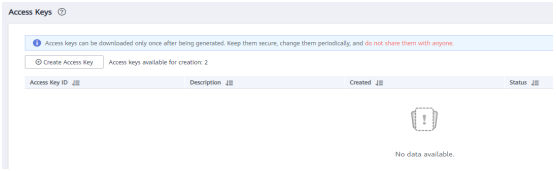


Table 10-3 MRS Hive link parameters

Parameter	Description	Example Value
Name	Link name, which should be defined based on the data source type, so it is easier to remember what the link is for	hivelink

Parameter	Description	Example Value
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
Authentication Method	Authentication method used for accessing MRS • **SIMPLE**: Select this for non-security mode. • **KERBEROS**: Select this for security mode.	SIMPLE
HIVE Version	Set this 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 columns of an MRS component, you also need to add the database, table, and column permissions of the MRS component to the user by following the instructions in the MRS documentation. • 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** or **System_administrator** to create links on CDM. • A user with only the **Manager_tenant** or **Manager_auditor** permission cannot create connections.	cdm
Password	Password used for logging in to MRS Manager	-

Parameter	Description	Example Value
Enable ldap	This parameter is available when **Proxy connection** is selected for **Connection Type**. If LDAP authentication is enabled for an external LDAP server connected to MRS Hive, the LDAP username and password are required for authenticating the connection to MRS Hive. In this case, this option must be enabled. Otherwise, the connection will fail.	No
ldapUsername	This parameter is mandatory when **Enable ldap** is enabled. Enter the username configured when LDAP authentication was enabled for MRS Hive.	-
ldapPassword	This parameter is mandatory when **Enable ldap** is enabled. Enter the password configured when LDAP authentication was enabled for MRS Hive.	-
OBS storage support	The server must support OBS storage. When creating a Hive table, you can store the table in OBS.	No

Parameter	Description	Example Value
AK	This parameter is mandatory when **OBS storage support** is enabled. The account corresponding to the AK/SK pair must have the OBS Buckets Viewer permission. Otherwise, OBS cannot be accessed and the "403 AccessDenied" error is reported. You need to create an access key for the current account and obtain an AK/SK pair. 1. Log in to the management console, move the cursor to the username in the upper right corner, and select **My Credentials** from the drop-down list. 2. On the **My Credentials** page, choose **Access Keys**, and click **Create Access Key**. See **Figure 10-13**. **Figure 10-13** Clicking Create Access Key  3. Click **OK** and save the access key file as prompted. The access key file will be saved to your browser's configured download location. Open the **credentials.csv** file to view **Access Key Id** and **Secret Access Key**. **NOTE** – Only two access keys can be added for each user. – To ensure access key security, the access key is automatically downloaded only when it is generated for the first time and cannot be obtained from the management console later. Keep them properly.	-
SK		-

Parameter	Description	Example Value
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, **Standalone** prevails. **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
Check Hive JDBC Connectivity	Whether to check the Hive JDBC connectivity	No
Use Cluster Config	You can use the cluster configuration to simplify parameter settings for the Hadoop connection.	No
Cluster Config Name	This parameter is valid only when **Use Cluster Config** is set to **Yes**. Select a cluster configuration that has been created. For details about how to configure a cluster, see **Managing Cluster Configurations**.	hive_01

----End

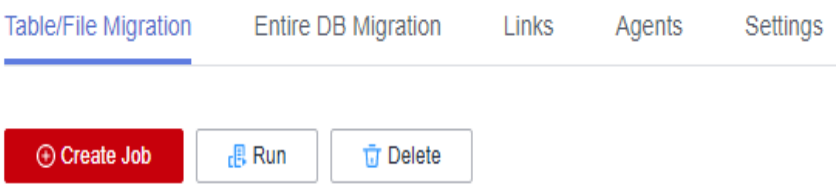
Creating a Job for Migrating Data from OBS to MySQL

To facilitate demonstration, you need to import the sample data in CSV format from OBS to the MySQL database.

Step 1 On the DataArts Migration console, click **Cluster Management** in the left navigation pane, locate the required cluster in the cluster list, and click **Job Management**.

Step 2 On the **Job Management** page, click **Table/File Migration** and click **Create Job**.

Figure 10-14 Table/File Migration

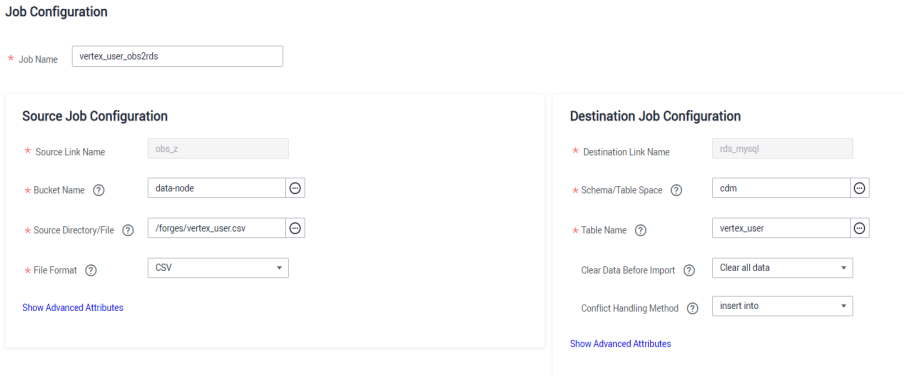


Step 3 Perform the following steps to migrate the four original data tables in **Preparing Data Sources** from OBS to the MySQL database:

1. Configure the vertex_user_obs2rds job.

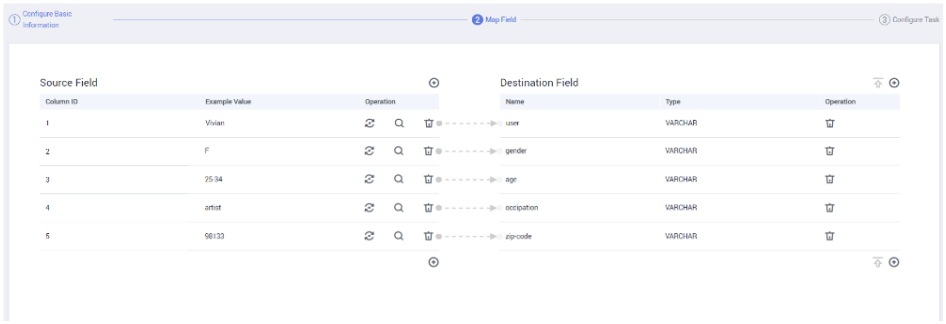
   Set **Source Directory/File** to the **vertex_user.csv** file uploaded to OBS in **Preparing Data Sources**. Because the table contains Chinese characters, you need to set encoding type to **GBK**. Set destination **Table Name** to the **vertex_user** table created in **Making Preparations**. Click **Next**.

   Figure 10-15 vertex_user_obs2rds job configuration

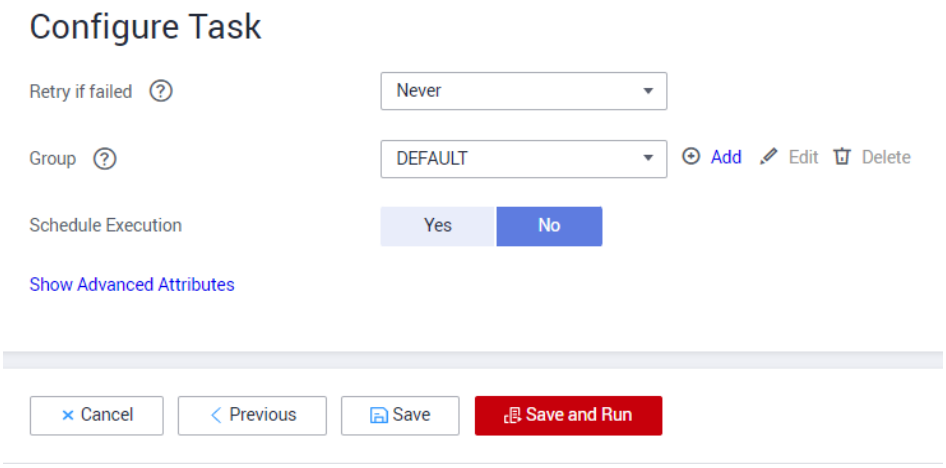
2. In **Map Field**, check whether the field mapping sequence is correct. If the field mapping sequence is correct, click **Next**.

   Figure 10-16 Filed mapping of the vertex_user_obs2rds job

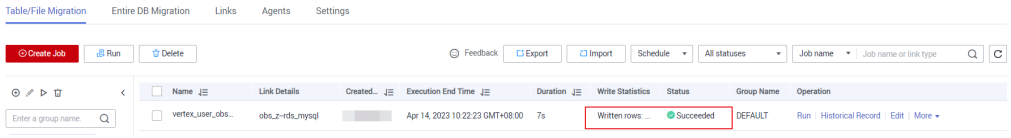
3. The task configuration does not need to be modified. Save the configuration and run the task.

Figure 10-17 Configuring the task



Step 4 Wait till the job is complete. If the job is successfully executed, the **vertex_user** table has been successfully migrated to the MySQL database.

Figure 10-18 vertex_user_obs2rds job completed



Step 5 Create the **vertex_movie_obs2rds**, **edge_friends_obs2rds**, and **edge_rate_obs2rds** jobs and migrate the four created original tables from OBS to MySQL based on instructions from **Step 2** to **Step 4**.

----End

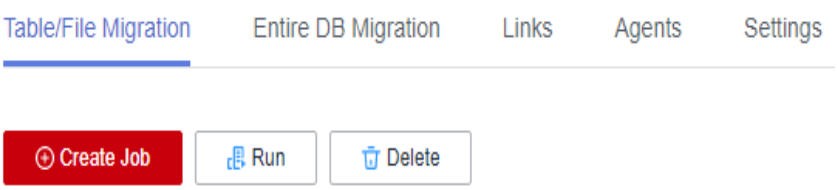
Creating a Job for Migrating Data from MySQL to OBS

In the formal service process, the original sample data in MySQL needs to be imported to OBS and converted to standard vertex data sets and edge data sets.

Step 1 On the DataArts Migration console, click **Cluster Management** in the left navigation pane, locate the required cluster in the cluster list, and click **Job Management**.

Step 2 On the **Job Management** page, click **Table/File Migration** and click **Create Job**.

Figure 10-19 Table/File Migration



Step 3 Perform the following steps to migrate the four original data tables from MySQL to OBS buckets in sequence:

1. Configure the vertex_user_rds2obs job.

   Set the source table name to the name of the **vertex_user** table migrated to MySQL in **Creating a Job for Migrating Data from OBS to MySQL**. Set the destination write directory to a directory other than the directory where the raw data is stored, to avoid file overwriting. Set file format to **CSV**. Because the table contains Chinese characters, you need to set encoding type to **GBK**.

   Note: You need to configure custom file name in the advanced attributes of the destination end. The value is *${tableName}*. If this parameter is not set, the names of CSV files migrated to OBS will contain extra fields such as timestamps. As a result, the names of files obtained each time a migration job is executed are different, and the graph data cannot be automatically imported to GES after a migration.

   Other advanced attributes do not need to be set. Click **Next**.

   Figure 10-20 Basic configuration of the vertex_user_rds2obs job

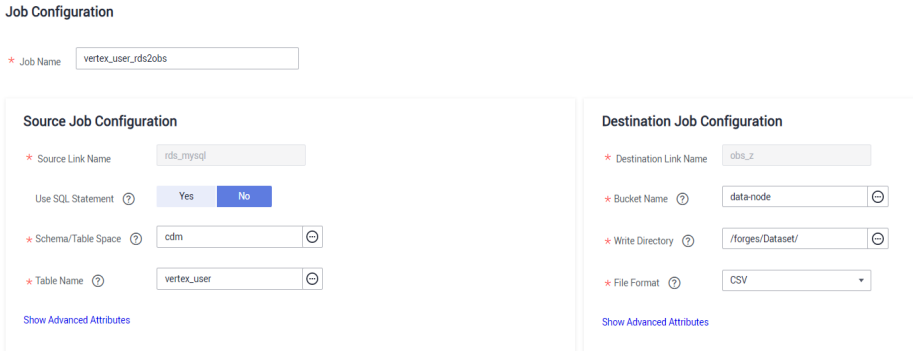
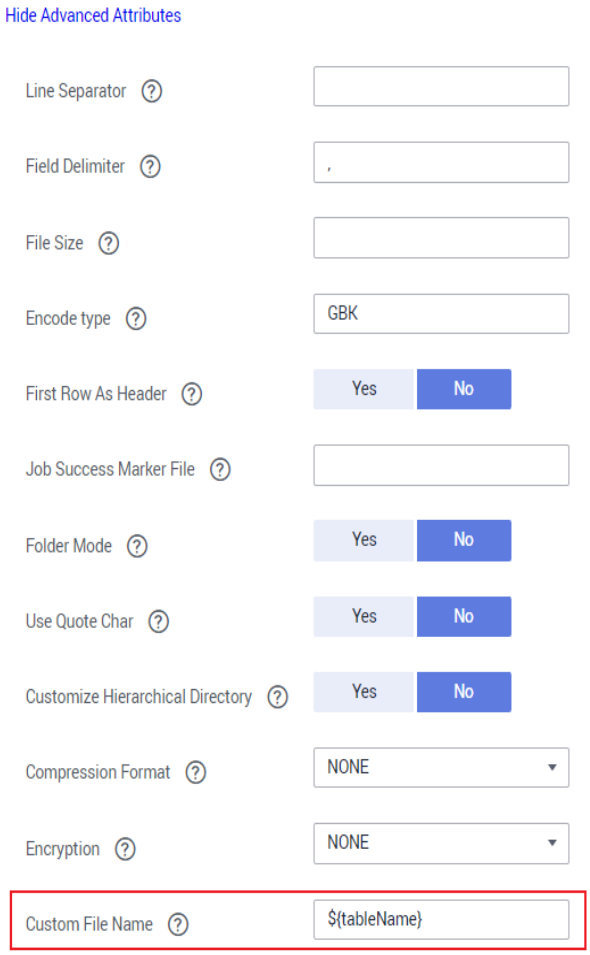


Figure 10-21 Advanced configuration of the vertex_user_rds2obs job



2. In field mapping, the **label** field needs to be added as the label of the graph file based on the GES graph data requirements.
   - **vertex_user**: Set **label** to **user** and move this field to the second column.
   - **vertex_movie**: Set **label** to **movie** and move this field to the second column.
   - **edge_friends**: Set **label** to **friends** and move this field to the third column.
   - **edge_rate**: Set **label** to **rate** and move this field to the third column.

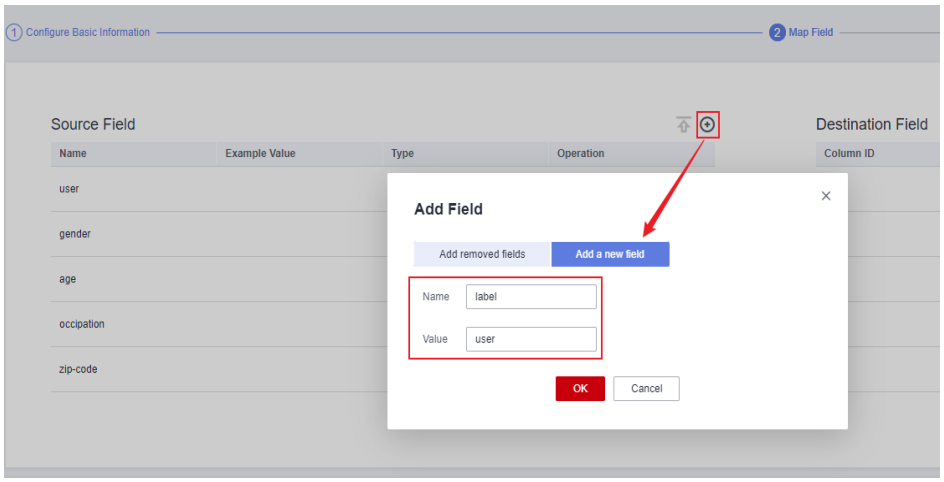
   Standardize the raw data structure based on the GES graph import requirements, that is, add labels to the second columns of vertex tables **vertex_user** and **vertex_movie**, and add labels to the third columns of edge tables **edge_rate** and **edge_friends**.

NOTICE

The vertex and edge data sets must comply with the data format requirements of GES graphs. The graph data format requirements are briefed as follows. For details, see **Graph Data Formats**.

- The vertex data set contains the data of each vertex. Each row is the data of a vertex. The format is as follows. **id** is the unique identifier of vertex data.
  ```
  id,label,property 1,property 2,property 3,...
  ```
- The edge data set contains the data of each edge. Each row is the data of an edge. Graph specifications in GES are defined based on the edge quantity, for example, one million edges. The format is as follows. **id 1** and **id 2** are the IDs of the two endpoints of an edge.
  ```
  id 1, id 2, label, property 1, property 2,...
  ```

Figure 10-22 Adding field mapping in the vertex_user_rds2obs job



3. Adjust the field sequence. For the vertex data set, move **label** to the second column. For the edge data set, move **label** to the third column. After the adjustment is complete, as shown in **Figure 10-24**. Then, click **Next**.

Figure 10-23 Adjusting the field sequence of the vertex_user_rds2obs job

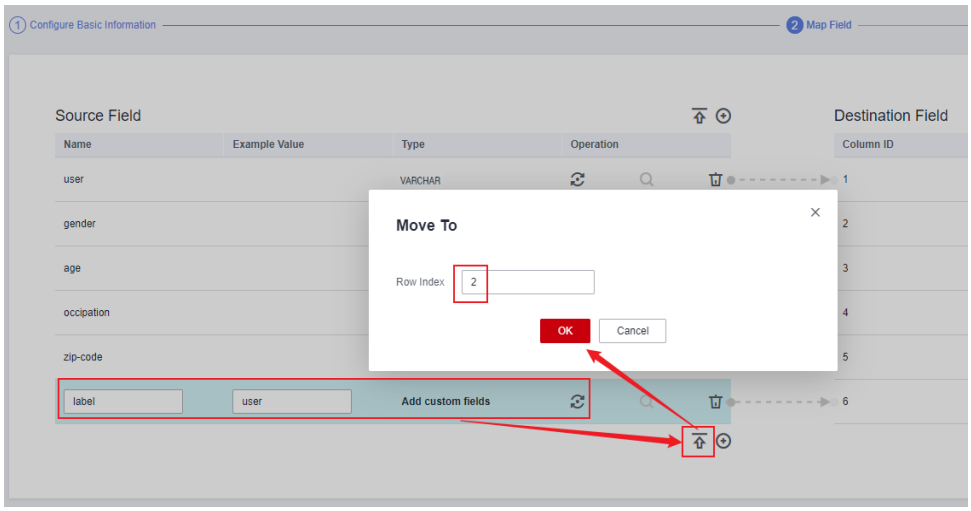
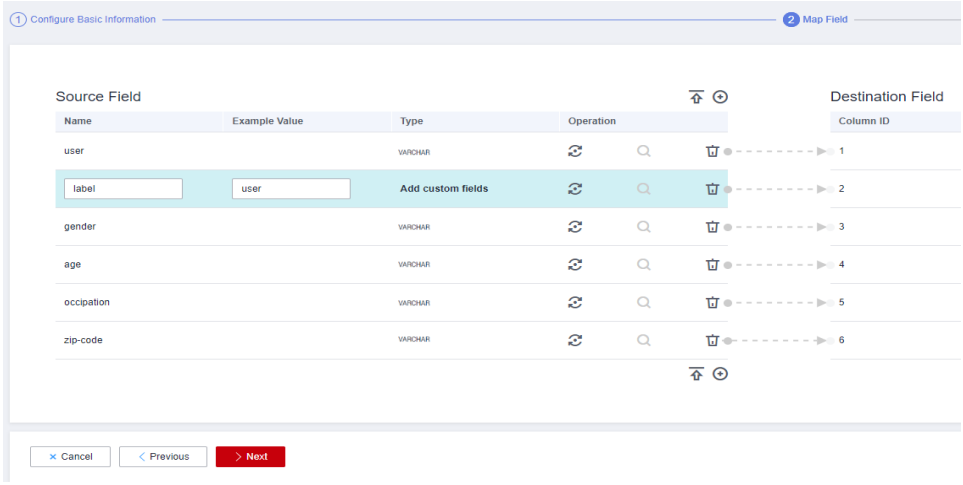
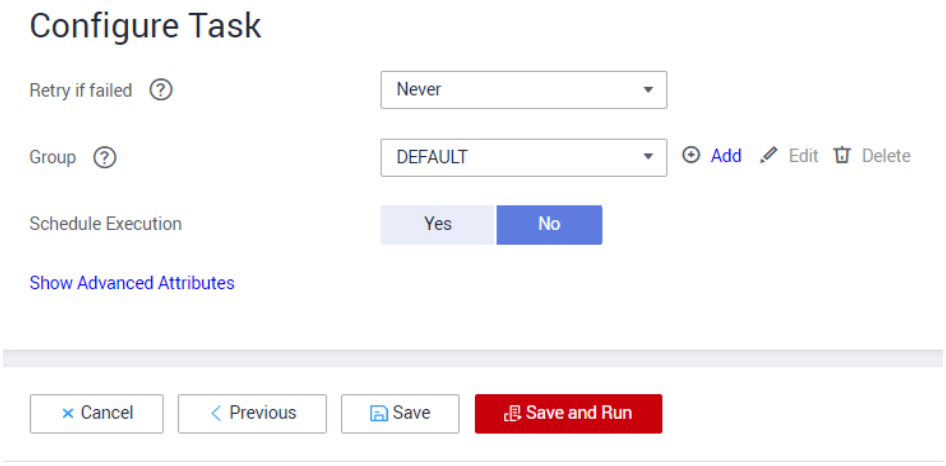


Figure 10-24 Field mapping of the vertex_user_rds2obs job



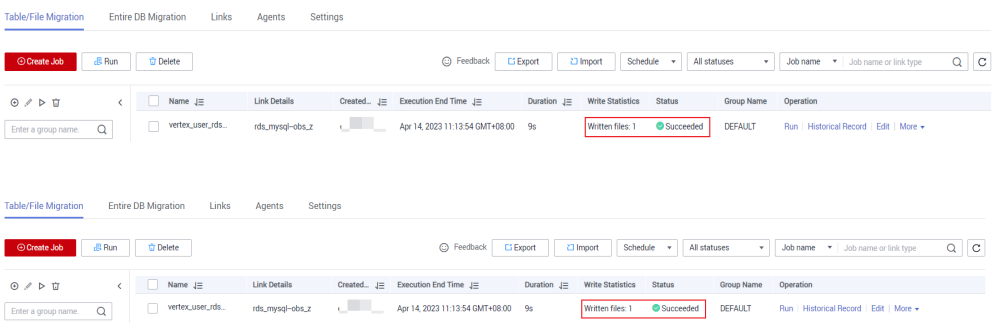
4. The task configuration does not need to be modified. Save the configuration and run the task.

Figure 10-25 Configuring the task



Step 4 Wait till the job is complete. If the job is successfully executed, the **vertex_user.csv** table has been successfully migrated to the OBS buckets.

Figure 10-26 vertex_user_rds2obs job completed



Step 5 Create the **vertex_movie_rds2obs**, **edge_friends_rds2obs**, and **edge_rate_rds2obs** jobs and write the four original tables from MySQL to OBS buckets based on instructions from **Step 2** to **Step 4**.

----End

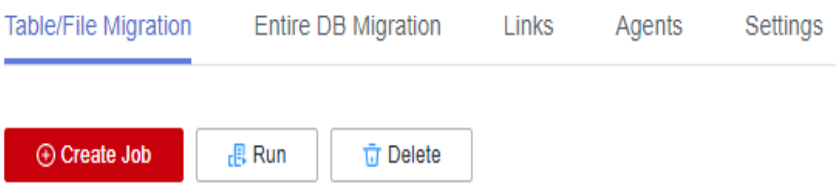
Creating a Job for Migrating Data from MySQL to MRS Hive

In the formal service process, the original sample data in MySQL needs to be imported to MRS Hive and converted to standard vertex data sets and edge data sets.

Step 1 On the DataArts Migration console, click **Cluster Management** in the left navigation pane, locate the required cluster in the cluster list, and click **Job Management**.

Step 2 On the **Job Management** page, click **Table/File Migration** and click **Create Job**.

Figure 10-27 Table/File Migration

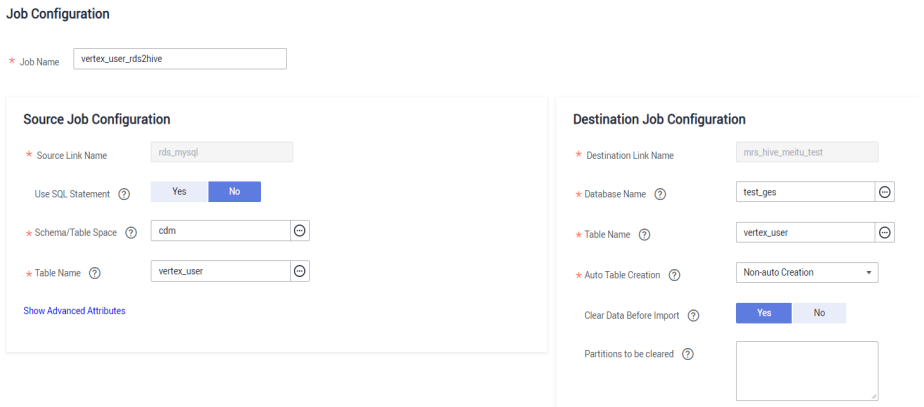


Step 3 Perform the following steps to migrate the four original data tables from MySQL to MRS Hive in sequence:

1. Configure the vertex_user_rds2hive job.

   Set source table name to the name of the **vertex_user** table migrated to the MySQL database in **Creating a Job for Migrating Data from OBS to MySQL**. Set the destination table name to the name of the **vertex_user** table created in **Making Preparations**. Set other parameters as shown in the following figure. The advanced attributes are not required. Then, click **Next**.

   Figure 10-28 Basic configuration of the vertex_user_rds2hive job



2. In field mapping, the **label** field needs to be added as the label of the graph file based on the GES graph data requirements.

- **vertex_user**: Set **label** to **user** and move this field to the second column.
- **vertex_movie**: Set **label** to **movie** and move this field to the second column.
- **edge_friends**: Set **label** to **friends** and move this field to the third column.
- **edge_rate**: Set **label** to **rate** and move this field to the third column.

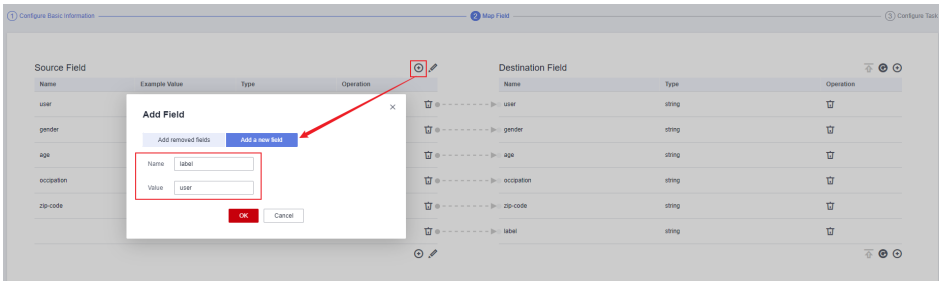
Standardize the raw data structure based on the GES graph import requirements, that is, add labels to the second columns of vertex tables **vertex_user** and **vertex_movie**, and add labels to the third columns of edge tables **edge_rate** and **edge_friends**.

NOTICE

The vertex and edge data sets must comply with the data format requirements of GES graphs. The graph data format requirements are briefed as follows. For details, see **Graph Data Formats**.

- The vertex data set contains the data of each vertex. Each row is the data of a vertex. The format is as follows. **id** is the unique identifier of vertex data.
  ```
  id,label,property 1,property 2,property 3,...
  ```
- The edge data set contains the data of each edge. Each row is the data of an edge. Graph specifications in GES are defined based on the edge quantity, for example, one million edges. The format is as follows. **id 1** and **id 2** are the IDs of the two endpoints of an edge.
  ```
  id 1, id 2, label, property 1, property 2,...
  ```

Figure 10-29 Adding field mapping in the vertex_user_rds2hive job



3. Adjust the field sequence. For the vertex file, move **label** to the second column. For the edge file, move **label** to the third column. After the adjustment is complete, as shown in **Figure 10-31**. Then, click **Next**.

Figure 10-30 Adjusting the field sequence of the vertex_user_rds2hive job

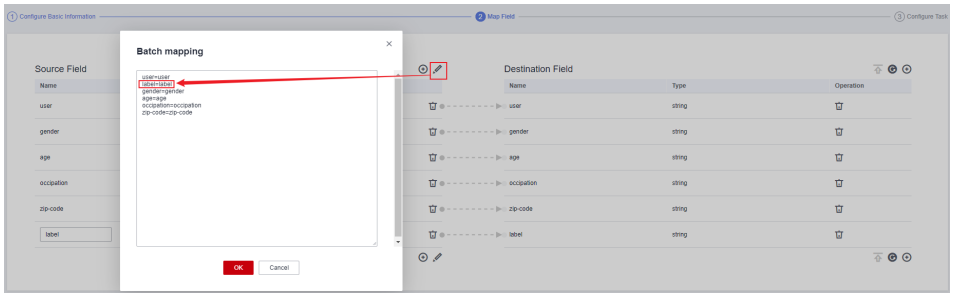
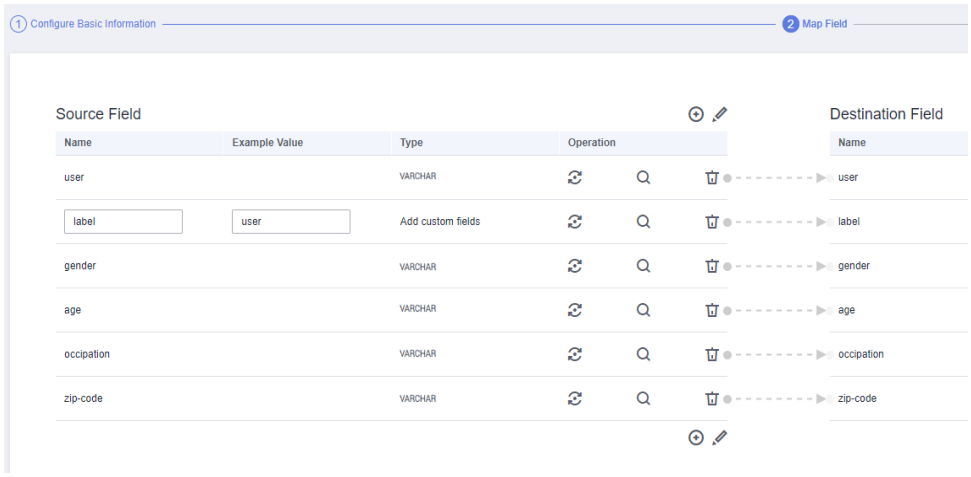
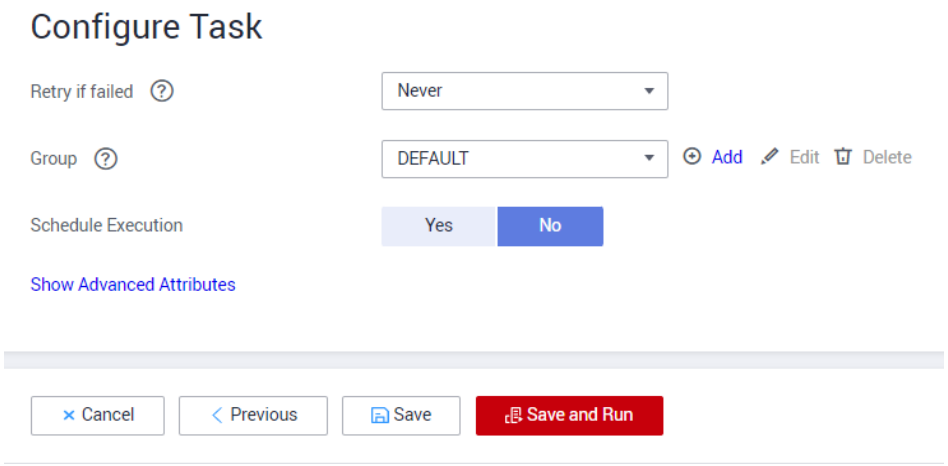


Figure 10-31 Field mapping of the vertex_user_rds2hive job



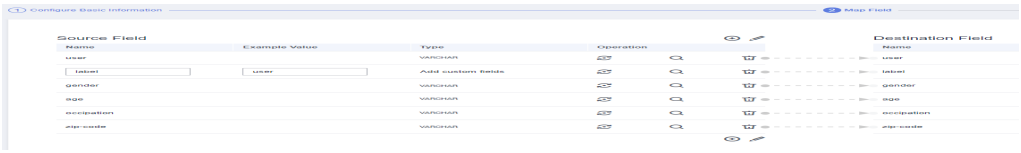
4. The task configuration does not need to be modified. Save the configuration and run the task.

Figure 10-32 Configuring the task



Step 4 Wait till the job is complete. If the job is successfully executed, the **vertex_user** table has been successfully migrated to MRS Hive.

Figure 10-33 vertex_user_rds2hive job completed



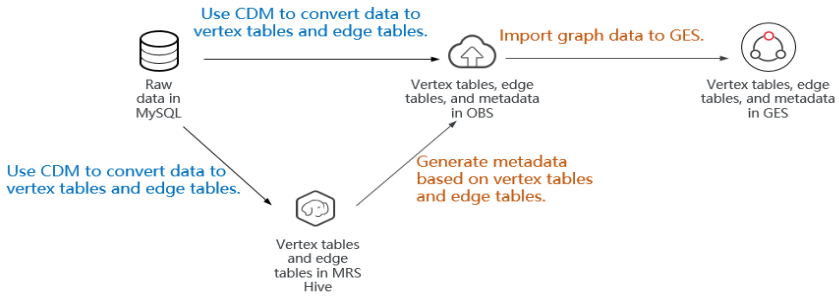
Step 5 Create the **vertex_movie_rds2hive**, **edge_friends_rds2hive**, and **edge_rate_rds2hive** jobs and migrate the four original tables from MySQL to MRS Hive based on instructions from **Step 2** to **Step 4**.

----End

10.4 Developing and Scheduling an Import GES Job

This section describes how to invoke a data integration job through data development to periodically synchronize raw data in MySQL to OBS and MRS Hive and convert the data to standard GES vertex/edge data sets. Then, graph metadata is automatically generated based on the standard vertex and edge data sets. In this way, graph data (vertex data sets, edge data sets, and metadata) is periodically imported to GES.

Figure 10-34 Business scenarios



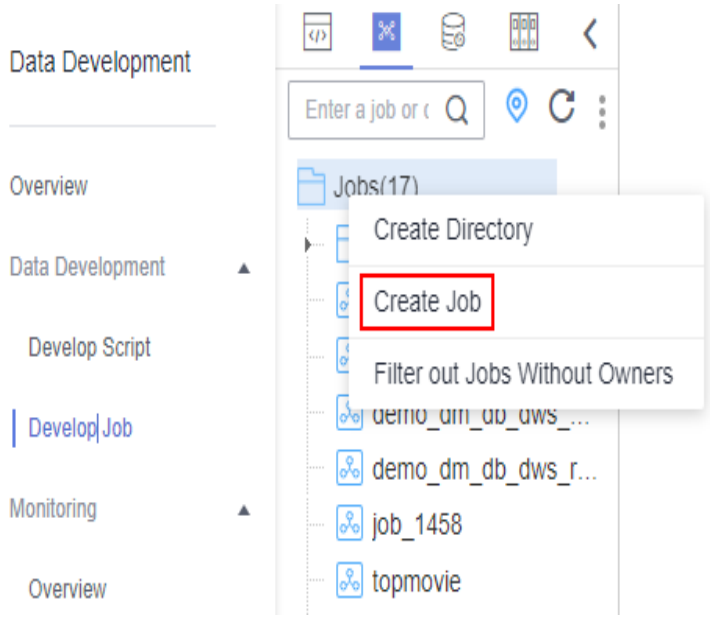
Procedure

Assume that the raw data tables in the MySQL database are updated every day. If you want to update the latest graph data generated based on the raw data to GES every day, perform the following steps to compile jobs and periodically schedule the jobs:

Step 1 On the DataArts Studio console, locate a workspace and click **DataArts Factory**.

Step 2 Create a data development batch processing job and name it **import_ges**.

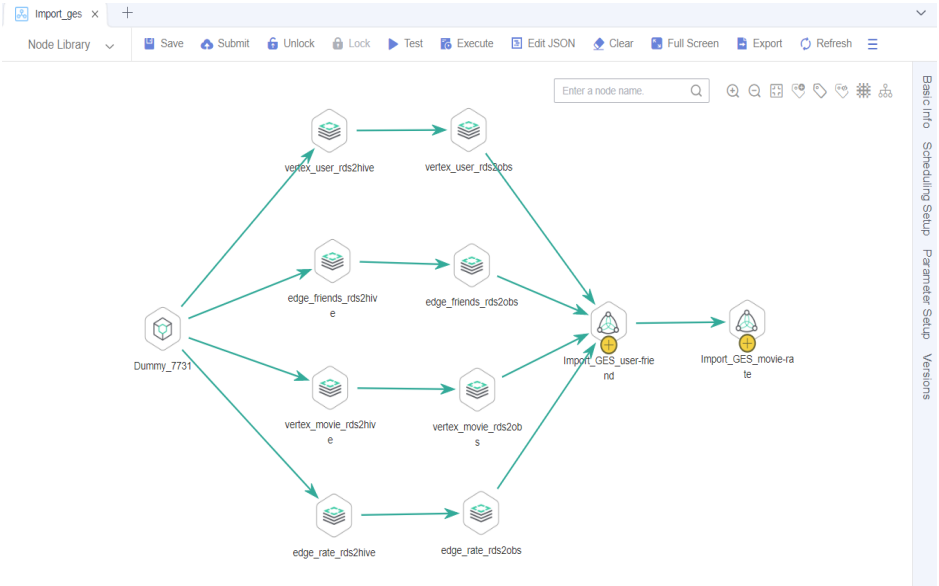
Figure 10-35 Creating a job



Step 3 On the job development page, drag one Dummy node, eight CDM Job nodes, and two Import GES nodes to the canvas, select and drag , as shown in **Figure 10-36**.

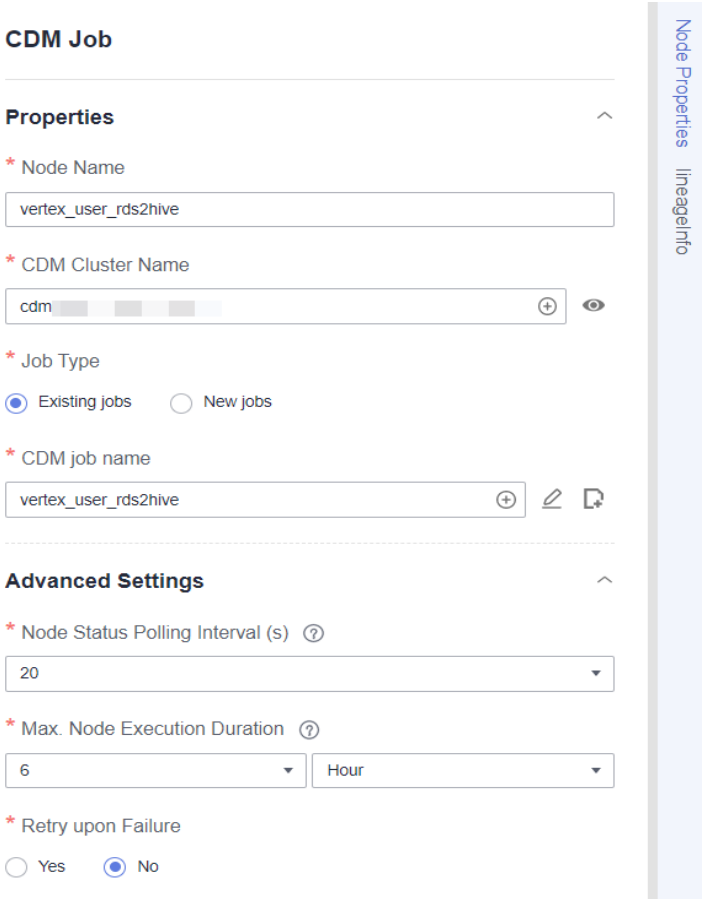
The Dummy node serves only as an end identifier. The CDM Job nodes are used to invoke data integration jobs created in **Creating a Data Integration Job**. The Import GES nodes are used to import graph data to GES.

Figure 10-36 Compiling a job



Step 4 Configure the eight CDM Job nodes in the job. Invoke the created data integration job to convert the raw data to the standard GES vertex/edge data sets and synchronize the data to OBS and MRS Hive.

Figure 10-37 Configuring CDM nodes



CDM node description:

- **vertex_user_rds2hive** (CDM Job node): In **Node Properties**, select the CDM cluster in **Creating a Data Integration Job** and associate it with the CDM job **vertex_user_rds2hive**.
- **vertex_user_rds2obs** (CDM Job node): In **Node Properties**, select the CDM cluster in **Creating a Data Integration Job** and associate it with the CDM job **vertex_user_rds2obs**.
- **edge_friends_rds2hive** (CDM Job node): In **Node Properties**, select the CDM cluster in **Creating a Data Integration Job** and associate it with the CDM job **edge_friends_rds2hive**.
- **edge_friends_rds2obs** (CDM Job node): In **Node Properties**, select the CDM cluster in **Creating a Data Integration Job** and associate it with the CDM job **edge_friends_rds2obs**.
- **vertex_movie_rds2hive** (CDM Job node): In **Node Properties**, select the CDM cluster in **Creating a Data Integration Job** and associate it with the CDM job **vertex_movie_rds2hive**.
- **vertex_movie_rds2obs** (CDM Job node): In **Node Properties**, select the CDM cluster in **Creating a Data Integration Job** and associate it with the CDM job **vertex_movie_rds2obs**.

- **edge_rate_rds2hive** (CDM Job node): In **Node Properties**, select the CDM cluster in **Creating a Data Integration Job** and associate it with the CDM job **edge_rate_rds2hive**.
- **edge_rate_rds2obs** (CDM Job node): In **Node Properties**, select the CDM cluster in **Creating a Data Integration Job** and associate it with the CDM job **edge_rate_rds2obs**.

Step 5 Configure the two Import GES nodes in the job separately. Only one vertex table and one edge table can be selected for one Import GES node to generate metadata. Therefore, two Import GES nodes are used in this practice to import data in sequence.

Import GES node description:

- **Import_GES_user-friend**: In **Node Properties**, after a graph name is selected, set the edge data set and vertex data set to the **edge_friends** edge table and **vertex_user** vertex table, respectively. In addition, select **Overwrite previous repetitive edges**. Otherwise, a large number of duplicate edges are generated after periodic scheduling.

  Set **Metadata Source** to **New** and click next to **Metadata**. The **New** dialog box is displayed, as shown in **Figure 10-39**. In the **New** dialog box, select the **edge_friends** edge table and **vertex_user** vertex table in MRS, set **Output Directory** to the directory where the OBS vertex tables and edge tables are located, and click **Create**. The system automatically populates the **Metadata** field to the OBS directory where the generated metadata schema is located.

- **Import_GES_movie-rate**: In **Node Properties**, after a graph name is selected, set the edge data set and vertex data set to the **edge_rate** edge table and **vertex_movie** vertex table, respectively. In addition, select **Overwrite previous repetitive edges**. Otherwise, a large number of duplicate edges are generated after periodic scheduling.

  Set **Metadata Source** to **New** and click the generation button next to **Metadata**. The **New** dialog box is displayed, as shown in **Figure 10-39**. In the **New** dialog box, select the **edge_rate** edge table and **vertex_movie** vertex table in MRS, set **Output Directory** to the directory where the OBS vertex tables and edge tables are located, and click **Create**. The system automatically populates the **Metadata** field to the OBS directory where the generated metadata schema is located.

Figure 10-38 Configuring the Import GES node

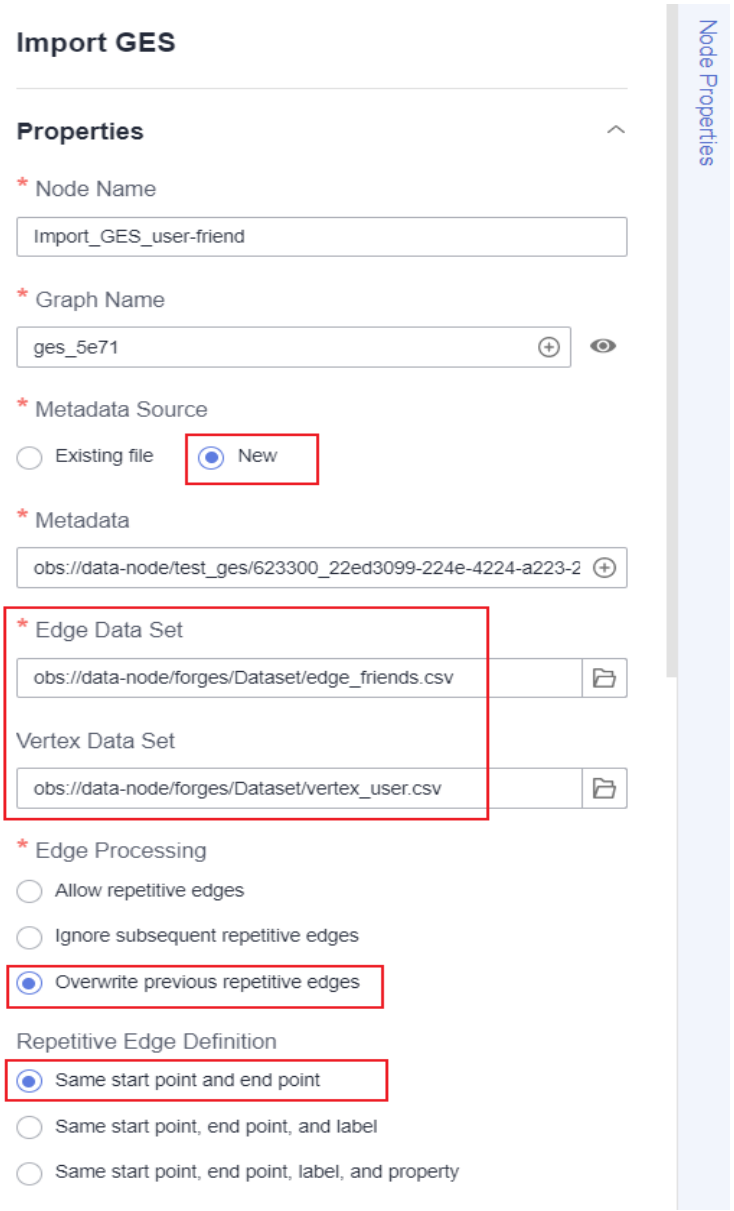
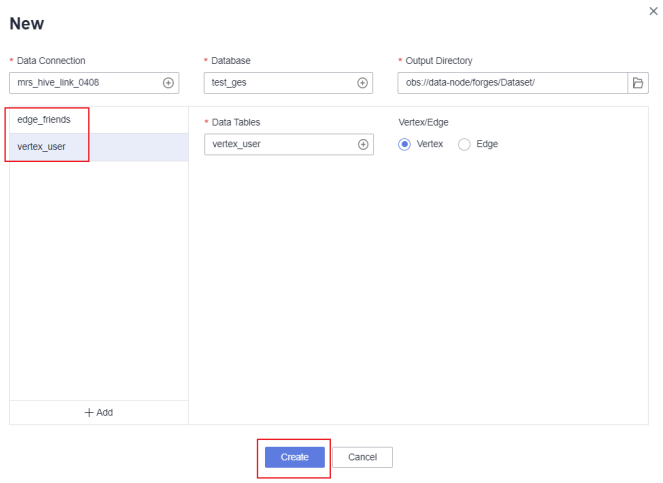
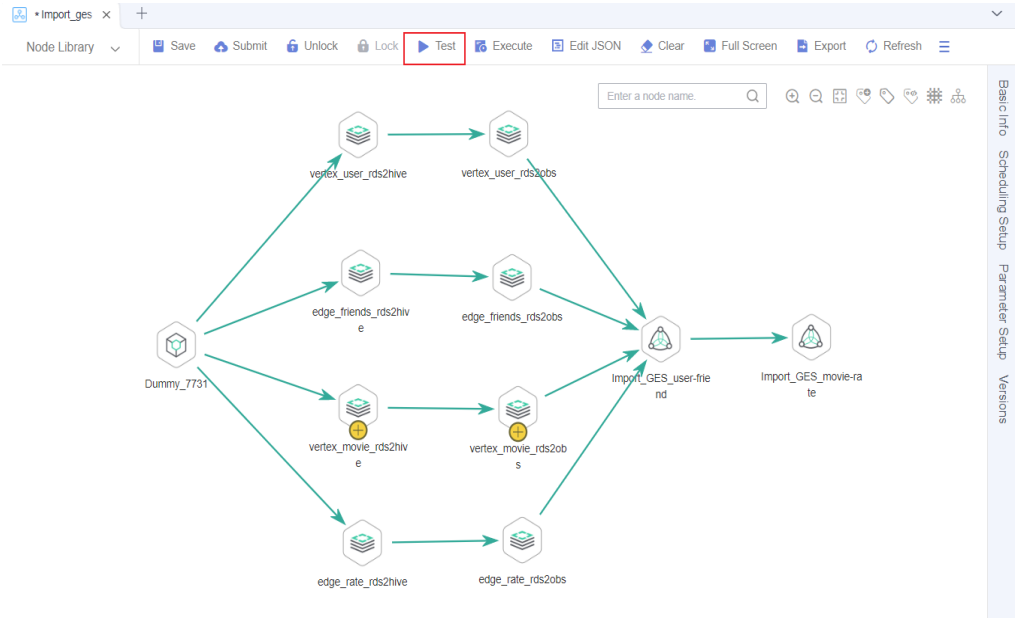


Figure 10-39 New



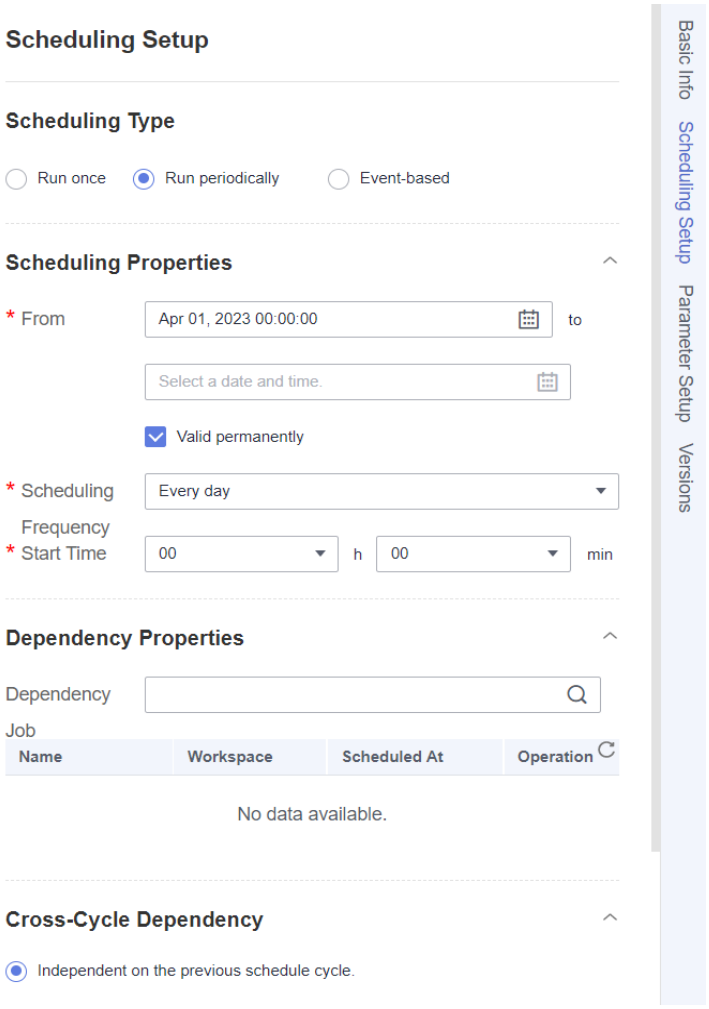
Step 6 After configuring the job, click to test it.

Figure 10-40 Testing the job



Step 7 If the job runs properly, click **Scheduling Setup** in the right pane and configure the scheduling policy for the job.

Figure 10-41 Configuring scheduling type



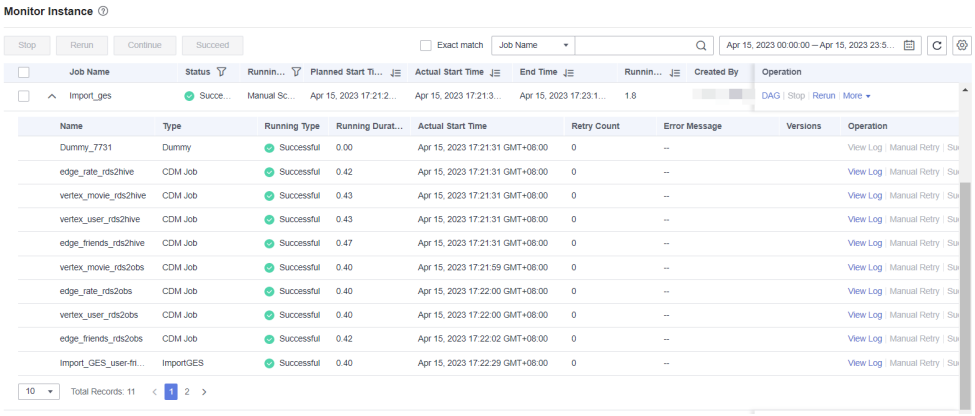
Parameter descriptions:

- **From**: From 00:00 on April 1, 2023, the job is executed at 00:00 every day.
- **Dependency Properties**: You can configure a dependency job for this job. You do not need to configure it in this practice.
- **Cross-Cycle Dependency**: Select **Independent on the previous schedule cycle**.

Step 8 Click to save and submit configuration and click to execute the scheduling job. The job will be automatically executed every day, and daily data will be automatically imported to GES.

Step 9 If you want to check the job execution result, choose **Monitoring** > **Monitor Instance** in the left navigation pane.

Figure 10-42 Viewing the job execution status



----End

10.5 Analyzing Graph Data

This section describes how to analyze graph data in a visualized manner using GES.

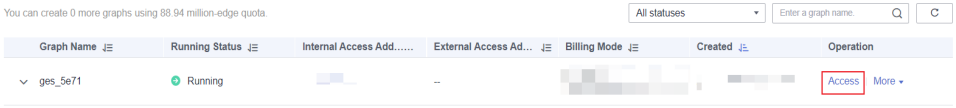
Prerequisites

Developing and Scheduling an Import GES Job has been completed and the job is running successfully.

Procedure

1. Log in to the GES console. Choose **Graph Management** in the navigation pane. On the displayed page, locate a graph, and click **Access** in the **Operation** column.

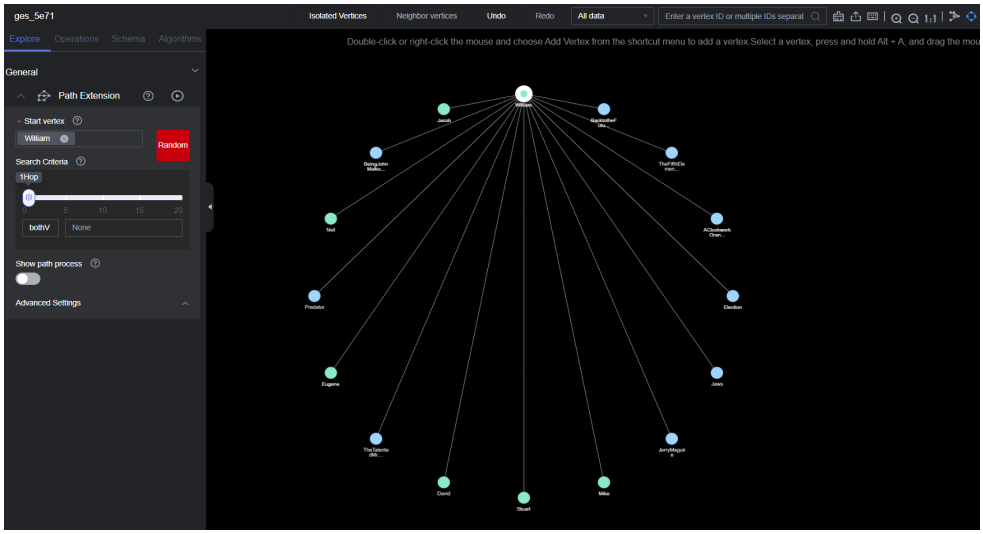
   Figure 10-43 Accessing a graph

2. Perform visualized analysis on the imported graph data by referring to **Accessing and Analyzing Graph Data**.

   The following example uses the graph exploration function to describe how to view information about users and movies related to the user William, as shown in **Figure 10-44**.

Figure 10-44 Analyzing graph 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 workforce and limited capabilities of its big data cluster. As a result, only 4% of 100 TB inventory data was useful.

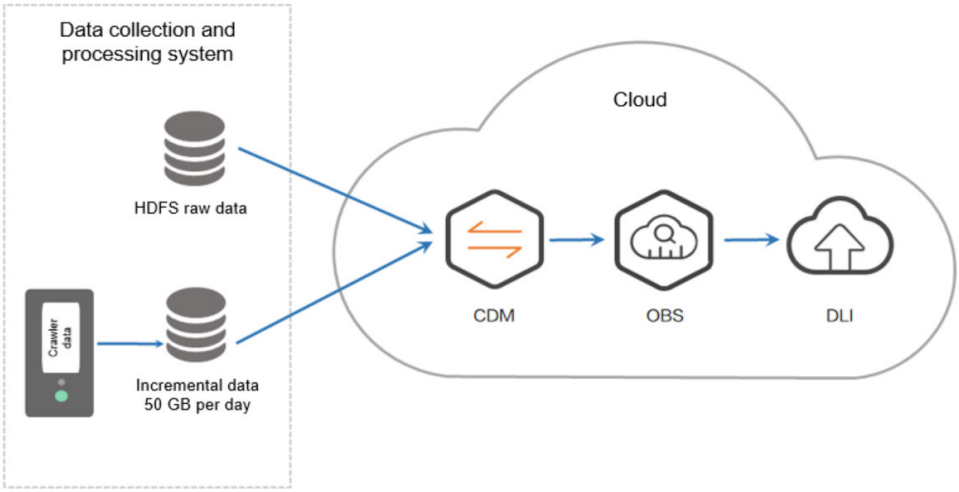
After migrating local trade statistics to Huawei Cloud, company H can make full use of the 100 TB inventory data in maximizing asset monetization, without the need of constructing and maintaining infrastructures but relying on Huawei Cloud's big data analysis capabilities.

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



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.

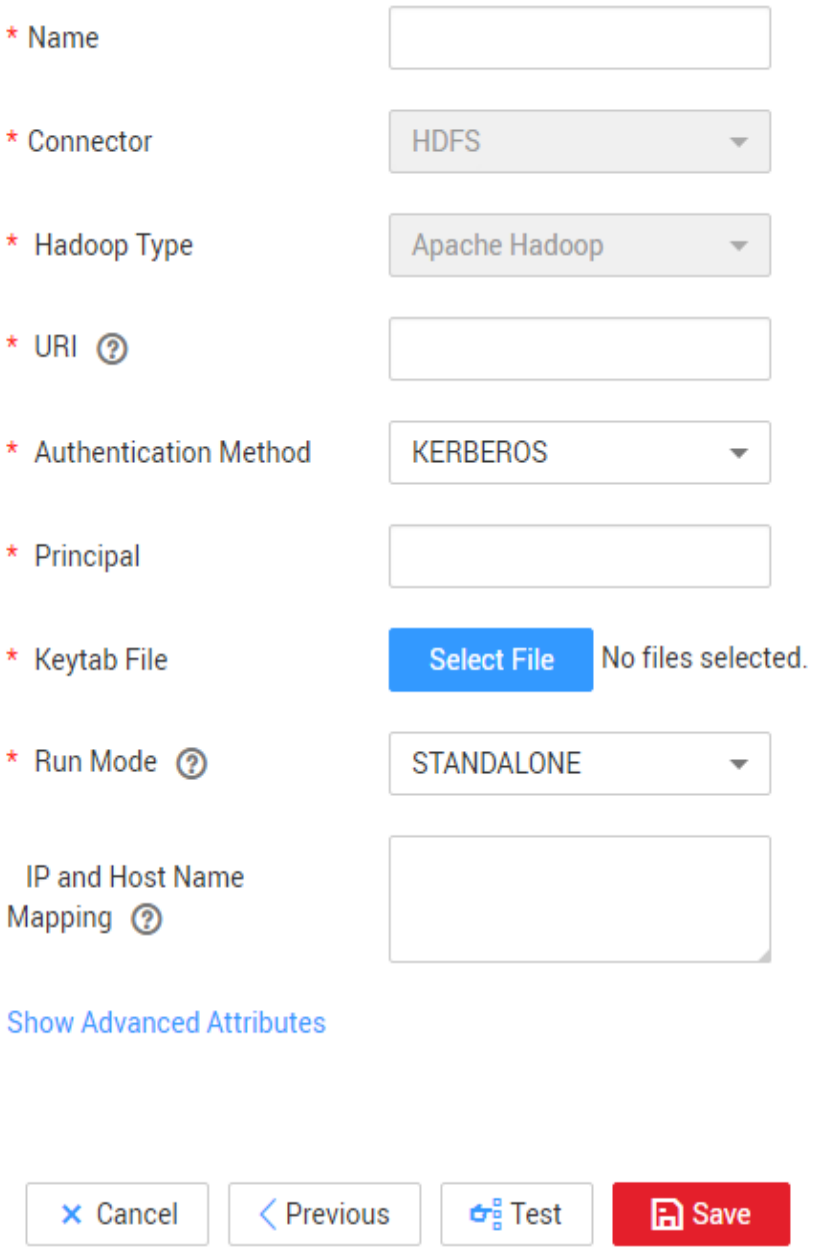
   If you need to create another CDM cluster, buy a DataArts Studio incremental package by referring to **Buying a DataArts Studio Incremental Package**.

   - **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



NOTE

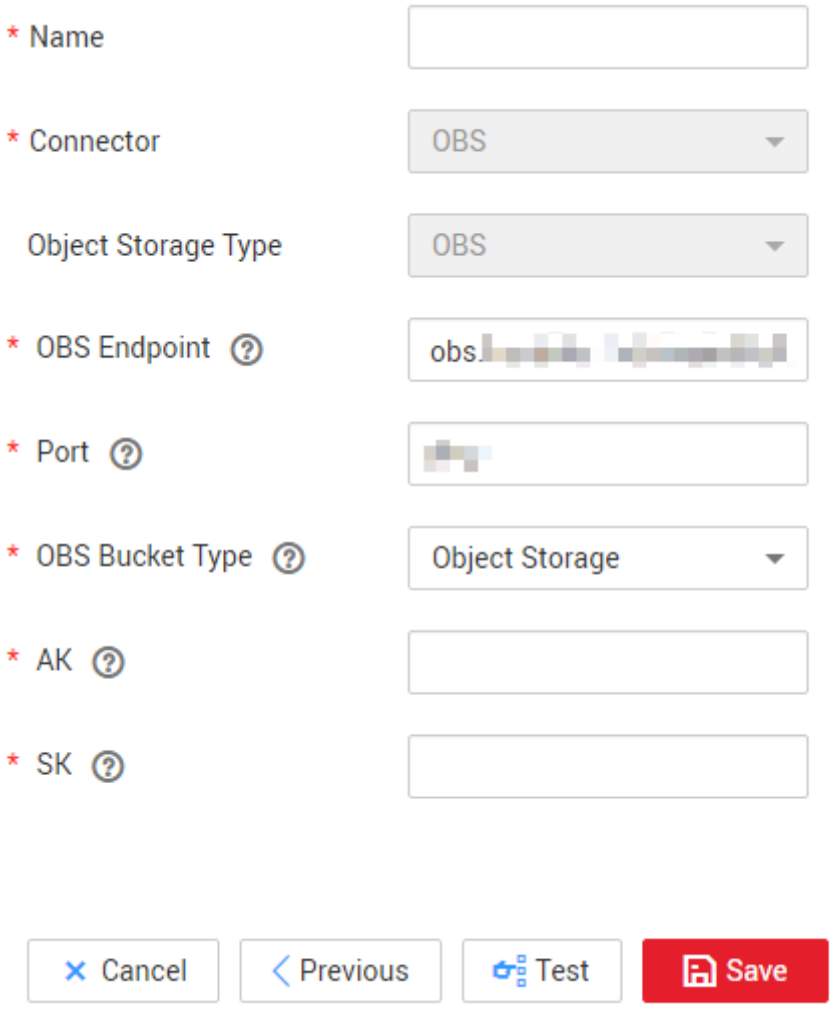
- **Name**: Enter a custom link name, for example, **hdfs_link**.
- **URI**: Enter the NameNode URI of HDFS of company *H*.
- **Authentication Method**: Select **KERBEROS** if Hadoop is in security mode to obtain the **principal** and **keytab** files from the client for authentication.
- **Principal** 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

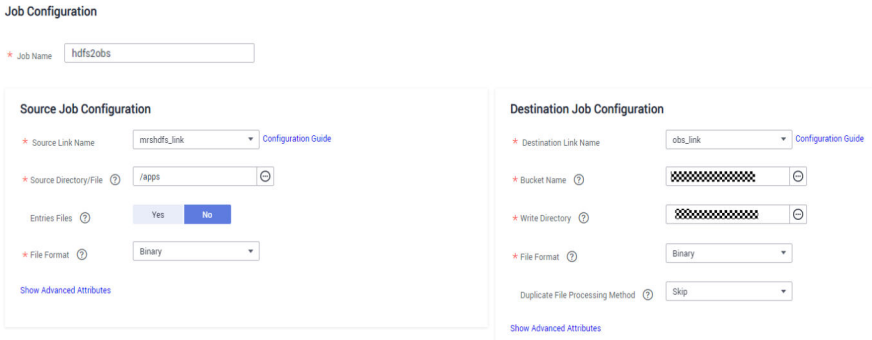


NOTE

- **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



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.

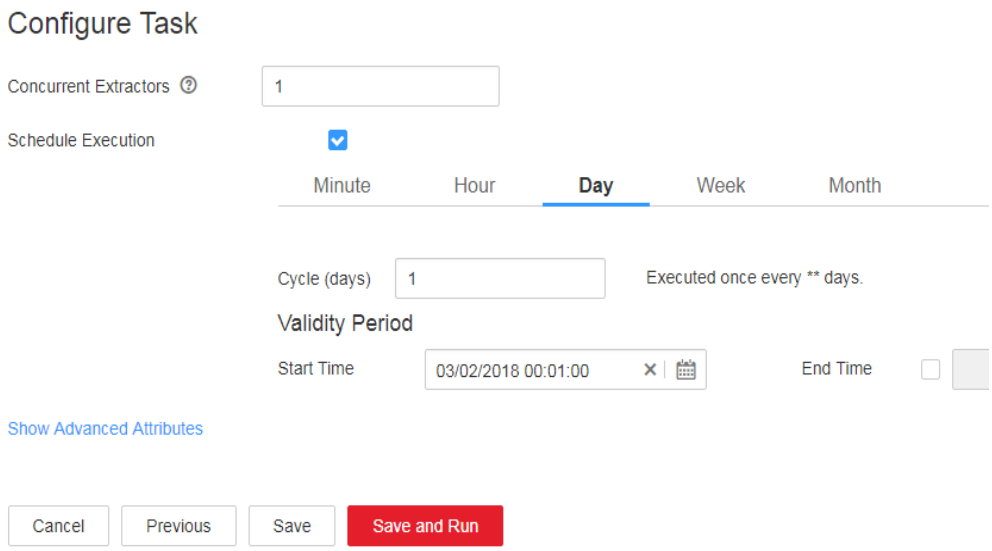
10. Click **Next** to go to the tab page for configuring the task parameters. For the migration of inventory data, retain the default values of the parameters.
11. Click **Save and Run**. On the displayed job management page, you can view the job execution progress and result.
12. After the job is successfully 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



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.

12 Case: IoV Big Data Service Migration to Cloud

12.1 Scenario

Background

Company *H* intends to build an enterprise-class cloud management platform for its IoV service to centrally manage and deploy hardware resources and common software resources, and implement cloud-based and service-oriented transformation of IT applications. Cloud Data Migration (CDM) helps company *H* build the platform without code modification and data loss.

Constraints

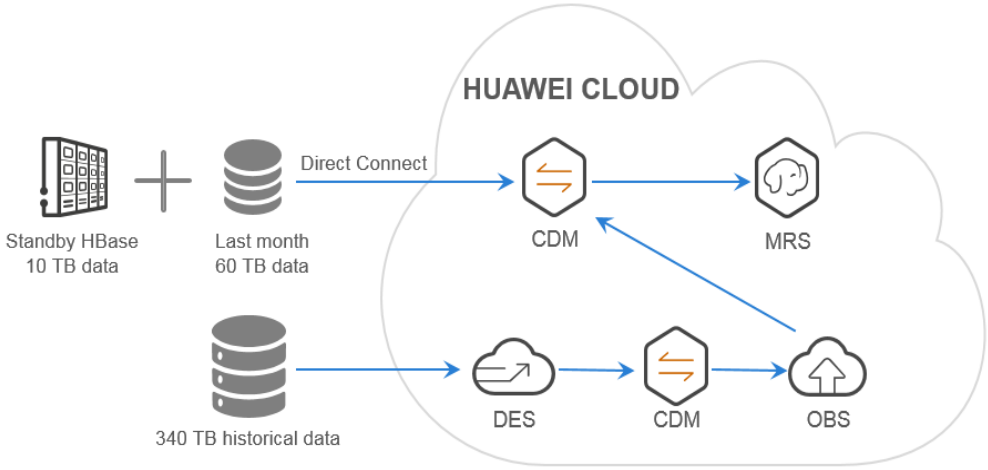
This solution supports only data migration to MRS 1.x clusters. In MRS 2.x and later versions, HBase tables cannot be rebuilt by running HBase repair commands.

NOTICE

If the target cluster version is 2.x or later, the HBase repair command is no longer supported, and the HBase data directory migration cannot be implemented.

Migration Scheme

Figure 12-1 Migration scheme



Company *H* stores 854 tables (400 TB) in the Cloudera Hadoop (CDH) HBase cluster and 149 tables (about 10 TB) in the standby HBase cluster. An amount of 60 TB data is increased in the last month.

Use CDM to extract HBase HFiles from the CDH cluster and save the extracted data to MapReduce Service (MRS) HDFS, and run the HBase repair command to rebuild the HBase table. Based on this migration scheme, the following two migration modes are optional:

1. CDM migrates data of the last month and data of the standby HBase cluster through the private line.

   CDH → CDM (HUAWEI CLOUD) → MRS

   NOTE

   The advantage and disadvantage of direct migration using the private line are as follows:

   - Advantage: Data does not need to be migrated multiple times, which shortens the overall migration duration.
   - Disadvantage: When a large amount of data is transmitted, the private line bandwidth is heavily occupied, which affects the concurrent services of the customer and crosses multiple switches.

2. Use CDM to migrate historical data generated one month ago from **Data Express Service (DES)**. The migration path is as follows:

   CDH → DES → CDM (HUAWEI CLOUD) → OBS → CDM (HUAWEI CLOUD) → MRS

   NOTE

   DES is well suited to the scenario where a large amount of data is to be transmitted, no private line is set up between the private cloud and HUAWEI CLOUD, and the bandwidth from the private cloud network to the public network is limited.

   - Advantage: The transmission is highly reliable without depending on the private line and network quality.
   - Disadvantage: The migration takes a long time.

12.2 Migration Preparation

Prerequisites

- The CDH HBase version is earlier than or equal to the MRS HBase version.
- No write, split, or merge operations can be performed on the tables that are being migrated.
- A **Direct Connect** connection has been established between the CDH cluster and Huawei Cloud Virtual Private Cloud (VPC).

Migration Process

1. Estimate the amount of data to be migrated and the migration duration.
2. Output the detailed data tables, the number and sizes of files to be migrated for subsequent verification.
3. Configure migration tasks in batches to ensure the migration progress and speed.
4. Check the number and sizes of files.
5. Restore the HBase table on MRS and verify the restoration.

Required Information

Item	Information	Description	Example Value
DES Teleport	Mount address	Address to which the DES Teleport box is mounted on the customer's VM	//*IP address of the VM*/huawei
	DeviceManager	Storage management system of the DES Teleport box, which is related to the management IP address	https://*Management IP address*:8088/deviceManager/devicemanager/login/login.html
	Username	Username for logging in to DeviceManager	admin
	Password	Password for logging in to DeviceManager	-
CDH cluster	NameNode IP	IP address of the active NameNode in the CDH cluster	192.168.2.3
	HDFS port	The default port number is 9000.	9000

Item	Information	Description	Example Value
	HDFS URI	NameNode URI of HDFS in the CDH cluster	hdfs://192.168.2.3:9000
OBS	OBS endpoint	Endpoint of OBS	obs.ap-southeast-1.myhuaweicloud.com
	OBS bucket	OBS bucket that stores historical data one month ago of the CDH cluster	cdm
	AK/SK	AK and SK for accessing OBS	-
MRS	Manager IP	IP address of MRS Manager	192.168.3.11

12.3 Using CDM to Migrate Data of the Last Month

The standby HBase cluster stores about 10 TB data, and the amount of data increased in the last month is about 60 TB. Therefore, the total amount of data is about 70 TB. Company *H*'s 20GE private line supports the cdm.xlarge cluster of CDM. Considering the migration duration, costs, and performance, two cdm.xlarge clusters are used to perform concurrent migrations. **Table 12-1** describes the cluster specifications.

Table 12-1 CDM cluster specifications

Instance Flavor	vCPUs/ Memory	Maximum/ Assured Bandwidth	Concurrent Extractors	Scenario
cdm.large	8 vCPUs and 16 GB memory	3/0.8 Gbit/s	16	A single table with 10 million or more than 10 million pieces of data
cdm.xlarge	16 vCPUs and 32 GB memory	10/4 Gbit/s	32	TB-level data migration requiring 10GE bandwidth
cdm.4xlarge	64 vCPUs and 128 GB memory	40/36 Gbit/s	64	-

> **NOTE**
>
> You can use multiple CDM clusters to perform migrations concurrently to improve migration efficiency. The MRS HDFS multi-replica policy occupies network bandwidth and affects the migration efficiency.

Creating Links on HUAWEI CLOUD CDM

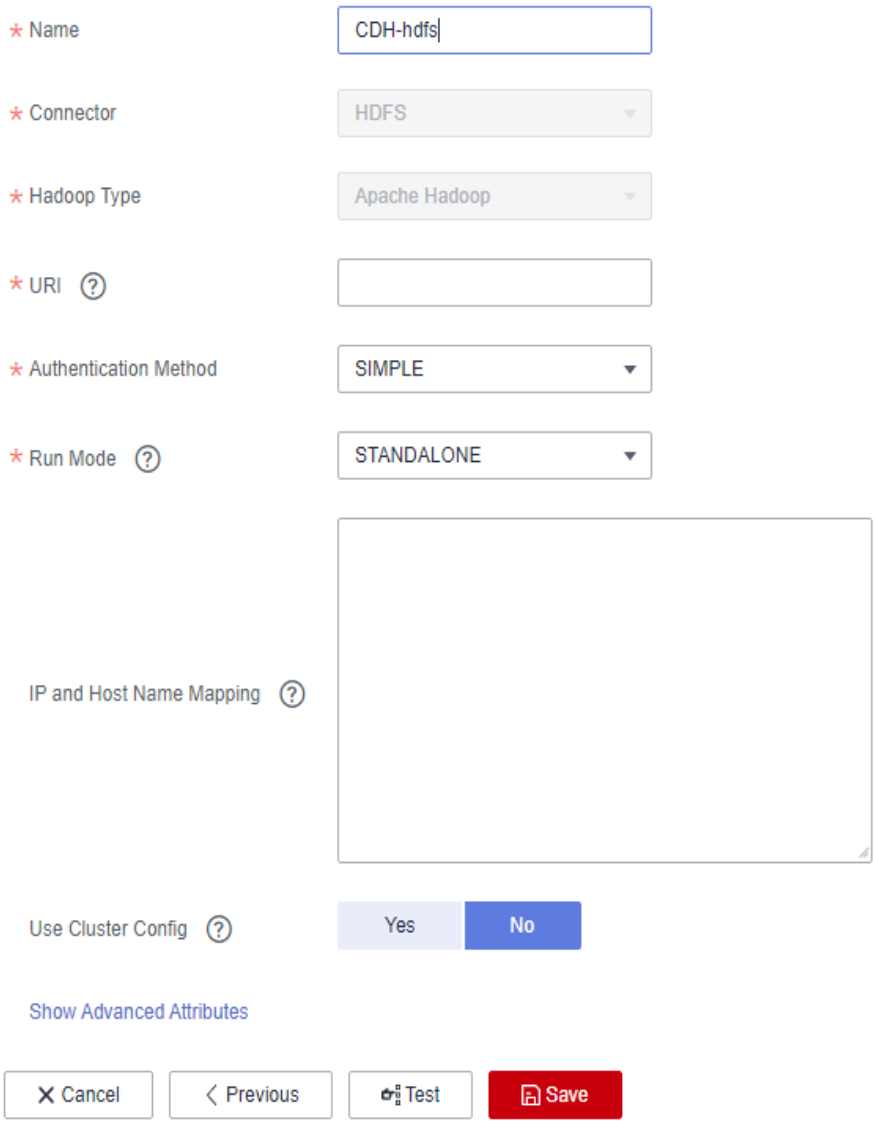
1. Create two CDM clusters.

   > **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**.

   - Select the **cdm.xlarge** flavor.
   - The clusters must reside in the same VPC as MRS and DirectConnect.
   - Configure other parameters as required or retain the default values.

2. Perform the following operations to create a CDH HDFS link:
   a. In the **Operation** column, click **Job Management**.
   b. Click the **Links** tab and then **Create Link**. On the page that is displayed, select **Apache HDFS**.

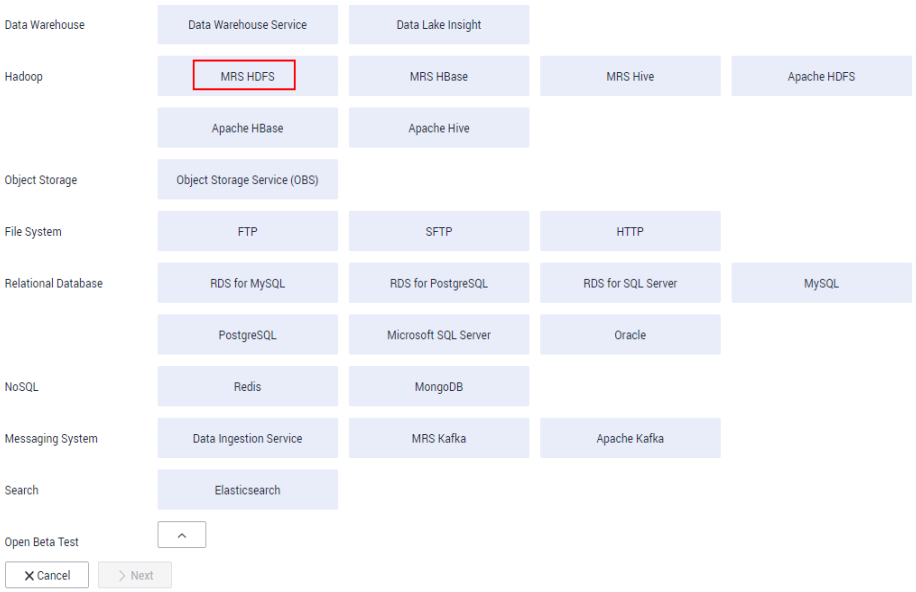
   c. Click **Next** and configure the link parameters. The URI format is *hdfs://NameNode IP address:Port number*. If Kerberos authentication is not enabled in the CDH cluster, set **Authentication Method** to **SIMPLE**.



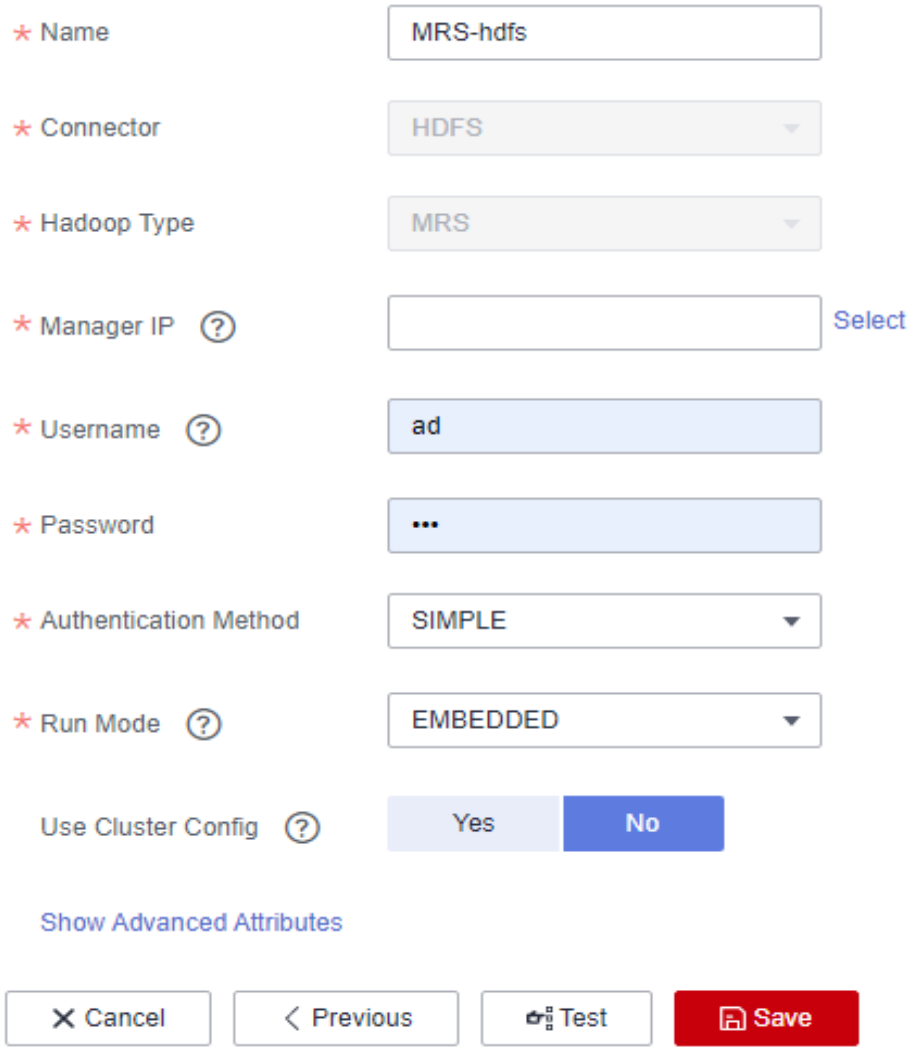
d. Click **Test**. If a test success message is displayed in the upper right corner, the link works properly. Click **Save**.

3. Perform the following operations to create an MRS HDFS link:

   a. Choose **Link Management** > **Create Link**. On the page that is displayed, select **MRS HDFS**.



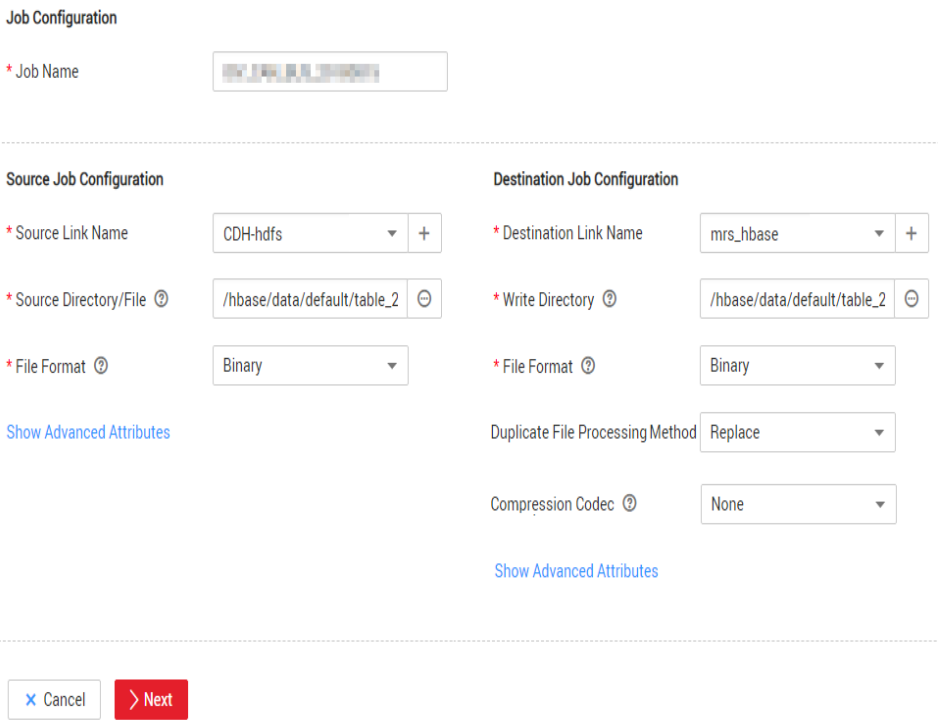
b. Click **Next** and configure the link parameters. Set **Authentication Method** to **SIMPLE** and retain the default run mode.



c. Click **Test**. If a test success message is displayed in the upper right corner, the link works properly. Click **Save**.

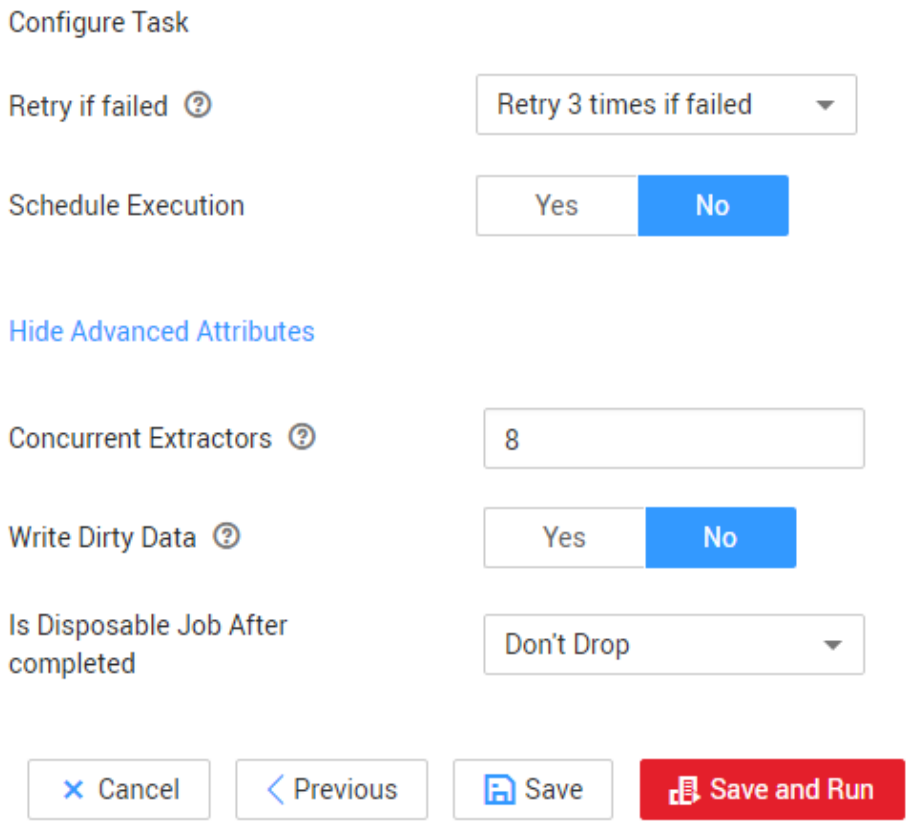
Creating a Migration Job on HUAWEI CLOUD CDM

1. On the job management page of the CDM cluster, choose **Table/File Migration** > **Create Job** to create jobs. Create a migration job for each table file directory.

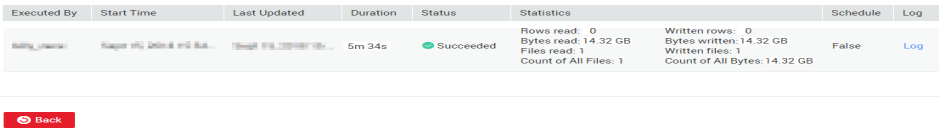


   - **Source Job Configuration**
     - **Source Link Name**: Select the created **CDH HDFS link**.
     - **Source Directory/File**: Select the directory where the HBase table of the CDH cluster resides. For example, **/hbase/data/default/table_20180815** indicates that all files in the **table_20180815** directory will be migrated.
     - **File Format**: Select **Binary** for copying files.
   - **Destination Job Configuration**
     - **Destination Link Name**: Select the created **MRS HDFS link**.
     - **Write Directory**: Select the MRS HBase directory, for example, **/hbase/data/default/table_20180815/**. The directory must carry a table name (for example, **table_20180815**). If the directory does not exist, CDM automatically creates it.
     - **File Format**: Select **Binary**.
   - Retain the default values of other parameters.
2. Click **Next** to configure the task. By default, **Concurrent Extractors** is **3**. You can increase the number of concurrent extractors (set it to **8** in this example)

to improve the migration efficiency. Retain the default values of other parameters.



3. Repeat the preceding operations to create migration jobs for other directories. The parameter settings are the same. The number of jobs in the two CDM clusters is evenly allocated and executed concurrently.
4. After a job is executed, you can view the detailed statistics by clicking **Historical Record** in the **Operation** column.



12.4 Using DES to Migrate Historical Data Generated One Month Ago

Migration Process

1. Use a script to import the historical data generated one month ago to the DES Teleport box. For details about the operations related to DES Teleport boxes, see **Data Express Service (DES)**.
2. Use DES to deliver data to HUAWEI CLOUD data center.
3. Use CDM to migrate data from DES to Object Storage Service (OBS).
4. Use CDM to migrate data from OBS to MRS.

The operations on CDM are the same as those described in **Using CDM to Migrate Data of the Last Month**. File directories are transmitted in binary format and two clusters concurrently execute jobs.

Precautions

- If the migration affects the source HDFS cluster, manually stop the job.
- If a large number of jobs fail, perform the following operations:
  a. Check whether the DES Teleport box is fully written. If the Teleport box is fully written, clear the failed directories to ensure that the data written later is complete.
  b. Check the network connectivity.
  c. Check the source HDFS cluster. Check whether indicators are abnormal. If any indicator is abnormal, suspend the migration task.

12.5 Restoring the HBase Table on MRS

After the CDH HBase table directories are migrated to MRS HBase, you can run commands to restore the directories. For data that may change, create snapshots to ensure that the data remains unchanged, and then migrate and restore the data.

Constraints

This solution supports only data migration to MRS 1.x clusters. In MRS 2.x and later versions, HBase tables cannot be rebuilt by running HBase repair commands.

NOTICE

If the target cluster version is 2.x or later, the HBase repair command is no longer supported, and the HBase data directory migration cannot be implemented.

Running Commands to Restore the Data Remaining Unchanged

For example, to restore the **/hbase/data/default/table_20180811** table, perform the following operations:

1. Access the node where MRS Client is located, for example, **master1**.
2. Run the following command to switch to user **omm**:
   su – omm
3. Run the following command to load environment variables:
   source /opt/client/bigdata_env
4. Run the following command to change the directory permission:
   hdfs dfs –chown *omm:hadoop* **–R** */hbase/data/default/table_20180811*
   - **omm:hadoop**: Indicates the username. Replace it with the actual username.
   - **/hbase/data/default/table_20180811**: Indicates the path of the table.

5. Run the following command to restore metadata:

   hbase hbck –fixMeta *table_20180811*

6. Run the command to restore regions:

   hbase hbck –fixAssignments *table_20180811*

7. If the message "Status: OK" is displayed, the table is restored successfully.

Using Snapshots to Migrate and Restore the Data that May Change

1. Run the following command in the HBase shell of the source CDH cluster:

   flush <*table name***>**

2. Run the following command in the HBase shell of the source CDH cluster:

   compact *<table name>*

3. If the Snap function is not enabled in the table, run the following command to enable the function:

   hadoop dfsadmin -allowSnapshot $path

4. Run the following commands to create an HDFS snapshot named **s0**:

   hdfs dfs -createSnapshot *<snapshotDir>* [*s0*]

   hdfs dfs -createSnapshot test

5. CDM copies files to MRS using the HDFS snapshot. Configure the migration job on CDM as follows:
   - **Source Directory/File**: **/hbase/data/default/src_test/.snapshot/s0**
   - **Write Directory**: **/hbase/data/default/***Table name*

6. Run the **fixMeta** and **fixAssignments** commands to restore the table. For details, see **Running Commands to Restore the Data Remaining Unchanged**.

7. Run the following command to delete the snapshot from the CDH cluster:

   hdfs dfs -deleteSnapshot *<snapshotDir> s0*

Rectifying the Fault That Occurs When Restoring a Table

1. After the **fixMeta** command is executed, the error message "xx inconsistent" is displayed.

   The **fixMeta** command is used to check the consistency of metadata between HDFS and HBase. This is a normal situation. Proceed to run the **fixAssignments** command.

2. After the **fixAssignments** command is executed, the error message "xx inconsistent" is displayed.

   The **fixAssignments** command is used to restore all regions. Sometimes, some regions go online slowly. You can run the following command to check whether the HBase table is successfully restored:

   hbase hbck *Table name*

   If the message "Status: OK" is displayed, the HBase table is restored successfully.

3. After the **fixAssignments** command is executed, an error message is displayed, indicating that multiple regions have the same startkey and some regions overlap.

Run the following command:

hbase hbck –fixHdfsOverlaps *Table name*

Then run the **fixMeta** and **fixAssignments** commands.

13 Case: Building a Real-Time Alarm Platform

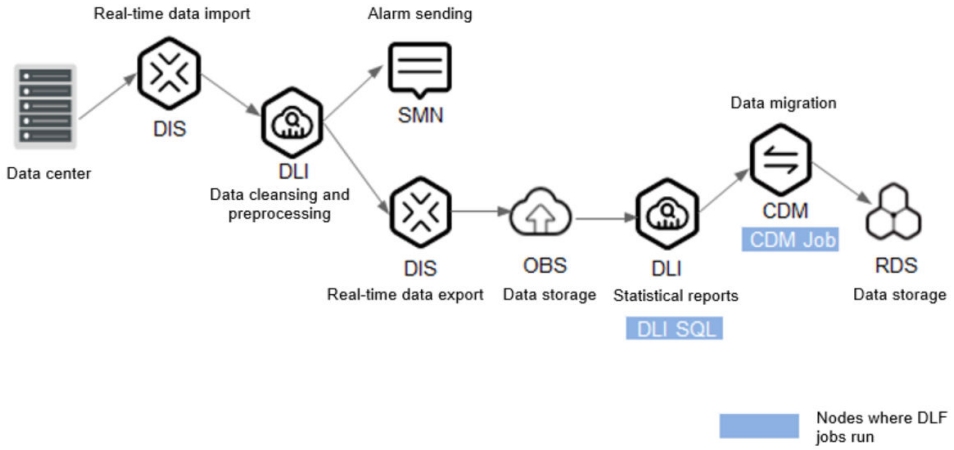
In this practice, you can learn how to set up a simple real-time alarm platform by using the job editing and scheduling functions of DataArts Factory, as well as other cloud services.

The following gives a sample scenario where a customer has deployed many applications in a data center and requires a unified O&M system to receive alarm information in real time and:

- To send a message to the user when the alarm severity is Major or higher.
- To provide an O&M report every day used for collecting statistics on alarm severity of each application.

The following solution is developed to meet the preceding requirements:

Figure 13-1 Solution design



The general procedure is as follows:

1. Real-time data import: Data Ingestion Service (DIS) is used to import alarm data from the data center to Data Lake Insight (DLI) in real time.
2. Data cleansing and preprocessing: DLI cleanses and preprocesses alarm data.

3. Alarm information sending: When the alarm severity exceeds the specified level, an SMS message is sent to the user.
4. Data export and storage: The cleaned data is exported from DLI to DIS. DIS imports the alarm data to buckets in Object Storage Service (OBS). All the buckets are created based on the import time.
5. Alarm statistic table output: The DLI SQL script is used to create an alarm statistic table.
6. Data migration: After the alarm statistic table is created, the data is exported to RDS for MySQL using Cloud Data Migration (CDM).

Environment preparations

- OBS has been enabled and buckets have been created, for example, **obs://dlfexample/alarm_info** and **obs://dlfexample/alarm_count_info**, which are used to store the raw alarm table and alarm statistic table, respectively.
- DataArts Studio has been enabled, and the **cdm-alarm** cluster is available for **Creating a CDM job**.
- DLI has been enabled.
- Simple Message Notification (SMN) has been enabled.

Data Preparation

The raw alarm table records the real-time data of the data center, including alarm IDs and alarm severity. **Table 13-1** shows the sample data.

Table 13-1 Raw data sample

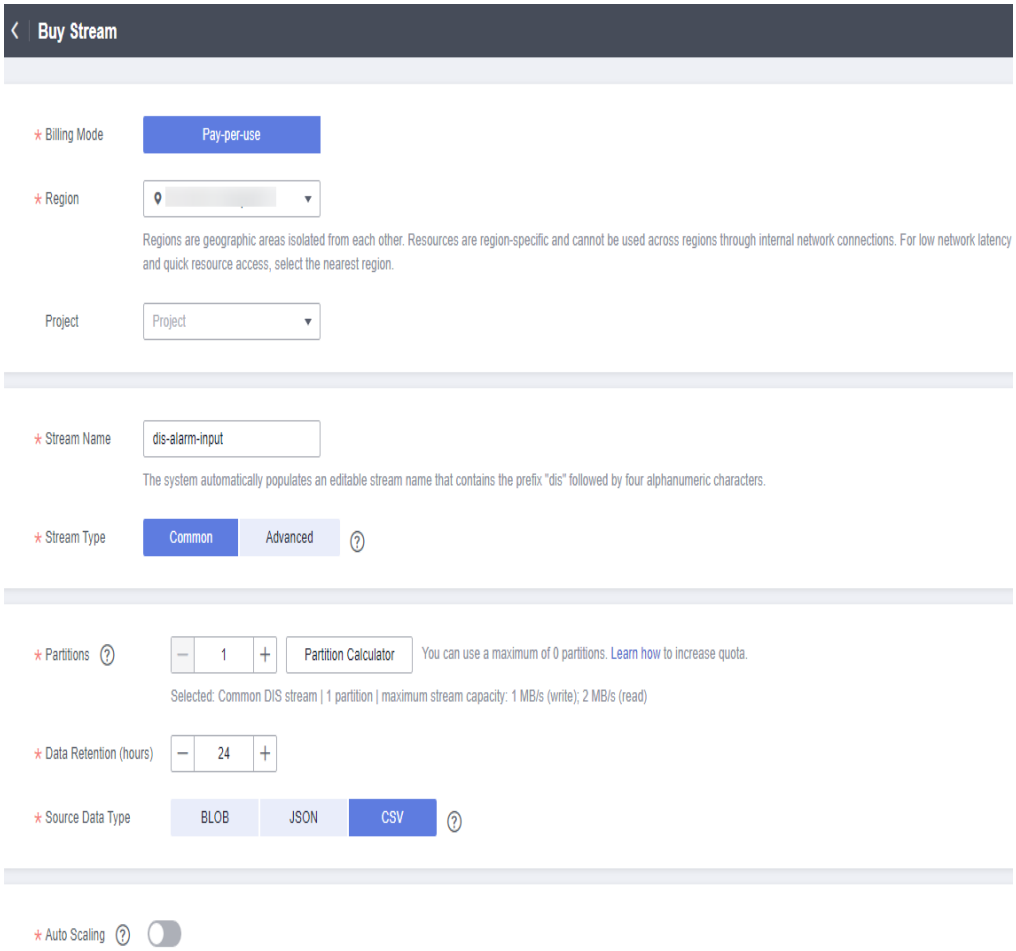
alarm_id	alarm_type
00440114	3
00440121	5
00440122	6
00440123	7
00440124	8
00440126	0

Creating a DIS Stream

Create two DIS streams on the DIS console, one for inputting real-time data to DLI and one for outputting real-time data to OBS.

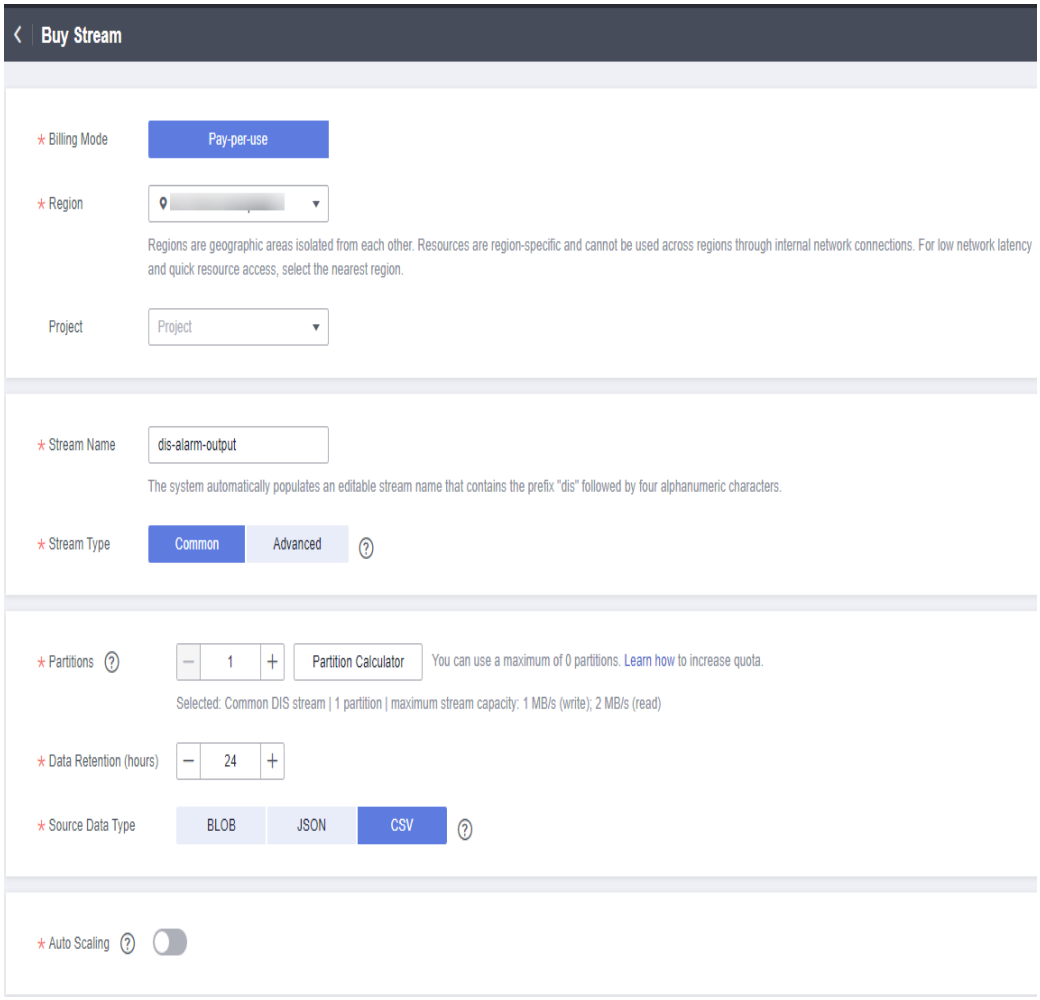
Step 1 Create a stream for inputting real-time data to DLI. The stream is named **dis-alarm-input**.

Figure 13-2 Creating an input stream



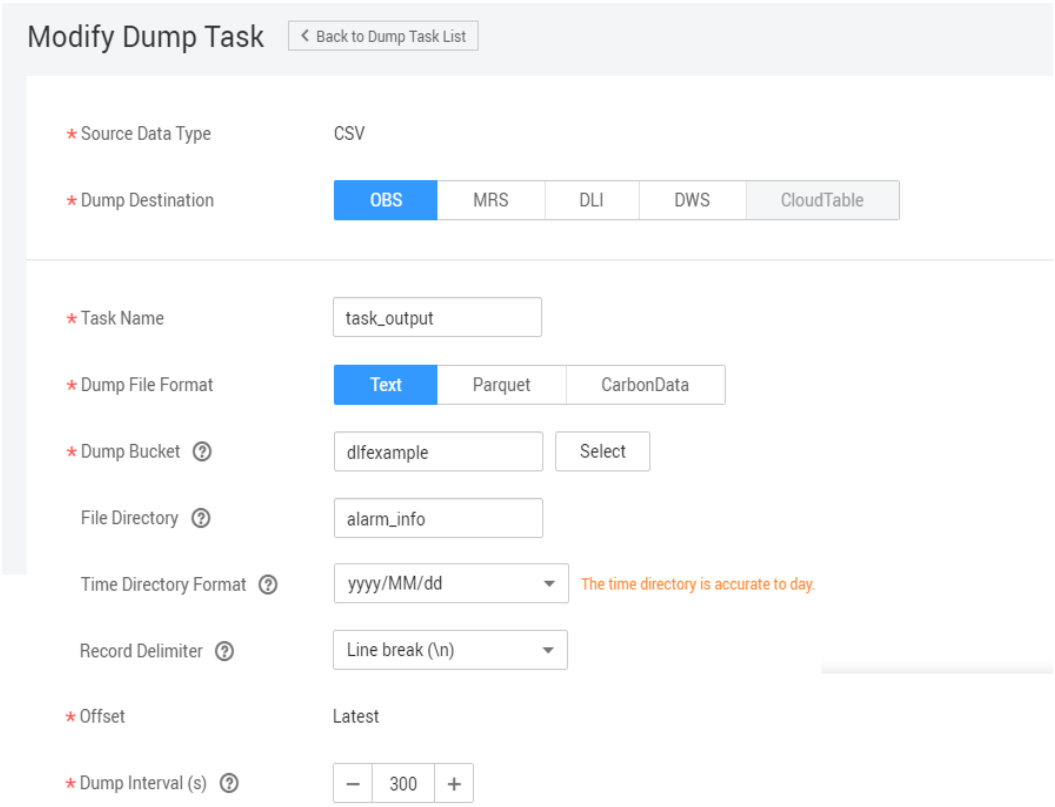
Step 2 Create a stream for outputting real-time data to OBS. The stream is named **dis-alarm-output**.

Figure 13-3 Creating an output stream



Configure a dump task for **dis-alarm-output** to dump the data in the stream to the **obs://dlfexample/alarm_info** directory of the OBS based on the output time.

Figure 13-4 Configuring a dump task for an output stream



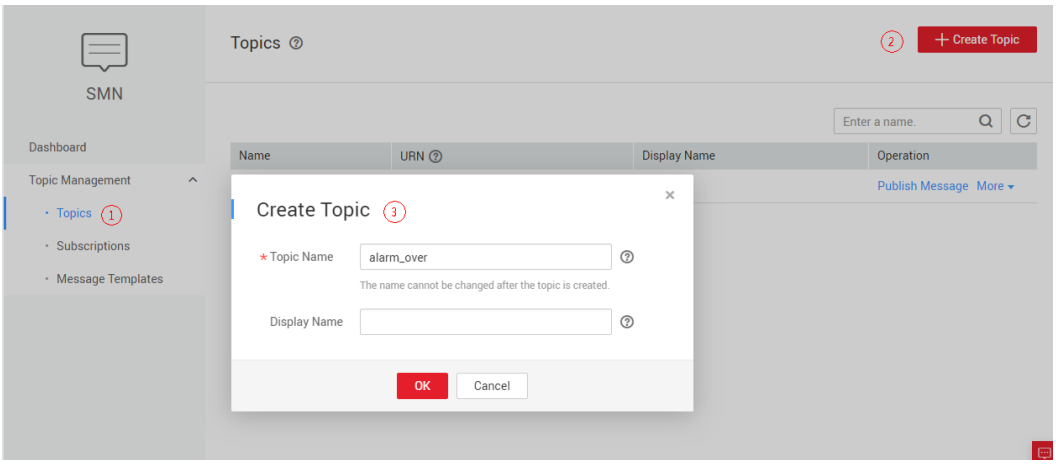
----End

Creating an SMN Topic

This section describes how to create an SMN topic, add a subscription to the topic, and add users who need to receive alarm notifications to a subscription terminal.

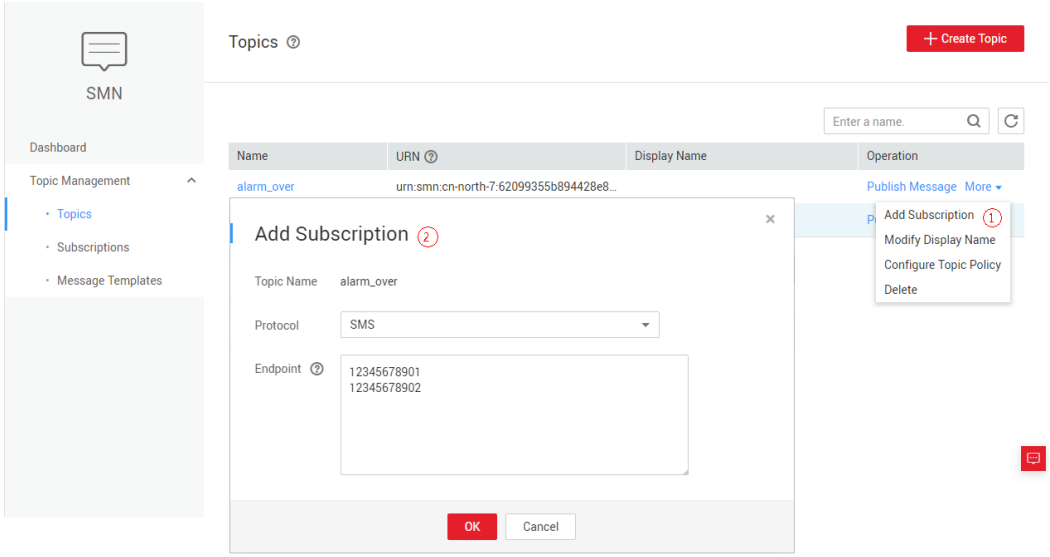
Step 1 Create an SMN topic named **alarm_over**.

Figure 13-5 Create SMN Topic



Step 2 Add a subscription to the topic, specifying alarm types and users who need to receive alarm notifications.

Figure 13-6 Adding a subscription



The key parameters are as follows:

- **Protocol**: Select **SMS**. After **SMS** is selected, an SMS notification will be sent when the alarm severity reaches the specified value.
- **Endpoint**: Enter the mobile number of the user who needs to receive alarm notifications.

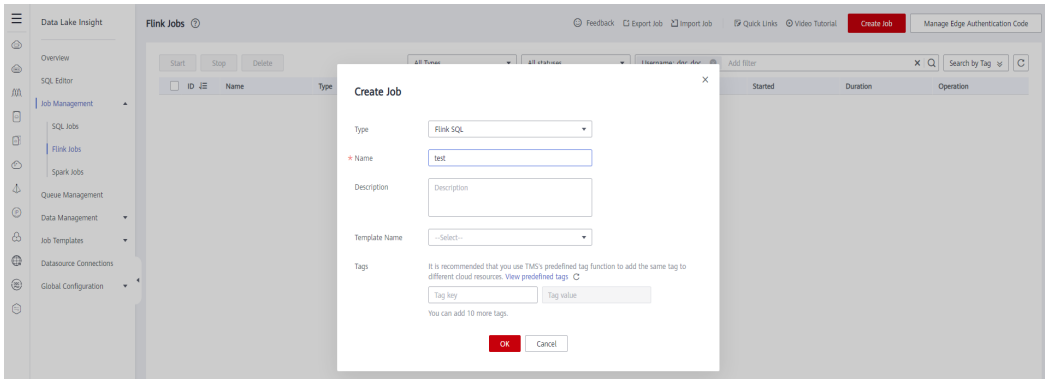
----End

Using DLI to Construct an Alarm Notification Project

After creating DIS streams and SMN topics, you can set up an alarm notification project in DLI. For details about how to create a DIS stream and SMN topic, see **Creating a DIS Stream** and **Creating an SMN Topic**, respectively.

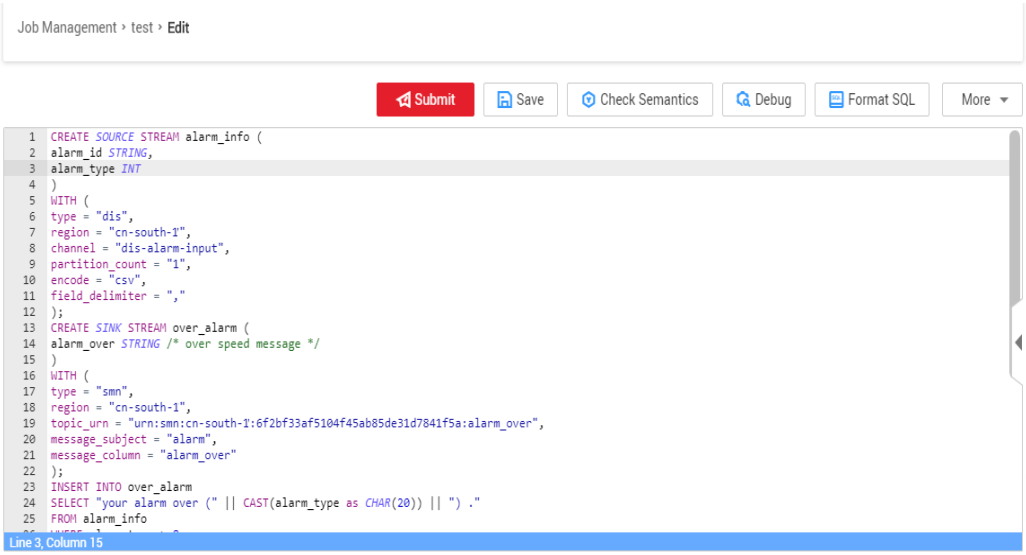
Step 1 Create a Flink job in DLI. The job is named **test**.

Figure 13-7 Creating a Flink SQL Job



Step 2 Edit the Flink SQL job and enter statements in the SQL editor.

Figure 13-8 Editing a Flink SQL job



Functions provided by running the SQL statements:

1. Upload real-time data from the **dis-alarm-input** created in **Step 1** to DLI.
2. Send an SMS notification when the alarm severity reaches the specified value.
3. Output the processed real-time data from DLI to OBS through **dis-alarm-output**.

```
CREATE SOURCE STREAM alarm_info (
  alarm_id STRING,
  alarm_type INT
)
WITH (
  type = "dis",
  region = "cn-south-1",
  channel = "dis-alarm-input",
  partition_count = "1",
  encode = "csv",
  field_delimiter = ","
);
CREATE SINK STREAM over_alarm (
  alarm_over STRING /* over speed message */
)
WITH (
  type = "smn",
  region = "cn-south-1",
  topic_urn = "urn:smn:cn-south-1:6f2bf33af5104f45ab85de31d7841f5a:alarm_over",
  message_subject = "alarm",
  message_column = "alarm_over"
);
INSERT INTO over_alarm
SELECT "your alarm over (" || CAST(alarm_type as CHAR(20)) || ") ."
FROM alarm_info
WHERE alarm_type > 8;
CREATE SINK STREAM alarm_info_output (
  alarm_id STRING,
  alarm_type INT
)WITH (
  type ="dis",
  region = "cn-south-1",
  channel = "dis-alarm-output",
  PARTITION_KEY = "alarm_type",
  encode = "csv",
  field_delimiter = ","
```



```
);
INSERT INTO alarm_info_output
SELECT *
FROM alarm_info
WHERE alarm_type > 0;
```

Step 3 After the Flink SQL job is created, save and start the job.

----End

Using a DLI SQL Script to Develop and Construct an Alarm Table

This section describes how to create an SQL script for creating an OBS table to store data tables on DLI and create one more SQL script for collecting alarm information.

Step 1 On the DataArts Studio **Management Center** page, create a data connection named **dli** to DLI.

Step 2 On the **DataArts Factory** page, create a database named **dlitest** in DLI to store data tables.

Step 3 Create a DLI SQL script used to create table **alarm_info** and **alarm_count_info** by entering SQL statements in the editor.

alarm_info and **alarm_count_info** are OBS tables used to store a raw alarm table and alarm statistic table, respectively.

Figure 13-9 Creating a data table

Description of key operations:

- The script development area in **Figure 13-9** is a temporary debugging area. After you close the script tab, the development area will be cleared. To retain the SQL script, click  to save the script to a specific directory.

The key parameters are as follows:

- **Data Connection**: DLI data connection created in **Step 1**.
- **Database**: database created in **Step 2**.
- **Resource Queue**: the default resource queue **default** provided by DLI.
- SQL statements:

```
create table  alarm_info(alarm_time string, alarm_id string, alarm_type int ) using csv options(path 'obs://dlfexample/alarm_info') partitioned by(alarm_time);
create table  alarm_count_info(alarm_time string, alarm_type int, alarm_count int) using csv options(path 'obs://dlfexample/alarm_count_info');
```

Step 4 Click  to run the script for creating the **alarm_info** and **alarm_count_info** data tables.

Step 5 Clear the editor and enter SQL statements again.

```
ALTER TABLE alarm_info  ADD PARTITION (alarm_time = ${dayParam})
LOCATION 'obs://dlfexample/alarm_info/${obsPathYear}';
insert into alarm_count_info
select alarm_time,alarm_type,count(alarm_type) from alarm_info where alarm_time = ${dayParam} group
by alarm_time,alarm_type;
```

Functions provided by running the SQL statements:

1. Create a DLI partition based on the date in the **obs://dlfexample/alarm_info** directory of OBS. If the current date is 2018-10-10, create a DLI partition named **2018/10/09** to store the data tables generated on **2018/10/09** to the **obs://dlfexample/alarm_info** directory.
2. Collect statistics based on the alarm partition time and alarm type, and insert the statistic result into the **alarm_count_info** table.

The key parameters are as follows:

- ${dayParam}: **dayParam** indicates the partition value of the **alarm_info** table. Enter **$getCurrentTime(@@yyyyMMdd@@,-24*60*60)** in the lower part of the script editor.
- ${obsPathYear}: **obsPathYear** indicates the directory of the OBS partition. Enter **$getCurrentTime(@@yyyy/MM/dd@@,-24*60*60)** in the lower part of the script editor.

Step 6 After the script is debugged, save the script. The script name is **dli_partition_count**. In subsequent operations, set the script to be executed periodically so that an alarm statistics table can be exported periodically. For details about how to export alarm statistics tables periodically, see **Exporting Alarm Statistics Tables Periodically**.

----End

Creating a CDM job

This section describes how to use CDM to migrate the alarm statistic table from OBS to RDS for MySQL.

The key parameters are as follows:

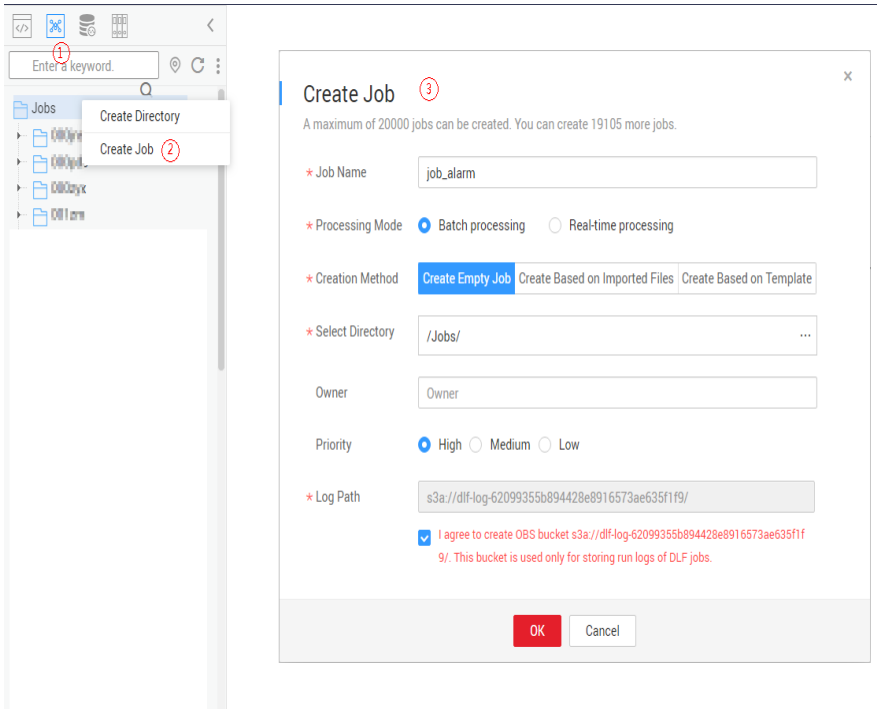
- **Job Name**: obs_rds. In subsequent operations, set the job to be periodically executed so that data will be migrated periodically. For details about how to export alarm statistics tables periodically, see **Exporting Alarm Statistics Tables Periodically**.
- **Source Job Configuration**: OBS directory for storing alarm statistic tables. The source link **obs_link** must be created in CDM in advance.
- **Destination Job Configuration**: RDS MySQL space for storing alarm statistic tables. The destination link **mysql_link** must be created in CDM in advance.

Exporting Alarm Statistics Tables Periodically

After the alarm statistic table script and data migration CDM job are created, you can create a job in DataArts Factory and execute the job periodically so that you can export alarm statistic tables and migrate data periodically.

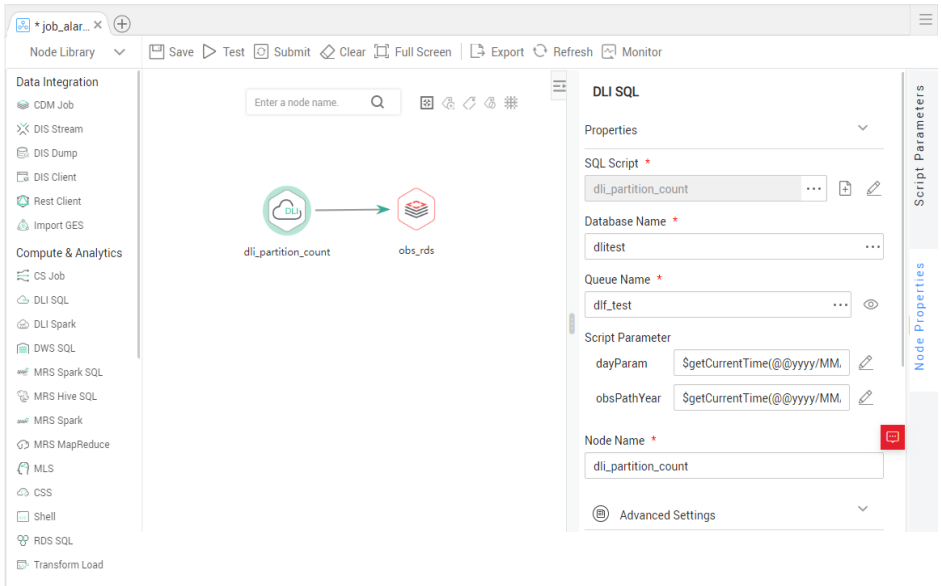
Step 1 Create a batch job named **job_alarm**.

Figure 13-10 Creating a DLF Job



Step 2 On the job development page, drag the DLI SQL and CDM Job nodes to the canvas, connect the nodes, and then click them to configure node properties.

Figure 13-11 Connecting nodes and configuring node properties



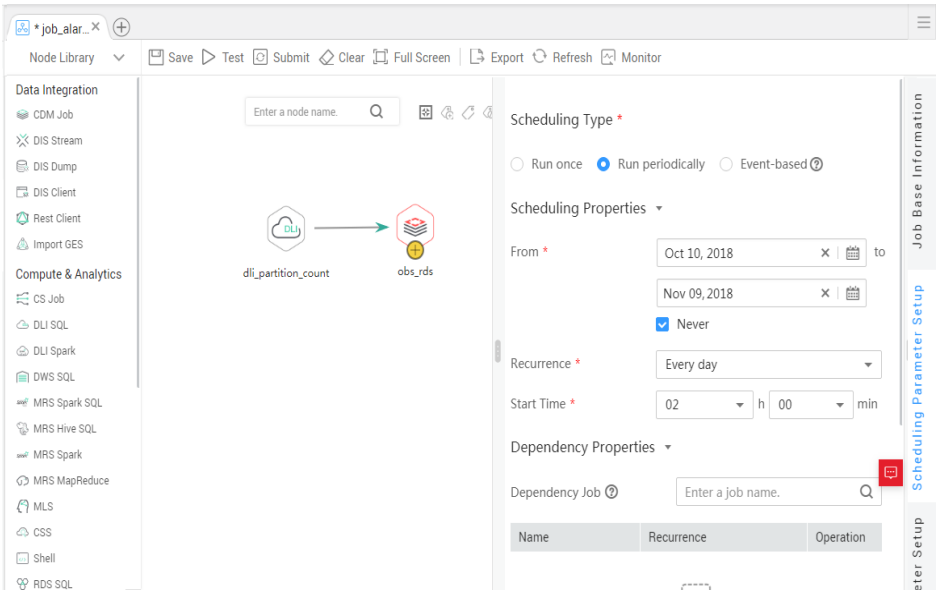
Key notes:

- **dli_partition_count** (DLI SQL node): In **Node Properties**, associates with the DLI SQL script **dli_partition_count** developed in **Using a DLI SQL Script to Develop and Construct an Alarm Table**.
- **obs_rds** (CDM Job node): In **Node Properties**, associates with the CDM job **obs_rds** developed in **Creating a CDM job**.

Step 3 After configuring the job, click  to test it.

Step 4 If the log shows that the job runs properly, click **Scheduling Setup** in the right pane and configure the scheduling policy for the job.

Figure 13-12 Configuring scheduling type



Parameter descriptions:

- 2018/10/10 to 2018/11/09: The job is executed at 02:00 a.m. every day.

Step 5 Save and submit the job. Then click  to execute the job so that it runs automatically every day.

----End

